



Hewlett-Packard Company

TPC Benchmark™ C

Full Disclosure Report
for

HP ProLiant DL580 - PDC 32P

Using

Oracle9i Enterprise Edition Release 2 with Real Application
Clusters and Partitioning Options and
Red Hat Linux Advanced Server

Third Edition
October 2002

Third Edition – October 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.

Copyright 2002 Hewlett Packard Company.

All rights reserved. Permission is hereby granted to reproduce this document in whole or in part provided the copyright notice printed above is set forth in full text or on the title page of each item reproduced.

Printed in U.S.A., 2002

Parallel Database Cluster Model PDC 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.

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.

The result was found to have an “Insignificant Deviation” from the specification by the FDR missing some documentation. These files are now included in the appendix.

Table of Contents

TABLE OF CONTENTS	1
PREFACE	3
TPC BENCHMARK C OVERVIEW.....	3
ABSTRACT	4
OVERVIEW	4
TPC BENCHMARK C METRICS.....	4
STANDARD AND EXECUTIVE SUMMARY STATEMENTS	4
AUDITOR.....	4
GENERAL ITEMS	8
APPLICATION CODE AND DEFINITION STATEMENTS	8
TEST SPONSOR.....	8
PARAMETER SETTINGS.....	8
CONFIGURATION ITEMS.....	8
CLAUSE 1 RELATED ITEMS	10
TABLE DEFINITIONS	10
PHYSICAL ORGANIZATION OF DATABASE.....	10
<i>Priced Configuration:</i>	11
INSERT AND DELETE OPERATIONS	11
PARTITIONING	11
REPLICATION, DUPLICATION OR ADDITIONS	11
CLAUSE 2 RELATED ITEMS	12
RANDOM NUMBER GENERATION	12
INPUT/OUTPUT SCREEN LAYOUT	12
PRICED TERMINAL FEATURE VERIFICATION.....	12
PRESENTATION MANAGER OR INTELLIGENT TERMINAL.....	12
TRANSACTION STATISTICS	13
QUEUING MECHANISM.....	13
CLAUSE 3 RELATED ITEMS	14
TRANSACTION SYSTEM PROPERTIES (ACID).....	14
ATOMICITY	14
<i>Completed Transactions</i>	14
<i>Aborted Transactions</i>	14
CONSISTENCY	14
ISOLATION.....	14
DURABILITY	14
<i>Durable Media Failure</i>	14
<i>Loss of Data</i>	15
<i>Loss of Log</i>	15
<i>Instantaneous Interruption, Loss of Memory (8 Nodes)</i>	16
<i>Instantaneous Interruption, Loss of Memory (1 Node)</i>	16
<i>Loss of Cluster interconnect</i>	17
CLAUSE 4 RELATED ITEMS	18

INITIAL CARDINALITY OF TABLES	18
DATABASE LAYOUT	18
TYPE OF DATABASE	18
DATABASE MAPPING.....	19
60 DAY SPACE	19
CLAUSE 5 RELATED ITEMS.....	20
THROUGHPUT.....	20
RESPONSE TIMES	20
KEYING AND THINK TIMES	20
RESPONSE TIME FREQUENCY DISTRIBUTION CURVES AND OTHER GRAPHS	21
STEADY STATE DETERMINATION	26
WORK PERFORMED DURING STEADY STATE	26
MEASUREMENT PERIOD DURATION	26
REGULATION OF TRANSACTION MIX	26
TRANSACTION STATISTICS	27
CHECKPOINT COUNT AND LOCATION	27
CHECKPOINT DURATION	27
CLAUSE 6 RELATED ITEMS.....	29
RTE DESCRIPTIONS	29
EMULATED COMPONENTS	29
FUNCTIONAL DIAGRAMS	29
NETWORKS	29
OPERATOR INTERVENTION.....	29
CLAUSE 7 RELATED ITEMS.....	30
SYSTEM PRICING.....	30
AVAILABILITY, THROUGHPUT, AND PRICE PERFORMANCE	30
COUNTRY SPECIFIC PRICING	30
USAGE PRICING.....	30
CLAUSE 9 RELATED ITEMS.....	31
AUDITOR’S REPORT	31
AVAILABILITY OF THE FULL DISCLOSURE REPORT	31
.....	32
APPENDIX A: SOURCE CODE.....	34
APPENDIX B: DATABASE DESIGN	172
APPENDIX C: TUNABLE PARAMETERS.....	203
APPENDIX D: THIRD PARTY LETTERS	209
APPENDIX G:.....	214

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 - PDC 32P. The operating system used for the benchmark was Linux 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:

138362.03 tpmC
\$17.38 per tpmC
Available as March 5, 2003.

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.

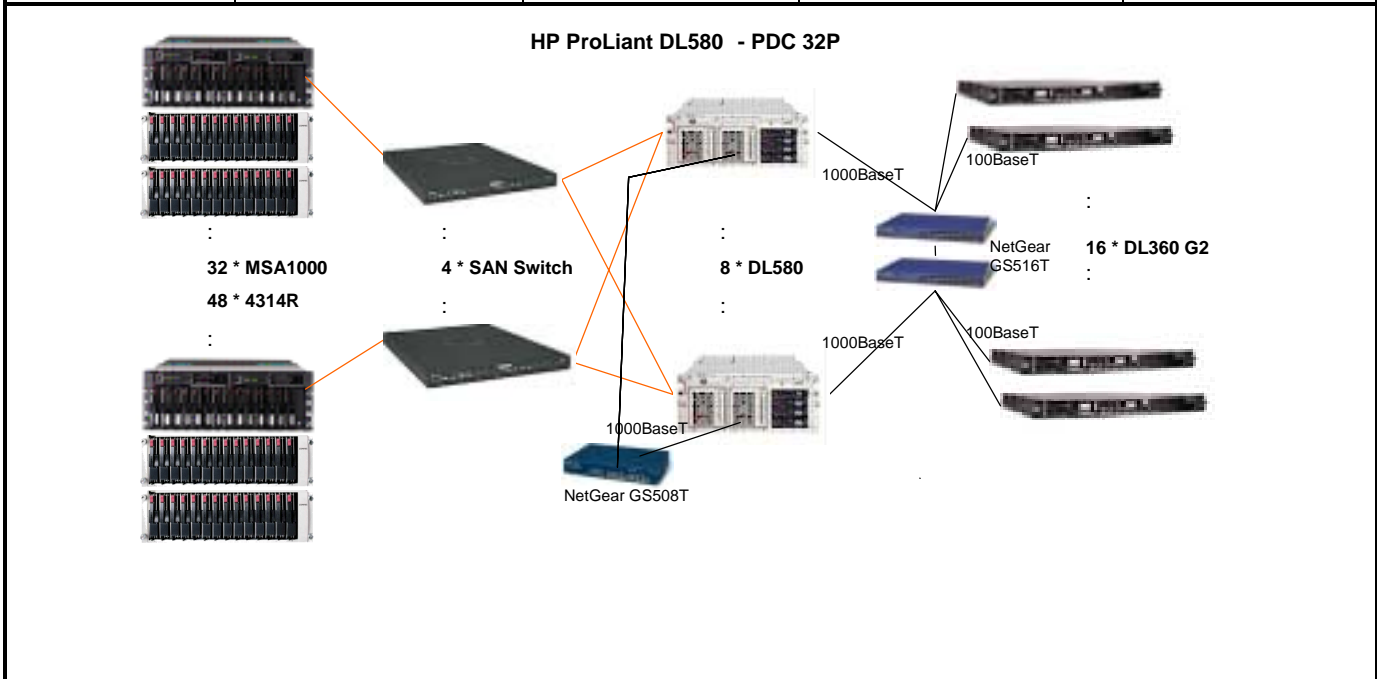


HP ProLiant DL580 - PDC 32P C/S


TPC-C Rev. 5.0

Report Date: October 2, 2002

Total System Cost	TPC-C Throughput	Price Performance	Availability Date
\$2,404,503	138,362.03 tpmC	\$17.38	March 5, 2003* Hardware available now
Processors	Database Manager	Operating System	Other Software
32 PIII Xeon 900/ MHz/2MB-Servers 16 PIII 1400 MHz/ 133/256K - Clients	Oracle 9i Enterprise Edi. Rel. 2 with Real Application Clusters and Partitioning Options	Red Hat Linux Advanced Server	BEA Tuxedo 8.0
			115200



System Components	Server		Clients	
	Quantity	Description	Quantity	Description
Processor	32	Pentium III Xeon 900MHz/2MB	32	Pentium III 1400 MHz/133/ 256K
Memory	32	4 GB	64	1 GB
Disk Controllers	32	FCA 2214 - Fibre Channel HBA		
	32	Modular SAN Array 1000		
	48	StorageWorks Enclosure Model 4314R		
	4	StorageWorks SAN Switch 2/16 EL		
Disk Drives	976	18.2-GB 15K	16	18.2-GB 10K
	16	18.2-GB 10K		
Total Storage		18054 GB		
Tape Drives	1	12/24-Gigabyte DAT		

		HP ProLiant DL580 - PDC 32P with 16 ProLiant DL360R			TPC-C Rev. 5.0		
					Report Date:		2-Oct-02
Description	Part Number	Third Party	Unit Price	Qty	Extended Price	3 yr. Maint. Price	
Server Hardware							
ProLiant DL580R X900 2MB 1024	155618-003	Brand Pricing	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	32	127,968	
NC7770 PCI-X Gigabit Server Adapter	244948-B21		1	227	16	3,632	
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	
StorageWorks SAN Switch 2/16 EL	283056-B21		1	14,775	4	59,100	
5m LC-LC cable kit	221692-B22		1	82	64	5,248	
2GB Small Form Pluggable Adapter Kit	221470-B21		1	369	64	23,616	
FCA 2214 - Fibre Channel HBA	281541-B21		1	1,590	32	50,880	
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	447	976	436,272	
18.2-GB Pluggable 1" Universal WideUltra3 15K HDD (10% spares)	188122-B22		1	447	98		43,806
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
NetGear GS508T 10/100/1000 Copper Gigabit Switch	GS508TNA	NetGear	5	719	3	2,157	
					Subtotal	1,421,586	146,254
Server Software							
Oracle9i DB Ent. Edi. Rel. 2, per processor for 3 years	Run time	Oracle	3	20,000	32	640,000	
Real Application Clusters, per processor for 3 Years	Run time	Oracle	3	10,000	32	320,000	
Partitioning, per processor for 3Years	Run time	Oracle	3	5,000	32	160,000	
Oracle Database Server Support Package for 3 years	Run time	Oracle	3	48,000	1		48,000
Red Hat Linux Advanced Server 2.1 (Basic)	RHF0090US	Red Hat	2	800	8	6,400	
Red Hat Two additional year maintenance and support	RHF0090US	Red Hat	2	800	16		12,800
					Subtotal	1,126,400	60,800
Client Hardware							
ProLiant DL360R02 P1.40/133-256K 128MB	233271-001		1	2,229	16	35,664	
1G Reg 133MHz SDRAM DIMM	128280-B21		1	880	64	56,320	
1.40GHz PIII Processor Option Kit (DL360 G2)	233273-B21		1	804	16	12,864	
NC3134 64 PCI Dual 10/100 All option kit	138603-B21		1	285	16	4,560	
V570 Color Monitor - 15 inch CRT - Opal	228113-001		1	169	1	169	
HP Enhanced Keyboard	122660-006		1	44	1	44	
HP Mouse	170299-B21		1	23	1	23	
18.2GB Pluggable Ultra3 SCSI 10K 1" Universal HDD	142673-B22		1	319	16	5,104	
FM-L0724-363YR 24x7 4HR 300 SERIES SVR	162657-002		1	1,450	16		23,200
					Subtotal	114,748	23,200
Client Software							
BEA Tuxedo 8.0 Teir 1		BEA	4	3,000	16	48,000	30,240
Red Hat Linux Personal (unlimited copies)	RHF099US	Red Hat	2	60	1	60	
Red Hat Linux Personal 8 systems x 3 years m& s Bundle	MCT0172US	Red Hat	2	4,800	2		9,600
					Subtotal	48,060	39,840
User Connectivity							
NetGear GS516T 10/100/1000 Copper Gigabit Switch	GS516TNA	NetGear	5	1,400	4	5,598	
					Subtotal	5,598	0
					Total	\$2,163,216	\$241,287
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: \$2,404,503		
					tpmC Rating: 138,362.03		
					\$ / tpmC: \$17.38		
Pricing: 1=HP Direct 2= Red Hat 3=Oracle (contact MaryBeth Pierantoni @ 650-506-2118, See Appendix G of FDR) 4=BEA 5=CDW							
Note 1 = Discount based on HP Direct guidance 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 138,362.03 tpmC

Response Times (in seconds)	Average	90%	Maximum
New-Order	0.893	1.992	85.290
Payment	0.574	1.146	84.770
Order-Status	0.643	1.339	45.256
Delivery (interactive portion)	0.260	0.349	81.913
Delivery (deferred portion)	0.419	0.793	82.654
Stock-Level	0.581	0.831	81.376
Menu	0.102	0.102	0.328

Transaction Mix, in percent of total transaction

New-Order	44.916%
Payment	43.019%
Order-Status	4.020%
Delivery	4.025%
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	18.008/13.406	18.022/134.006
Payment	3.010/0.000	3.018/12.016	3.032/120.045
Order-Status	2.010/0.000	2.018/10.015	2.031/100.077
Delivery (interactive)	2.010/0.000	2.018/5.025	2.031/50.179
Stock-Level	2.010/0.000	2.018/5.014	2.030/50.096

Test Duration

Ramp-up time	102 minutes
Measurement interval	120 minutes
Transactions (all types) completed during measurement interval	38,453,521
Ramp down time	56 minutes

Checkpointing

Number of checkpoints	4
Checkpoint interval	30 minutes

NOTE: The result was found to have an "Insignificant Deviation" due to the original FDR missing some documentation. These files are now included in the appendix of the FDR.

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.

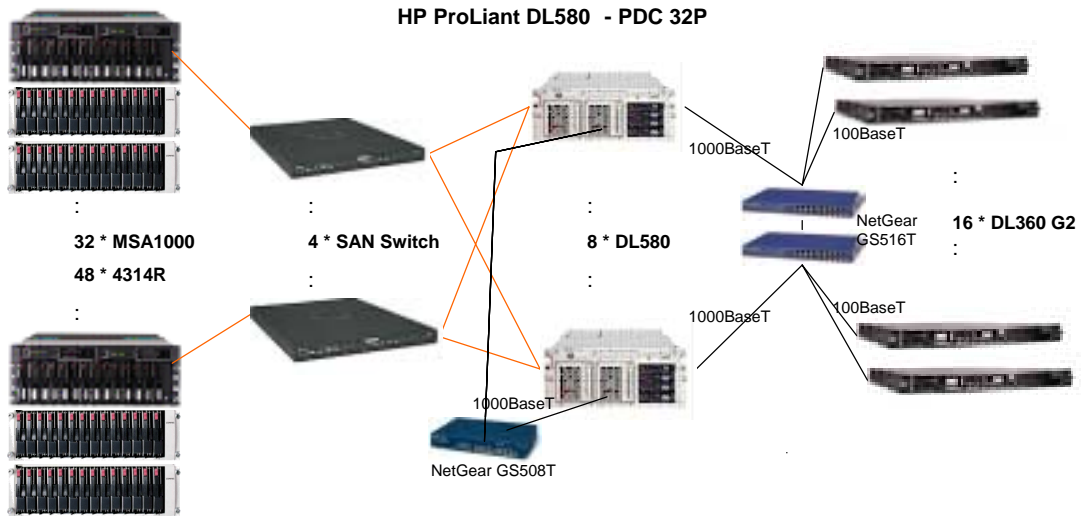
Appendix C contains the tunable parameters for the database, the operating system, and the transaction monitor.

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.

976 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	655GB	1/24 stock, cust, ordl, nord, hist & icust2, 1/8 ware & rollback segs, control file
3	1	655GB	1/24 stock, cust, ordl, nord, hist & icust2, 1/8 dist, 1/15 temp
4	1	655GB	1/24 stock, cust, ordl, nord, hist & icust2, 1/8 system, 1/15 temp
5	1	254GB	Database log for Node 2
6	1	655GB	1/24 stock, cust, ordl, nord, hist & icust2, 1/8 ware & rollback segs, cluster manager
7	1	655GB	1/24 stock, cust, ordl, nord, ihist & cust2, 1/8 dist, 1/15 temp
8	1	655GB	1/24 stock, cust, ordl, nord, hist & icust2, 1/8 system, 1/15 temp
9	2	254GB	Database log for Node 3
10	2	655GB	1/24 stock, cust, ordl, nord, hist & icust2, 1/8 ware & rollback segs, 1/2 icust-1 and istock
11	2	655GB	1/24 stock, cust, ordl, nord, ihist & cust2, 1/8 dist, 1/15 temp
12	2	655GB	1/24 stock, cust, ordl, nord, hist & icust2, 1/8 system, 1/15 temp
13	2	254GB	Database log for Node 4
14	2	655GB	1/24 stock, cust, ordl, nord, hist & icust2, 1/8 ware & rollback segs 1/2 icust-1 and istock
15	2	655GB	1/24 stock, cust, ordl, nord, ihist & cust2, 1/8 dist, 1/15 temp
16	2	655GB	1/24 stock, cust, ordl, nord, hist & icust2, 1/8 system, 1/15 temp
17	3	254GB	Database log for Node 5
18	3	655GB	1/24 stock, cust, ordl, nord, hist & icust2, 1/8 ware & rollback segs, statspack
19	3	655GB	1/24 stock, cust, ordl, nord, ihist & cust2, 1/8 dist, 1/15 temp
20	3	655GB	1/24 stock, cust, ordl, nord, hist & icust2, 1/8 system, 1/15 temp
21	3	254GB	Database log for Node 6
22	3	655GB	1/24 stock, cust, ordl, nord, hist & icust2, 1/8 ware & rollback segs, item table & index
23	3	655GB	1/24 stock, cust, ordl, nord, ihist & cust2, 1/8 dist, 1/15 temp
24	3	655GB	1/24 stock, cust, ordl, nord, hist & icust2, 1/8 system, 1/15 temp
25	4	254GB	Database log for Node 7
26	4	655GB	1/24 stock, cust, ordl, nord, hist & icust2, 1/8 ware & rollback segs, extra cust file for 5% space
27	4	655GB	1/24 stock, cust, ordl, nord, ihist & cust2, 1/8 dist, 1/15 temp
28	4	655GB	1/24 stock, cust, ordl, nord, hist & icust2, 1/8 system, 1/15 temp
29	4	254GB	Database log for Node 8
30	4	655GB	1/24 stock, cust, ordl, nord, hist & icust2, 1/8 ware & rollback segs, 1/2 extra stock 5% space
31	4	655GB	1/24 stock, cust, ordl, nord, ihist & cust2, 1/8 dist, 1/15 temp
32	4	655GB	1/24 stock, cust, ordl, nord, hist & icust2, 1/8 system, 1/2 extra stock 5% growth space

Priced Configuration:

The priced configuration has additional 12/24-Gigabyte DAT drive. 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 neworder (nord), orders (ordr), orderline (ordl) and history (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	84.99%
	Remote warehouse	15.01%
	Accessed by last name	59.99%
Order Status	Accessed by last name	60.02%
Delivery	Skipped transactions	None
Transaction Mix	New Order	44.916%
	Payment	43.019%
	Order status	4.020%
	Delivery	4.025%
	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. Isolation was tested in both a single node environment and in the multiple node environment. 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 2800 warehouses. The standard driving mechanism was used to generate the transaction load of 28000 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 28000 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.
- 15.

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 28000 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 11520 warehouses under a full load of 115200 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 115200 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 11520 warehouses under a full load of 115200 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 115200 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 11520 warehouses under a full load of 115200 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 115200 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	11520
District	115200
Customer	345600000
History	345600000
Order	345600000
New Order	10368000
Order Line	3456103850
Stock	1152000000
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 12 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 12 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 8 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.

Table 1

SEGMENT	BLOCKS	REQUIRED	STATIC	DYNAMIC	OVERSIZE
CUSTCLUSTER	91,110,400	91,098,005	91,098,005	-	12,395
DISTCLUSTER	138,240	120,965	120,965	-	17,275
HIST	9,216,000	7,319,806	-	6,139,904	1,896,194
ICUST1	8,345,600	2,123,520	2,123,520	-	6,222,080
ICUST2	6,137,856	4,729,805	4,729,805	-	1,408,051
IDIST	8,345,600	107,520	107,520	-	8,238,080
IITEM	15,360	499	499	-	14,861
INORD	2,457,600	1,268,736	1,268,736	-	1,188,864
IORDL	98,365,440	78,169,084	-	65,568,768	20,196,356
IORDR1	3,686,400	2,571,878	2,571,878	-	1,114,522
IORDR2	5,529,600	3,862,118	3,862,118	-	1,667,482
ISTOK	8,345,600	6,101,760	6,101,760	-	2,243,840
ITEMCLUSTER	15,360	3,413	3,413	-	11,947
IWARE	8,345,600	107,520	107,520	-	8,238,080
ORDR	6,144,000	4,878,243	-	4,091,904	1,265,757
STOKCLUSTER	121,651,200	121,464,005	121,464,005	-	187,195
SYSTEM	196,608	196,608	196,608	-	-
WARECLUSTER	13,824	13,613	13,613	-	211
STATIC	Dynamic	Oversize	Daily Growth	Daily Spread	Space 60
233,769,965	75,800,576	53,923,190	14,566,557	32,073,354	3,032,164,636
Block size	4,096		Log/NO_txn (KB)	12.5116	
60 Day (blocks)	3,032,164,636		tpmC	138,362.03	
60 Day (GB)	11,567		8hr Log (GB)	792.45	
Data Disks	1,008		raid-1	1,584.90	
Capacity (GB)	16.96				
Total (GB)	17,095.68		Log disks	112	
			Capacity (GB)	16.96	
			Total (GB)	1,899.52	

Clause 5 Related Items

Throughput

Measured tpmC must be reported

Measured tpmC 138362.03 tpmC

Price per tpmC \$17.38 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	0.893	85.290	1.992
Payment	0.574	84.770	1.146
Order-Status	0.643	45.256	1.339
Interactive Delivery	0.260	81.913	0.349
Deferred Delivery	0.419	82.654	0.793
Stock-Level	0.581	81.376	0.831
Menu	0.102	0.328	0.102

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	18.008	18.022
Payment	3.010	3.018	3.032
Order-Status	2.010	2.018	2.031
Interactive Delivery	2.010	2.018	2.031
Stock-Level	2.010	2.018	2.030

Table 5.3: Think Times

Type	Minimum	Average	Maximum
New-Order	0.000	13.406	134.006
Payment	0.000	12.016	120.045
Order-Status	0.000	10.015	100.077
Interactive Delivery	0.000	5.025	50.179
Stock-Level	0.000	5.014	50.096

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

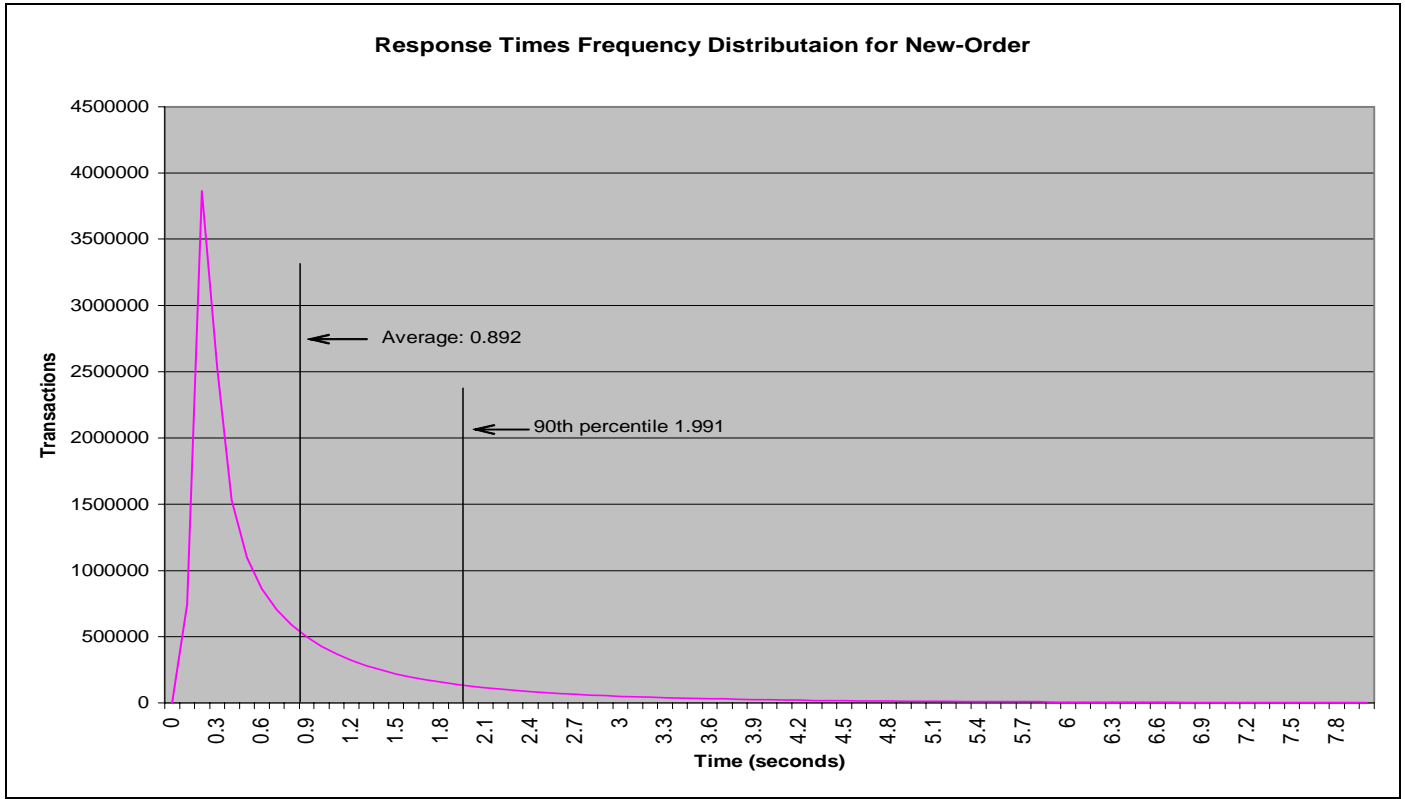


Figure 5.2: Response Times Frequency Distribution for Payment Transactions

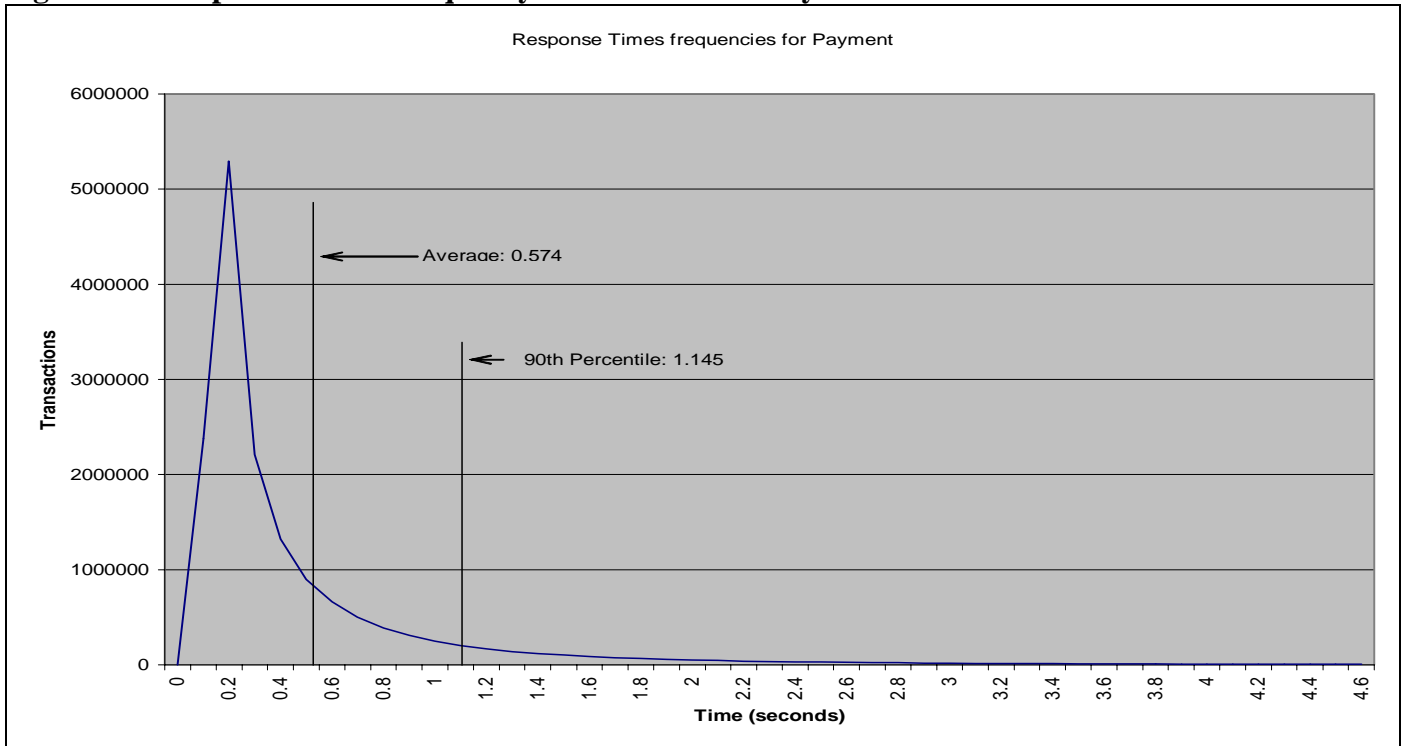


Figure 5.3: Response Times Frequency Distribution for Order Status Transactions

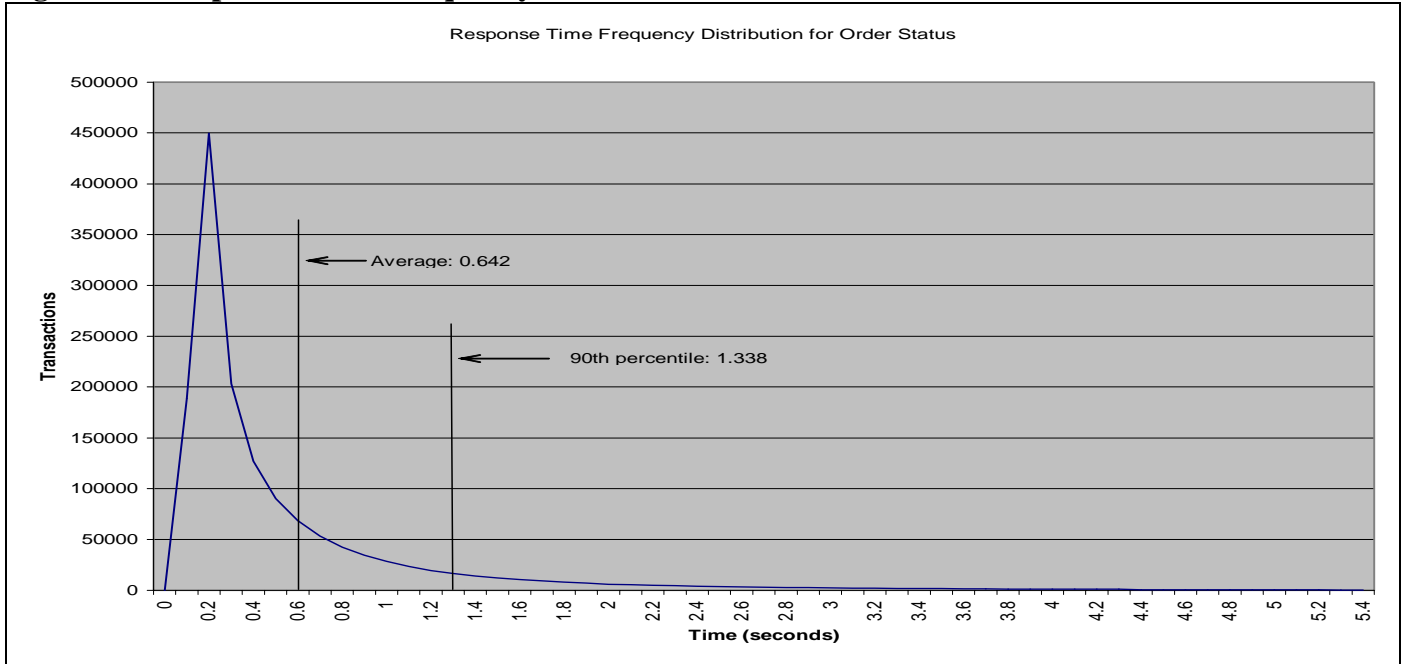


Figure 5.4: Response Times Frequency Distribution for Delivery Transactions

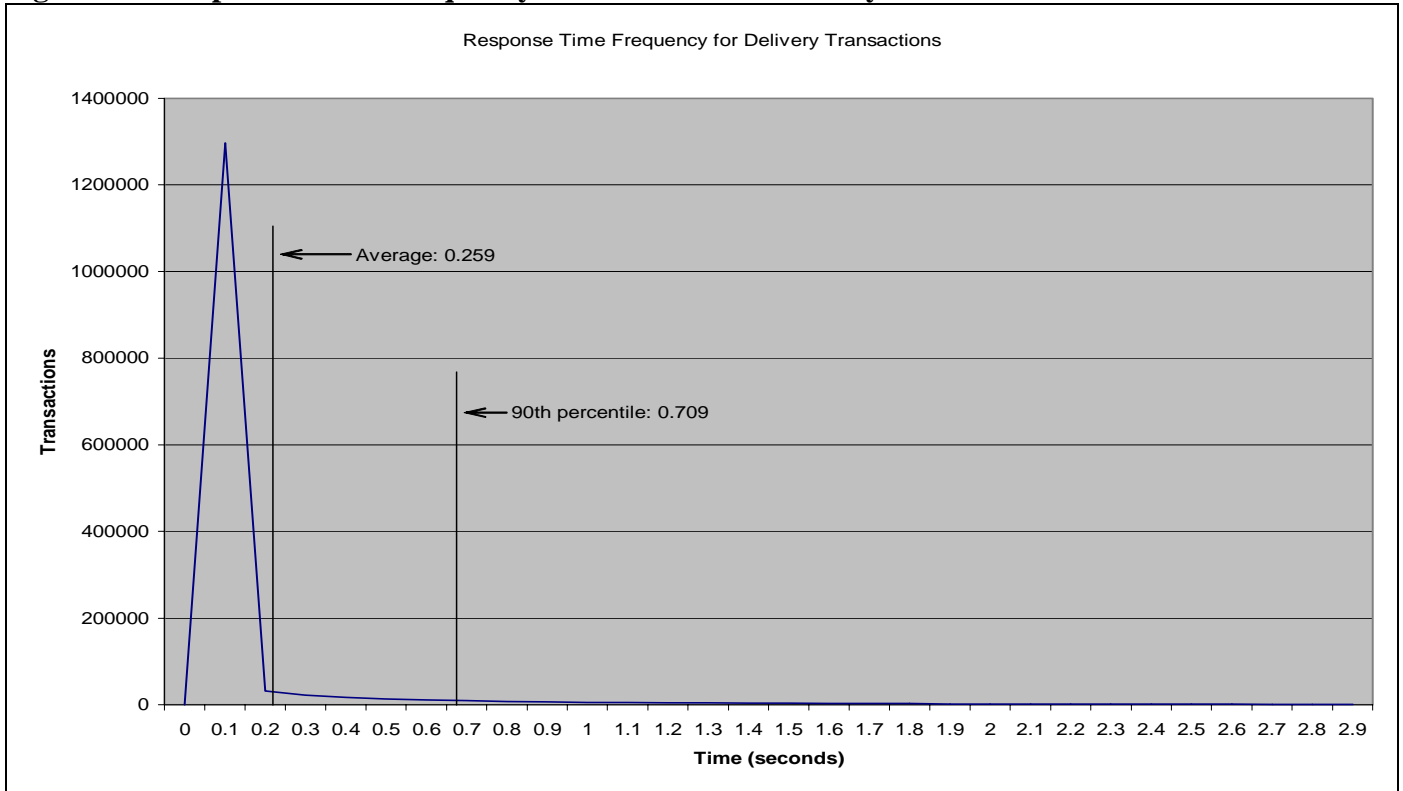


Figure 5.5: Response Times Frequency Distribution for Stock Level Transactions

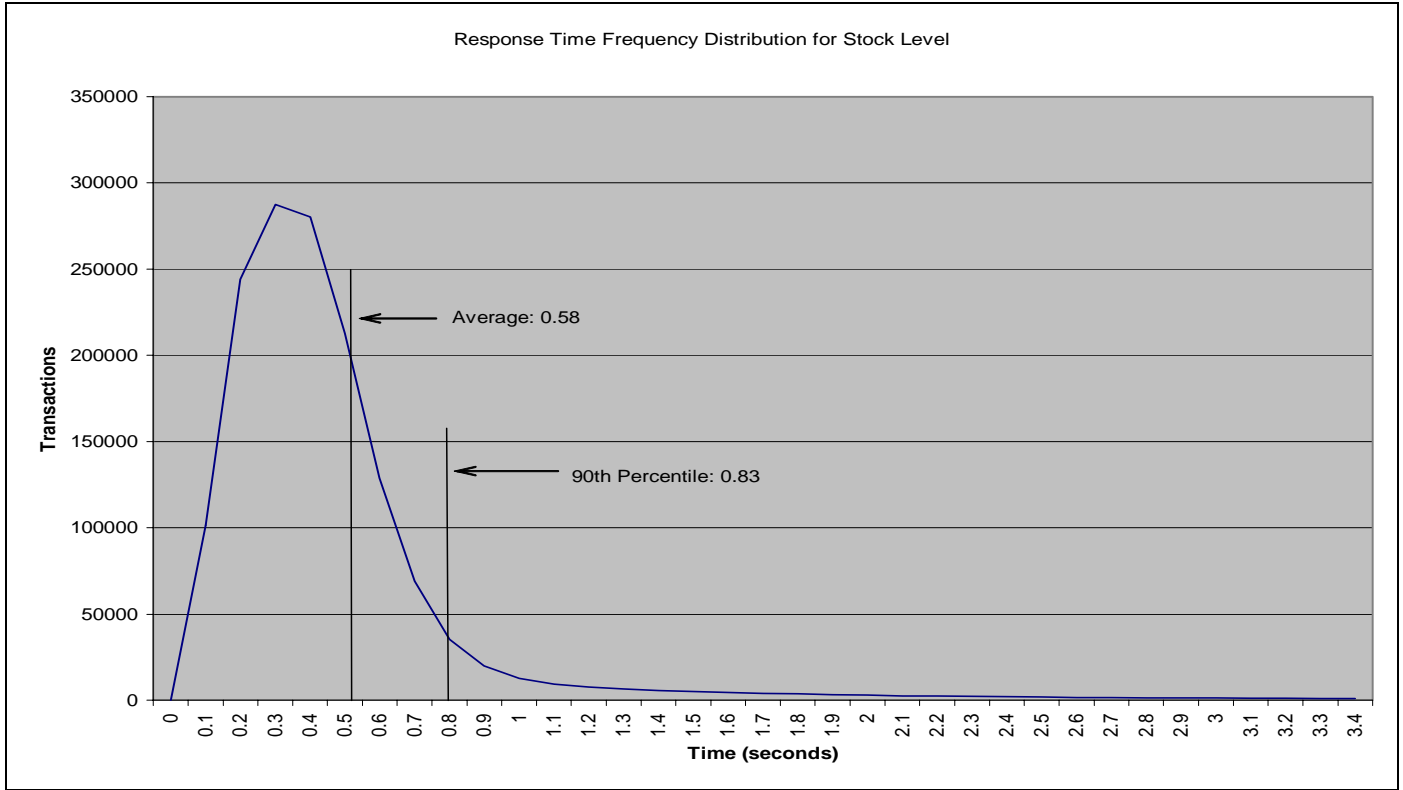


Figure 5.6: Response Time versus Throughput

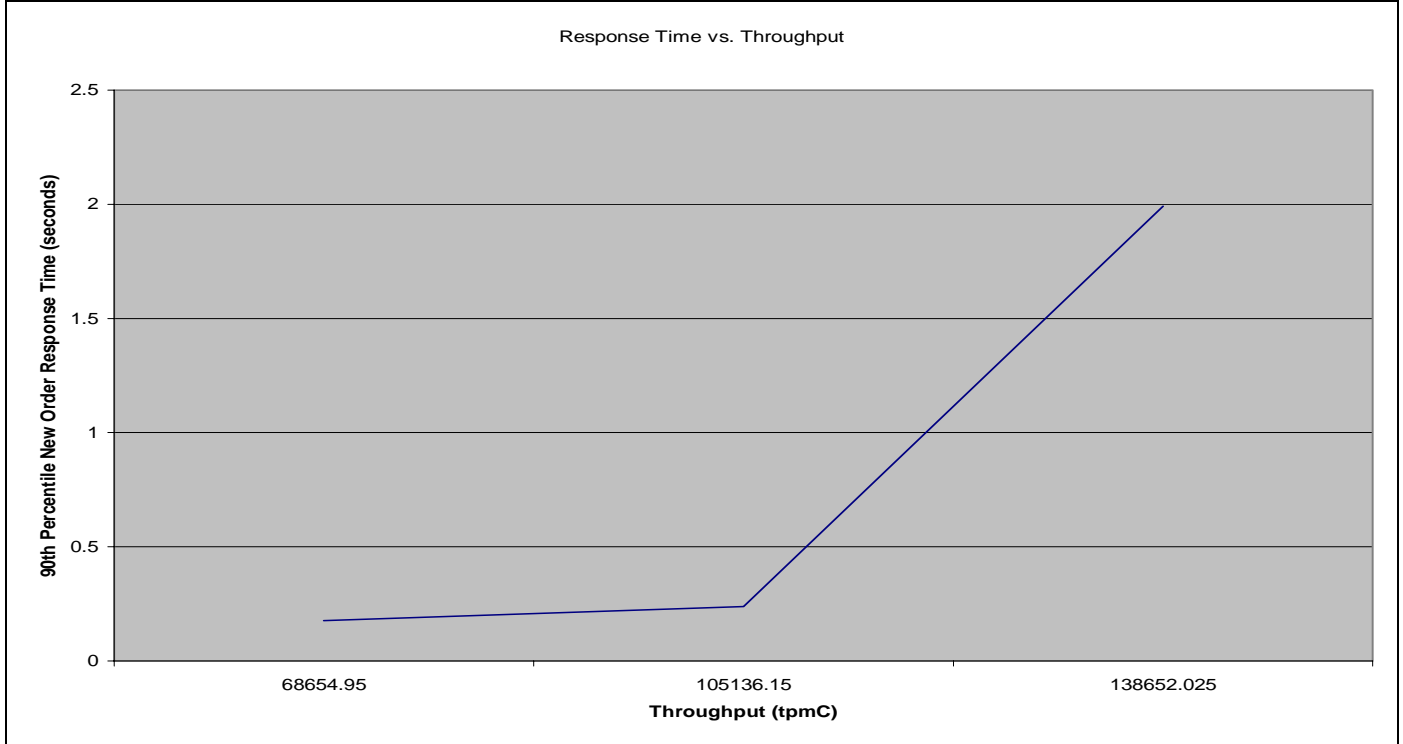


Figure 5.7: Think Times distribution for New Order Transactions

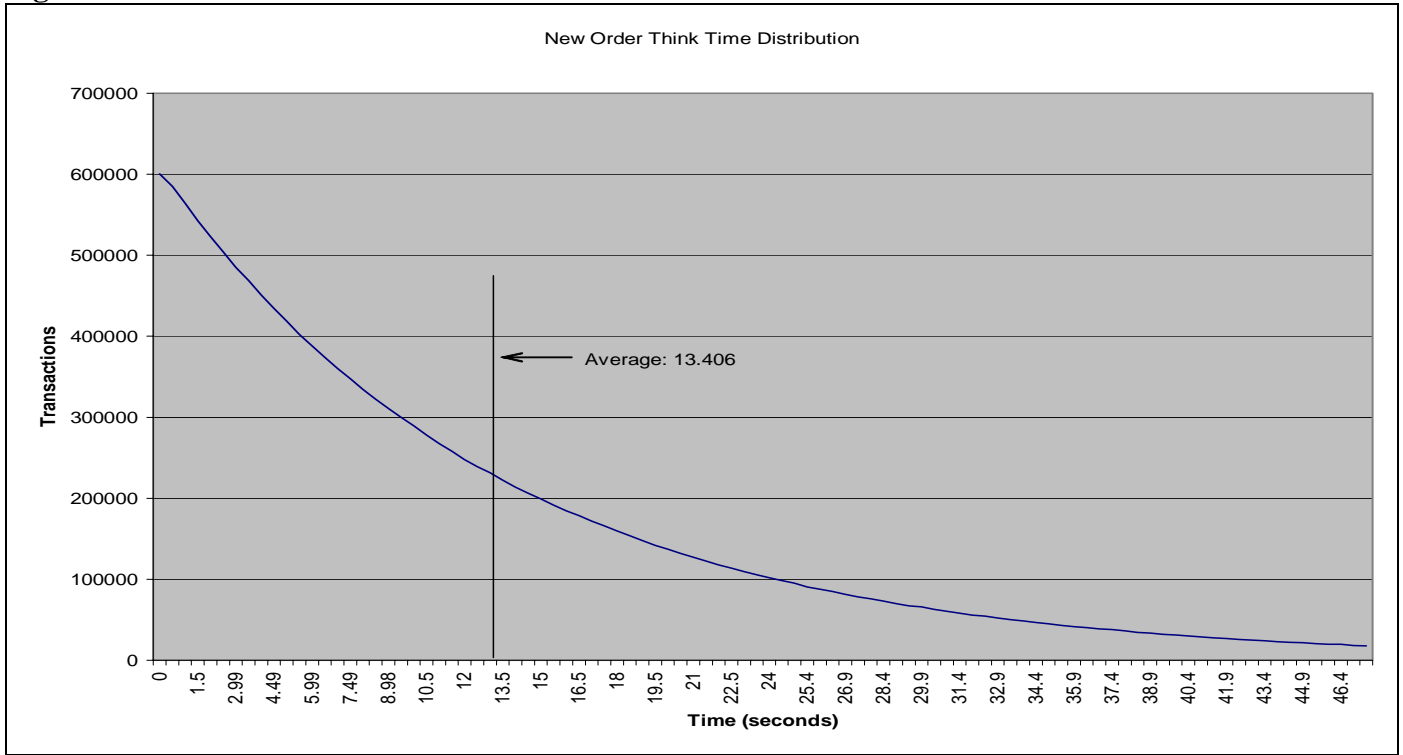
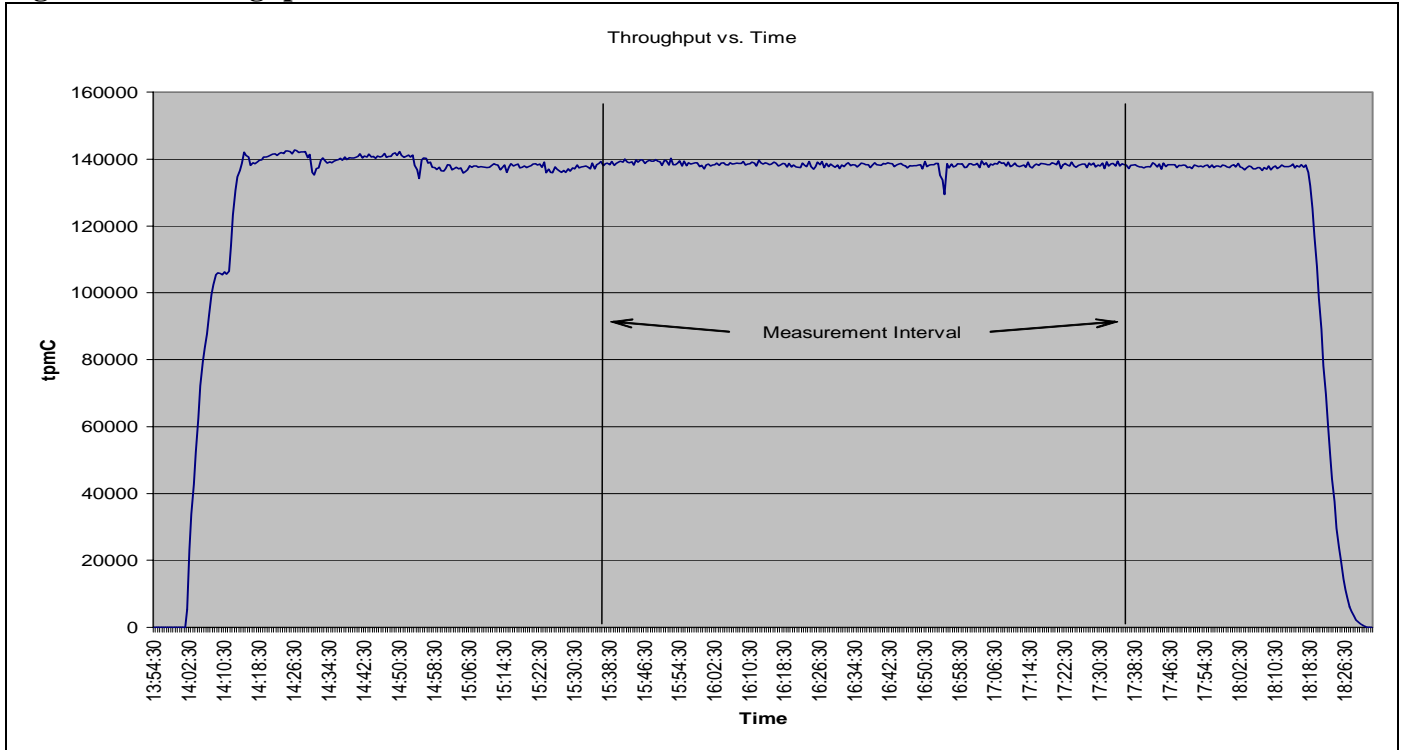


Figure 5.8: Throughput versus Time



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 Apache Web 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	84.99%
	Remote warehouse	15.01%
	Accessed by last name	59.99%
Order Status	Accessed by last name	60.02%
Delivery	Skipped transactions	0
Transaction Mix	New Order	44.92%
	Payment	43.02%
	Order status	4.02%
	Delivery	4.03%
	Stock level	4.02%

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 - PDC 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	16:01:31	16:28:10	0:26:39
2	16:02:08	16:28:44	0:26:36
3	16:02:52	16:29:49	0:26:57
4	16:02:01	16:28:37	0:26:36
5	16:01:50	16:28:21	0:26:31
6	16:02:03	16:28:37	0:26:34
7	16:02:23	16:28:57	0:26:34
8	16:02:04	16:28:36	0:26:32
1	16:31:06	16:57:26	0:26:20
2	16:31:38	16:57:52	0:26:14
3	16:31:47	16:57:32	0:25:45
4	16:31:33	16:57:50	0:26:17
5	16:31:15	16:57:28	0:26:13
6	16:31:34	16:57:55	0:26:21
7	16:31:53	16:58:11	0:26:18
8	16:31:31	16:57:46	0:26:15
1	17:00:19	17:26:41	0:26:22
2	17:00:46	17:27:13	0:26:27
3	17:01:29	17:27:32	0:26:03
4	17:00:47	17:27:17	0:26:30
5	17:00:23	17:26:51	0:26:28
6	17:00:49	17:27:24	0:26:35
7	17:01:07	17:27:40	0:26:33
8	17:00:42	17:27:14	0:26:32
1	17:29:38	17:56:05	0:26:27
2	17:30:09	17:56:33	0:26:24
3	17:30:30	17:57:24	0:26:54
4	17:30:13	17:56:43	0:26:30
5	17:29:47	17:56:10	0:26:23
6	17:30:20	17:56:44	0:26:24
7	17:30:34	17:56:59	0:26:25
8	17:30:11	17:56:34	0:26:23

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 16 client systems were connected into two cascaded 16 port 1GB Ethernet switches. In the tested configuration there were eight driver systems (RTE), each of them connected to two clients systems using 1000/100 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** 138,362.03 tpmC
- **Price per tpmC** \$17.38 per tpmC
- **Available** March 5, 2003

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 Red Hat Linux Advanced Server
- 16 Red Hat Linux Personal
- 16 BEA Tuxedo CTS 8.0

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

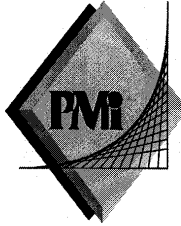
The Full Disclosure Report must be readily available to the public at a reasonable charge, similar to the charges for similar documents by the test sponsor. The report must be made available when results are made public. In order to use the phrase "TPC Benchmark™ C", the Full Disclosure Report must have 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

September 5, 2002

Mr. Raghunath Othayoth and
Mr. Bryon Georgson
Database Performance Engineers
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 ProLiant DL580 - O2000 32P
Database Manager: Oracle9i
Operating System: Red Hat Linux Advanced Server version 2.1
Transaction Monitor: BEA Tuxedo 8.0

Servers: 8 ProLiant DL580's with:				
CPU's	Memory	Disks (total)	90% Response	TpmC
4 Pentium III Xeon@900Mhz	Main: 16 GB Cache: 2 MB	976 @18GB(15K) 16 @ 18GB (10K)	1.99 sec	138,362.03
16 Clients: DL360R each with:				
2 Pentium III Xeon @ 1.4 Ghz	Main: 4 GB Cache: 512 K	1 @ 18 GB	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.

137 Yankton St, Suite 101, Folsom, CA 95630
(916) 985-113 fax: (916) 985-1185 email: Lorna@PerfMetrics.com

Page 1

PERFORMANCE METRICS INC.
TPC Certified Auditors

- The database was properly scaled with 11,520 warehouses. All warehouses were active during the performance run.
- The ACID properties were successfully demonstrated. Both a single node and a full system failures were 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:
None.

Sincerely,



Lorna Livingtree
Auditor

Appendix A: Source Code

```
*****
BS-7dc9.c
*****

#include <stdio.h>
#include <xa.h>
#include <atmi.h>

#if defined(__cplusplus)
extern "C" {
#endif
extern int tmrunserver _((int));
extern void dy_transaction _((TPSVINFO *));
extern void no_transaction _((TPSVINFO *));
extern void os_transaction _((TPSVINFO *));
extern void pt_transaction _((TPSVINFO *));
extern void sl_transaction _((TPSVINFO *));
#if defined(__cplusplus)
}
#endif

static struct tmdsptchtbl_t_tmdsptchtbl[] = {
  { (char*)"dy_transaction", (char*)"dy_transaction", (void *)
    _((TPSVINFO *)) dy_transaction, 0, 0 },
  { (char*)"no_transaction", (char*)"no_transaction", (void *)
    _((TPSVINFO *)) no_transaction, 1, 0 },
  { (char*)"os_transaction", (char*)"os_transaction", (void *)
    _((TPSVINFO *)) os_transaction, 2, 0 },
  { (char*)"pt_transaction", (char*)"pt_transaction", (void *)
    _((TPSVINFO *)) pt_transaction, 3, 0 },
  { (char*)"sl_transaction", (char*)"sl_transaction", (void *)
    _((TPSVINFO *)) sl_transaction, 4, 0 },
  { NULL, NULL, NULL, 0, 0 }
};

#ifdef _TMDLLIMPORT
#define _TMDLLIMPORT
#endif

#if defined(__cplusplus)
extern "C" {
#endif
#ifdef _TMDLLIMPORT extern struct xa_switch_t tnull_switch;
#endif
#if defined(__cplusplus)
}
#endif

typedef void (*tmp_void_cast)();
typedef void (*tmp_voidvoid_cast)(void);
typedef int (*tmp_intchar_cast)(int, char **);
typedef int (*tmp_int_cast)(int);
static struct tmsvrargs_t tmsvrargs = {
  NULL,
  &tmdsptchtbl[0],
  0,
  (tmp_intchar_cast)tpsvrinit,
  (tmp_voidvoid_cast)tpsvrdone,
  (tmp_int_cast)tmrunserver, /* PRIVATE */
  NULL, /* RESERVED */
  NULL, /* RESERVED */
  NULL, /* RESERVED */
  NULL, /* RESERVED */
  (tmp_intchar_cast)tpsvrthrininit,
  (tmp_voidvoid_cast)tpsvrthrdone
};

struct tmsvrargs_t *
#ifdef _TMPROTOTYPES
tmgetsvrargs(void)
#else
_tmgetsvrargs()
#endif
{
  tmsvrargs.reserved1 = NULL;
  tmsvrargs.reserved2 = NULL;
  tmsvrargs.xa_switch = &tnull_switch;
  return(&tmsvrargs);
}

int
#ifdef _TMPROTOTYPES
main(int argc, char **argv)
#else
main(argc, argv)
int argc;
char **argv;
#endif
{
#ifdef TMAINEXIT

```

```
#include "mainexit.h"
#endif

return( tmstartserver( argc, argv, tmgetsvrargs()));
}

*****
BS-deli.c
*****

#include <stdio.h>
#include <xa.h>
#include <atmi.h>

#if defined(__cplusplus)
extern "C" {
#endif
extern int tmrunserver _((int));
extern void dy_transaction _((TPSVINFO *));
#if defined(__cplusplus)
}
#endif

static struct tmdsptchtbl_t_tmdsptchtbl[] = {
  { (char*)"dy_transaction", (char*)"dy_transaction", (void *)
    _((TPSVINFO *)) dy_transaction, 0, 0 },
  { NULL, NULL, NULL, 0, 0 }
};

#ifdef _TMDLLIMPORT
#define _TMDLLIMPORT
#endif

#if defined(__cplusplus)
extern "C" {
#endif
#ifdef _TMDLLIMPORT extern struct xa_switch_t tnull_switch;
#endif
#if defined(__cplusplus)
}
#endif

typedef void (*tmp_void_cast)();
typedef void (*tmp_voidvoid_cast)(void);
typedef int (*tmp_intchar_cast)(int, char **);
typedef int (*tmp_int_cast)(int);
static struct tmsvrargs_t tmsvrargs = {
  NULL,
  &tmdsptchtbl[0],
  0,
  (tmp_intchar_cast)tpsvrinit,
  (tmp_voidvoid_cast)tpsvrdone,
  (tmp_int_cast)tmrunserver, /* PRIVATE */
  NULL, /* RESERVED */
  NULL, /* RESERVED */
  NULL, /* RESERVED */
  NULL, /* RESERVED */
  (tmp_intchar_cast)tpsvrthrininit,
  (tmp_voidvoid_cast)tpsvrthrdone
};

struct tmsvrargs_t *
#ifdef _TMPROTOTYPES
tmgetsvrargs(void)
#else
_tmgetsvrargs()
#endif
{
  tmsvrargs.reserved1 = NULL;
  tmsvrargs.reserved2 = NULL;
  tmsvrargs.xa_switch = &tnull_switch;
  return(&tmsvrargs);
}

int
#ifdef _TMPROTOTYPES
main(int argc, char **argv)
#else
main(argc, argv)
int argc;
char **argv;
#endif
{
#ifdef TMAINEXIT
#include "mainexit.h"
#endif

return( tmstartserver( argc, argv, tmgetsvrargs()));
}

*****
BS-newo.c
*****

#include <stdio.h>
#include <xa.h>
#include <atmi.h>

```

```

#if defined(__cplusplus)
extern "C" {
#endif
extern int _tmrunserver _((int));
extern void no_transaction _((TPSVCINFO *));
#if defined(__cplusplus)
}
#endif

static struct tmdsptchtbl_t _tmdsptchtbl[] = {
  { (char*)"no_transaction", (char*)"no_transaction", (void (*)
_((TPSVCINFO *))) no_transaction, 0, 0 },
  { NULL, NULL, NULL, 0, 0 }
};

#ifdef _TMDLLIMPORT
#define _TMDLLIMPORT
#endif

#if defined(__cplusplus)
extern "C" {
#endif
#ifdef _TMDLLIMPORT extern struct xa_switch_t tnull_switch;
#endif
#if defined(__cplusplus)
}
#endif

typedef void (*tmp_void_cast)();
typedef void (*tmp_voidvoid_cast)(void);
typedef int (*tmp_intchar_cast)(int, char **);
typedef int (*tmp_int_cast)(int);
static struct tmsvrargs_t tmsvrargs = {
  NULL,
  &_tmdsptchtbl[0],
  0,
  (tmp_intchar_cast)tpsvrinit,
  (tmp_voidvoid_cast)tpsvrdone,
  (tmp_int_cast)_tmrunserver, /* PRIVATE */
  NULL, /* RESERVED */
  NULL, /* RESERVED */
  NULL, /* RESERVED */
  NULL, /* RESERVED */
  (tmp_intchar_cast)tpsvrthrin,
  (tmp_voidvoid_cast)tpsvrthrdone
};

struct tmsvrargs_t *
#ifdef _TMPROTOTYPES
_tmgetsvrargs(void)
#else
_tmgetsvrargs()
#endif
{
  tmsvrargs.reserved1 = NULL;
  tmsvrargs.reserved2 = NULL;
  tmsvrargs.xa_switch = &tnull_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()));
}

*****
BS-ordo.c
*****

#include <stdio.h>
#include <xa.h>
#include <atmi.h>

#if defined(__cplusplus)
extern "C" {
#endif
extern int _tmrunserver _((int));
extern void os_transaction _((TPSVCINFO *));
#if defined(__cplusplus)
}
#endif

static struct tmdsptchtbl_t _tmdsptchtbl[] = {
  { (char*)"os_transaction", (char*)"os_transaction", (void (*)
_((TPSVCINFO *))) os_transaction, 0, 0 },
  { NULL, NULL, NULL, 0, 0 }
};

#ifdef _TMDLLIMPORT

```

```

#define _TMDLLIMPORT
#endif

#if defined(__cplusplus)
extern "C" {
#endif
#ifdef _TMDLLIMPORT extern struct xa_switch_t tnull_switch;
#endif
#if defined(__cplusplus)
}
#endif

typedef void (*tmp_void_cast)();
typedef void (*tmp_voidvoid_cast)(void);
typedef int (*tmp_intchar_cast)(int, char **);
typedef int (*tmp_int_cast)(int);
static struct tmsvrargs_t tmsvrargs = {
  NULL,
  &_tmdsptchtbl[0],
  0,
  (tmp_intchar_cast)tpsvrinit,
  (tmp_voidvoid_cast)tpsvrdone,
  (tmp_int_cast)_tmrunserver, /* PRIVATE */
  NULL, /* RESERVED */
  NULL, /* RESERVED */
  NULL, /* RESERVED */
  NULL, /* RESERVED */
  (tmp_intchar_cast)tpsvrthrin,
  (tmp_voidvoid_cast)tpsvrthrdone
};

struct tmsvrargs_t *
#ifdef _TMPROTOTYPES
_tmgetsvrargs(void)
#else
_tmgetsvrargs()
#endif
{
  tmsvrargs.reserved1 = NULL;

  tmsvrargs.reserved2 = NULL;
  tmsvrargs.xa_switch = &tnull_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()));
}

*****
BS-payo.c
*****

#include <stdio.h>
#include <xa.h>
#include <atmi.h>

#if defined(__cplusplus)
extern "C" {
#endif
extern int _tmrunserver _((int));
extern void pt_transaction _((TPSVCINFO *));
#if defined(__cplusplus)
}
#endif

static struct tmdsptchtbl_t _tmdsptchtbl[] = {
  { (char*)"pt_transaction", (char*)"pt_transaction", (void (*)
_((TPSVCINFO *))) pt_transaction, 0, 0 },
  { NULL, NULL, NULL, 0, 0 }
};

#ifdef _TMDLLIMPORT
#define _TMDLLIMPORT
#endif

#if defined(__cplusplus)
extern "C" {
#endif
#ifdef _TMDLLIMPORT extern struct xa_switch_t tnull_switch;
#endif
#if defined(__cplusplus)
}
#endif

typedef void (*tmp_void_cast)();
typedef void (*tmp_voidvoid_cast)(void);
typedef int (*tmp_intchar_cast)(int, char **);
typedef int (*tmp_int_cast)(int);

```

```

static struct tmsvrargs_t tmsvrargs = {
    NULL,
    &tmdsptchtbl[0],
    0,
    (tmp_intchar_cast)tpsvrinit,
    (tmp_voidvoid_cast)tpsvrdone,
    (tmp_int_cast)_tmrunserver, /* PRIVATE */
    NULL, /* RESERVED */
    NULL, /* RESERVED */
    NULL, /* RESERVED */
    NULL, /* RESERVED */
    (tmp_intchar_cast)tpsvrthrininit,
    (tmp_voidvoid_cast)tpsvrthrdone
};

struct tmsvrargs_t *
#ifdef _TMPROTOTYPES
_tmgetsvrargs(void)
#else
_tmgetsvrargs()
#endif
{
    tmsvrargs.reserved1 = NULL;
    tmsvrargs.reserved2 = NULL;
    tmsvrargs.xa_switch = &tmmnull_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()));
}

*****
BS-stoo.c
*****

#include <stdio.h>
#include <xa.h>
#include <atmi.h>

#ifdef __cplusplus
extern "C" {
#endif
extern int _tmrunserver _((int));
extern void sl_transaction _((TPSVCINFO *));
#ifdef __cplusplus
}
#endif

static struct tmdsptchtbl_t _tmdsptchtbl[] = {
    { (char*)"sl_transaction", (char*)"sl_transaction", (void (*)
_((TPSVCINFO *))) sl_transaction, 0, 0 },
    { NULL, NULL, NULL, 0, 0 }
};

#ifdef _TMDLLIMPORT
#define _TMDLLIMPORT
#endif

#ifdef __cplusplus
extern "C" {
#endif
_TMDLLIMPORT extern struct xa_switch_t tmmnull_switch;
#ifdef __cplusplus
}
#endif

typedef void (*tmp_void_cast)();
typedef void (*tmp_voidvoid_cast)(void);
typedef int (*tmp_intchar_cast)(int, char **);
typedef int (*tmp_int_cast)(int);
static struct tmsvrargs_t tmsvrargs = {
    NULL,
    &tmdsptchtbl[0],
    0,
    (tmp_intchar_cast)tpsvrinit,
    (tmp_voidvoid_cast)tpsvrdone,
    (tmp_int_cast)_tmrunserver, /* PRIVATE */
    NULL, /* RESERVED */
    NULL, /* RESERVED */
    NULL, /* RESERVED */
    NULL, /* RESERVED */
    (tmp_intchar_cast)tpsvrthrininit,
    (tmp_voidvoid_cast)tpsvrthrdone
};

struct tmsvrargs_t *

```

```

#ifdef _TMPROTOTYPES
_tmgetsvrargs(void)
#else
_tmgetsvrargs()
#endif
{
    tmsvrargs.reserved1 = NULL;
    tmsvrargs.reserved2 = NULL;
    tmsvrargs.xa_switch = &tmmnull_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()));
}

*****
BS-tpcc.c
*****

#include <stdio.h>
#include <xa.h>
#include <atmi.h>

#ifdef __cplusplus
extern "C" {
#endif
extern int _tmrunserver _((int));
extern void no_transaction _((TPSVCINFO *));
extern void os_transaction _((TPSVCINFO *));
extern void pt_transaction _((TPSVCINFO *));
extern void sl_transaction _((TPSVCINFO *));
#ifdef __cplusplus
}
#endif

static struct tmdsptchtbl_t _tmdsptchtbl[] = {
    { (char*)"no_transaction", (char*)"no_transaction", (void (*)
_((TPSVCINFO *))) no_transaction, 0, 0 },
    { (char*)"os_transaction", (char*)"os_transaction", (void (*)
_((TPSVCINFO *))) os_transaction, 1, 0 },
    { (char*)"pt_transaction", (char*)"pt_transaction", (void (*)
_((TPSVCINFO *))) pt_transaction, 2, 0 },
    { (char*)"sl_transaction", (char*)"sl_transaction", (void (*)
_((TPSVCINFO *))) sl_transaction, 3, 0 },
    { NULL, NULL, NULL, 0, 0 }
};

#ifdef _TMDLLIMPORT
#define _TMDLLIMPORT
#endif

#ifdef __cplusplus
extern "C" {
#endif
_TMDLLIMPORT extern struct xa_switch_t tmmnull_switch;
#ifdef __cplusplus
}
#endif

typedef void (*tmp_void_cast)();
typedef void (*tmp_voidvoid_cast)(void);
typedef int (*tmp_intchar_cast)(int, char **);
typedef int (*tmp_int_cast)(int);
static struct tmsvrargs_t tmsvrargs = {
    NULL,
    &tmdsptchtbl[0],
    0,
    (tmp_intchar_cast)tpsvrinit,
    (tmp_voidvoid_cast)tpsvrdone,
    (tmp_int_cast)_tmrunserver, /* PRIVATE */
    NULL, /* RESERVED */
    NULL, /* RESERVED */
    NULL, /* RESERVED */
    NULL, /* RESERVED */
    (tmp_intchar_cast)tpsvrthrininit,
    (tmp_voidvoid_cast)tpsvrthrdone
};

struct tmsvrargs_t *
#ifdef _TMPROTOTYPES
_tmgetsvrargs(void)
#else
_tmgetsvrargs()
#endif
{
    tmsvrargs.reserved1 = NULL;

```



```

tmsvargs.reserved2 = NULL;
tmsvargs.xa_switch = &tmmnull_switch;
return(&tmsvargs);
}

int
#ifdef _TMPROTOTYPES
main(int argc, char **argv)
#else
main(argc,argv)
int argc;
char **argv;
#endif
{
#ifdef TMAINEXIT
#include "mainexit.h"
#endif

return( _tmstartserver( argc, argv, _tmgetsvargs()));
}

*****
delirpt.c
*****

/* FILE: DELIRPT.C
* Microsoft TPC-C Kit Ver. 3.00.000
*
* Copyright Microsoft, 1996
*
* PURPOSE: Delivery report processing application
* Author: Philip Durr
* philipdu@Microsoft.com
*/

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

#define LOGFILE_READ_EOF 0 //check log file flag
return current state
#define LOGFILE_CLEAR_EOF 1 //clear end of log file
flag
#define LOGFILE_SET_EOF 2 //set flag end of log
file reached

#define INTERVAL .01 //90th percentile
calculation bucket interval

#define ERR_SUCCESS 1000 //success no error
#define ERR_READING_LOGFILE 1001 //io errors occured
reading delivery log file
#define ERR_INSUFFICIENT_MEMORY 1002 //insufficient
memory to process 90th percentile report
#define ERR_CANNOT_OPEN_RESULTS_FILE 1005 //Cannot open
delivery results file delilog.

#define TRUE 1
#define FALSE 0

typedef int BOOL;

typedef struct _DelTime
{
struct tm dtime;
int wMilliseconds;
} DelTime;

typedef struct _RPTLINE
{
DelTime start; //delilog report line start
time
DelTime end; //delilog report line end time
int response; //delilog report line time
delivery took in milliseconds
int w_id; //delilog report line warehouse
id for delivery
int o_carrier_id; //delilog report line carrier
id for delivery
int items[10]; //delilog report line
delivery line items
int day;
} RPTLINE, *PRPTLINE;

//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;

int versionMS = 3; //delirpt version
int versionMM = 0;
int versionLS = 2;
int iReport; //delirpt report to process
int iStartTime; //begin times to accept for
report
int iEndTime; //end times to accept for report
int StartDay;

```

```

int OverMidnight=0;

FILE *fpLog; //log file stream

//Local function prototypes
int main(int argc, char *argv[]);
static int Init(void);
static void Restore(void);
static int DoReport(void);
int AverageResponse(void);
int SkippedDelivery(void);
int Percentile90th(void);
int CheckTimes(PRPTLINE pRptLine);
static int OpenLogFile(void);
static void CloseLogFile(void);
static void ResetLogFile(void);
static BOOL LogEOF(int iOperation);
static BOOL ReadReportLine(char *szBuffer, PRPTLINE pRptLine);
static BOOL ParseReportLine(char *szLine, PRPTLINE pRptLine);
static BOOL ParseDate(char *szDate, DelTime *pTime);
static BOOL ParseTime(char *szTime, DelTime *pTime);
static void ErrorMessage(int iError);
static BOOL GetParameters(int argc, char *argv[]);
static void PrintParameters(void);
static void cls(void);
static BOOL IsNumeric(char *ptr);

/* FUNCTION: int main(int argc, char *argv[])
*
* PURPOSE: This function is the beginning execution point for the
delivery executable.
*
* ARGUMENTS: int argc number of command line arguments passed
to delivery
* char *argv[] array of command line argument pointers
*
* RETURNS: None
*
* COMMENTS: None
*/

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

if ( GetParameters(argc, argv) )
{
PrintParameters();
return -1;
}

if ( (iError=Init()) != ERR_SUCCESS )
{
ErrorMessage(iError);
Restore();
return -1;
}

if ( (iError = DoReport()) != ERR_SUCCESS )
ErrorMessage(iError);

Restore();

return 0;
}

/* FUNCTION: static int Init(void)
*
* PURPOSE: This function initializes the delirpt application.
*
* ARGUMENTS: None
*
* RETURNS: None
*
* COMMENTS: None
*/

static int Init(void)
{
int iError;

if ( (iError = OpenLogFile()) )
return iError;
return TRUE;
}

/* FUNCTION: static void Restore(void)
*
* PURPOSE: This function cleans up the delirpt application before
termination.
*
* ARGUMENTS: None
*
* RETURNS: None
*
* COMMENTS: None
*/

```

```

static void Restore(void)
{
    CloseLogFile();
    return;
}

/* FUNCTION: static int DoReport(void)
 * PURPOSE: This function dispatches the requested report.
 * ARGUMENTS: None
 * RETURNS: ERR_SUCCESS if successfull or error code if an error occurs.
 * COMMENTS: None
 */

static int DoReport(void)
{
    int iRc;

    switch(iReport)
    {
        case 1:
            iRc = AverageResponse();
            break;
        case 2:
            iRc = Percentile90th();
            break;
        case 3:
            iRc = SkippedDelivery();
            break;
        case 4:
            if ( (iRc = AverageResponse()) != ERR_SUCCESS )
                break;
            if ( (iRc = Percentile90th()) != ERR_SUCCESS )
                break;
            if ( (iRc = SkippedDelivery()) != ERR_SUCCESS )
                break;
            break;
    }
    return iRc;
}

/* FUNCTION: int AverageResponse(void)
 * PURPOSE: This function processes the AverageResponse report.
 * ARGUMENTS: None
 * RETURNS: ERR_SUCCESS if successfull or error code if an error occurs.
 * COMMENTS: None
 */

int AverageResponse(void)
{
    RPTLINE reportLine;
    unsigned long iTotResponse;
    unsigned long iLines;
    double fAverage;
    char szDelivery[128];

    ResetLogFile();

    iTotResponse = 0;
    iLines = 0;
    printf("\n\n***** Average Response Time Report *****\n");
    while ( !LogEOF(LOGFILE_READ_EOF) )
    {
        if ( ReadReportLine(szDelivery, &reportLine) )
            return ERR_READING_LOGFILE;
        if ( szDelivery[0] == '*' )
            continue;
        if ( !LogEOF(LOGFILE_READ_EOF) )
        {
            if ( CheckTimes(&reportLine) )
                continue;
            iLines++;
            iTotResponse += reportLine.response;

            if ( iLines % 10 == 0 )
                printf("Reading Report Line:\t%d\r", iLines);
        }
    }
    printf("                                \r");
    if ( iLines == 0 )
    {
        printf("No deliveries found.\n");
    }
    else
    {
        fAverage = (iTotResponse / iLines)/1000.0;
        printf("Total Deliveries:      %u\n", iLines);
        printf("Total Response Times:    %10.3f (sec)\n",
            (iTotResponse/1000.0));
    }
}

```

```

        printf("Average Response Time: %10.3f (sec)\n", fAverage);
    }
    return ERR_SUCCESS;
}

/* FUNCTION: int Percentile90th(void)
 * PURPOSE: This function processes the 90th percentile report.
 * ARGUMENTS: None
 * RETURNS: ERR_SUCCESS if successfull or error code if an error occurs.
 * COMMENTS: This function requires enough space to allocate
    needed
    buckets which will be 2 * max response time in
    deci-seconds.
 */

int Percentile90th(void)
{
    RPTLINE reportLine;
    int iBucketSize;
    int i;
    long iMaxSeconds;
    int iTotBuckets;
    double iTot;
    double i90thPercent;
    short *psBuckets;
    char szDelivery[128];

    printf("\n\n***** 90th Percentile *****\n");
    printf("Calculating Max Response Seconds...\n");

    ResetLogFile();

    iMaxSeconds = -1;
    while ( !LogEOF(LOGFILE_READ_EOF) )
    {
        if ( ReadReportLine(szDelivery, &reportLine) )
            return ERR_READING_LOGFILE;
        if ( szDelivery[0] == '*' )
            continue;
        if ( !LogEOF(LOGFILE_READ_EOF) )
        {
            if ( iMaxSeconds < reportLine.response )
                iMaxSeconds = reportLine.response;
        }
    }

    iTotBuckets = iMaxSeconds + 1;

    printf("Allocating Buckets...\n");

    iBucketSize = iTotBuckets * sizeof(short);

    if ( !(psBuckets = (short *)malloc(iBucketSize)) )
        return ERR_INSUFFICIENT_MEMORY;

    /*
    ZeroMemory(psBuckets, iBucketSize);
    */

    iTot = 0;

    ResetLogFile();
    printf("Calculating Distribution...\n");

    while ( !LogEOF(LOGFILE_READ_EOF) )
    {
        if ( ReadReportLine(szDelivery, &reportLine) )
            return ERR_READING_LOGFILE;
        if ( szDelivery[0] == '*' )
            continue;
        if ( !LogEOF(LOGFILE_READ_EOF) )
        {
            if ( CheckTimes(&reportLine) )
                continue;
            psBuckets[reportLine.response]++;
            iTot++;
        }
    }

    i90thPercent = iTot * .9;

    for(i=0, iTot = 0.0; iTot < i90thPercent; iTot +=
        (double)psBuckets[i] )
        i++;

    printf("90th Percentile = %d.%d\n", i/1000, (i % 1000));

    free(psBuckets);

    return ERR_SUCCESS;
}

/* FUNCTION: int SkippedDelivery(void)

```

```

*
* PURPOSE: This function processes the Skipped Deliveries
report.
*
* ARGUMENTS: None
*
* RETURNS: ERR_SUCCESS if successfull or error code if an error
occurs.
*
* COMMENTS: None
*/

int SkippedDelivery(void)
{
    RPTLINE reportLine;
    char szDelivery[128];
    int i;
    int items[10];

    ResetLogFile();

    printf("\n\n***** Skipped Delivery Report *****\n");
    memset(items, 0, sizeof(items));
    printf("Reading Delivery Log File...");

    while ( !LogEOF(LOGFILE_READ_EOF) )
    {
        if ( ReadReportLine(szDelivery, &reportLine) )
            return ERR_READING_LOGFILE;
        if ( szDelivery[0] == '*' )
            continue;
        if ( !LogEOF(LOGFILE_READ_EOF) )
        {
            if ( CheckTimes(&reportLine) )
                continue;
            for(i=0; i<10; i++)
            {
                if ( !reportLine.items[i] )
                    items[i]++;
            }
        }
    }
    printf("\n");
    printf("Skipped delivery table.\n");
    printf(" 1 2 3 4 5 6 7 8 9 10 \n");
    printf("-----\n");
    for(i=0; i<10; i++)
        printf("%4.4d ", items[i]);
    printf("\n");

    return ERR_SUCCESS;
}

/* FUNCTION: BOOL CheckTimes(PRPTLINE pRptLine)
*
* PURPOSE: This function checks to see of the delilog record falls
withing the
* begin and end time from the command line.
*
* ARGUMENTS: PRPTLINE pRptLine delilog processed report line.
*
* RETURNS: BOOL FALSE if report line is not within the
requested start and end times.
* TRUE if the report line is within the
requested start and end times.
*
* COMMENTS: If startTime and endTime are both 0 then the user
requested
* the default behavior which is all records in delilog are
valid.
*/

BOOL CheckTimes(PRPTLINE pRptLine)
{
    int iRptEndTime;
    int iRptStartTime;

    iRptStartTime = (pRptLine->start.dtime.tm_hour * 3600000) +
(pRptLine->start.dtime.tm_min * 60000) + (pRptLine-
->start.dtime.tm_sec * 1000) + pRptLine->start.wMilliseconds;
    iRptEndTime = (pRptLine->end.dtime.tm_hour * 3600000) +
(pRptLine->end.dtime.tm_min * 60000) + (pRptLine->end.dtime.tm_sec
* 1000) + pRptLine->end.wMilliseconds;

    if ( iStartTime == 0 && iEndTime == 0 )
        return FALSE;

    if ( !OverMidnight ) {
        if ( iStartTime <= iRptStartTime && iEndTime >= iRptEndTime )
            return FALSE;
    }
    else {
        if ( pRptLine->day == StartDay ) {
            if ( iStartTime <= iRptStartTime )
                return FALSE;
        }
        else {
            if ( iEndTime >= iRptEndTime )
                return FALSE;
        }
    }
}

```

```

}
return TRUE;
}

/* FUNCTION: int OpenLogFile(void)
*
* PURPOSE: This function opens the delivery log file for use.
*
* ARGUMENTS: None
*
* RETURNS: int ERR_CANNOT_OPEN_RESULTS_FILE Cannot create
results log file.
* ERR_SUCCESS Log file successfully opened
*
* COMMENTS: None
*/

static int OpenLogFile(void)
{
    fpLog = fopen("delilog", "rb");

    if ( !fpLog )
        return ERR_CANNOT_OPEN_RESULTS_FILE;

    return ERR_SUCCESS;
}

/* FUNCTION: int CloseLogFile(void)
*
* PURPOSE: This function closes the delivery log file.
*
* ARGUMENTS: None
*
* RETURNS: None
*
* COMMENTS: None
*/

static void CloseLogFile(void)
{
    if ( fpLog )
        fclose(fpLog);

    return;
}

/* FUNCTION: static void ResetLogFile(void)
*
* PURPOSE: This function prepares the delilog. file for reading
*
* ARGUMENTS: None
*
* RETURNS: None
*
* COMMENTS: None
*/

static void ResetLogFile(void)
{
    fseek(fpLog, 0L, SEEK_SET);
    LogEOF(LOGFILE_CLEAR_EOF);

    return;
}

/* FUNCTION: static BOOL LogEOF(int iOperation)
*
* PURPOSE: This function tracks and reports the end of file
condition
* on the delilog file.
*
* ARGUMENTS: int iOperation requested operation this can be:
* LOGFILE_READ_EOF check log file flag return
current state
* LOGFILE_CLEAR_EOF clear end of log file flag
* LOGFILE_SET_EOF set flag end of log file
reached
*
* RETURNS: None
*
* COMMENTS: None
*/

static BOOL LogEOF(int iOperation)
{
    static BOOL bEOF;

    switch(iOperation)
    {
        case LOGFILE_READ_EOF:
            return bEOF;
            break;
        case LOGFILE_CLEAR_EOF:

```

```

        bEOF = FALSE;
        break;
    case LOGFILE_SET_EOF:
        bEOF = TRUE;
        break;
    }
    return FALSE;
}

/* FUNCTION: static BOOL ReadReportLine(char *szBuffer, PRPTLINE
pRptLine)
*
* PURPOSE: This function reads a text line from the delilog file.
*          on the delilog file.
*
* ARGUMENTS: char *szBuffer buffer to placed read delilog file
line into.
*          PRPTLINE pRptLine returned structure containing parsed
delilog
*          report line.
*
* RETURNS: FALSE if successfull or TRUE if an error occurs.
*
* COMMENTS: None
*/

static BOOL ReadReportLine(char *szBuffer, PRPTLINE pRptLine)
{
    int i = 0;
    int ch;
    int iEof;

    while( i < 128 )
    {
        ch = fgetc(fpLog);
        if ( iEof = feof(fpLog) )
            break;
        if ( ch == '\r' )
        {
            if ( i )
                break;
            continue;
        }
        if ( ch == '\n' )
        {
            continue;
        }
        szBuffer[i++] = ch;
    }

    //delivery item format is to long cannot be a valid delivery item
    if ( i >= 128 )
        return TRUE;

    szBuffer[i] = 0;
    if ( iEof )
    {
        LogEOF(LOGFILE_SET_EOF);
        if ( i == 0 )
            return FALSE;
    }
    if ( szBuffer[0] == '*' )
    {
        //error line ignore
        return FALSE;
    }
    return ParseReportLine(szBuffer, pRptLine);
}

/* FUNCTION: static BOOL ParseReportLine(char *szLine, PRPTLINE
pRptLine)
*
* PURPOSE: This function reads a text line from the delilog file.
*          on the delilog file.
*
* ARGUMENTS: char *szLine buffer containing the delilog file
line to be parsed.
*          PRPTLINE pRptLine returned structure containing parsed
delilog
*          report line values.
*
* RETURNS: FALSE if successfull or TRUE if an error occurs.
*
* COMMENTS: None
*/

static BOOL ParseReportLine(char *szLine, PRPTLINE pRptLine)
{
    int i;

    if ( ParseDate(szLine, (DelTime *) &pRptLine->start) )
        return TRUE;

    pRptLine->end.dtime.tm_year = pRptLine->start.dtime.tm_year;
    pRptLine->end.dtime.tm_mon = pRptLine->start.dtime.tm_mon;
    pRptLine->end.dtime.tm_mday = pRptLine->start.dtime.tm_mday;

```

```

    pRptLine->day=(pRptLine->start.dtime.tm_mon*100) + pRptLine-
>start.dtime.tm_mday;
    if (StartDay == 0) {
        StartDay=pRptLine->day;
        printf("Setting Start Day to %d\n", StartDay);
    }

    if ( !(szLine = strchr(szLine, ',')) )
        return TRUE;
    szLine++;

    if ( ParseTime(szLine, (DelTime *) &pRptLine->start) )
        return TRUE;

    if ( !(szLine = strchr(szLine, ',')) )
        return TRUE;
    szLine++;

    if ( ParseTime(szLine, (DelTime *) &pRptLine->end) )
        return TRUE;

    if ( !(szLine = strchr(szLine, ',')) )
        return TRUE;
    szLine++;

    if ( !IsNumeric(szLine) )
        return TRUE;
    pRptLine->response = atoi(szLine);

    if ( !(szLine = strchr(szLine, ',')) )
        return TRUE;
    szLine++;

    if ( !IsNumeric(szLine) )
        return TRUE;
    pRptLine->w_id = atoi(szLine);

    if ( !(szLine = strchr(szLine, ',')) )
        return TRUE;
    szLine++;

    if ( !IsNumeric(szLine) )
        return TRUE;
    pRptLine->o_carrier_id = atoi(szLine);

    if ( !(szLine = strchr(szLine, ',')) )
        return TRUE;
    szLine++;

    for(i=0; i<10; i++)
    {
        if ( !IsNumeric(szLine) )
            return TRUE;
        pRptLine->items[i] = atoi(szLine);

        if ( i<9 && !(szLine = strchr(szLine, ',')) )
            return TRUE;
        szLine++;
    }

    return FALSE;
}

/* FUNCTION: static BOOL ParseDate(char *szDate, DelTime *pTime)
*
* PURPOSE: This function validates and extracts a date string in
the format
*          yy/mm/dd into an DelTime structure.
*
* ARGUMENTS: char *szDate buffer containing the date to be
parsed.
*          DelTime *pTime system time structure where date will
be placed.
*
* RETURNS: FALSE if successfull or TRUE if an error occurs.
*
* COMMENTS: None
*/

static BOOL ParseDate(char *szDate, DelTime *pTime)
{
    if ( !isdigit(*szDate) || !isdigit(*(szDate+1)) ||
!isdigit(*(szDate+2)) || !isdigit(*(szDate+3)) || *(szDate+4) !=
'/' ||
!isdigit(*(szDate+5)) || !isdigit(*(szDate+6)) || *(szDate+7)
!= '/' ||
!isdigit(*(szDate+8)) || !isdigit(*(szDate+9)) )
        return TRUE;

    pTime->dtime.tm_year = atoi(szDate);
    pTime->dtime.tm_mon = atoi(szDate+5);
    pTime->dtime.tm_mday = atoi(szDate+8);

    if ( pTime->dtime.tm_mon > 12 || pTime->dtime.tm_mon < 0 ||
pTime->dtime.tm_mday > 31 || pTime->dtime.tm_mday < 0 )
        return TRUE;

    return FALSE;
}

```

```

}

/* FUNCTION: static BOOL ParseTime(char *szTime, DelTime *pTime)
 *
 * PURPOSE: This function validates and extracts a time string in
 the format
 *          hh:mm:ss:mmm into an DelTime structure.
 *
 * ARGUMENTS: char      *szTime   buffer containing the time to be
 parsed.
 *              DelTime *pTime   system time structure where date will
 be placed.
 *
 * RETURNS:   FALSE if successfull or TRUE if an error occurs.
 *
 * COMMENTS:  None
 */

static BOOL ParseTime(char *szTime, DelTime *pTime)
{
    if ( !isdigit(*szTime) || !isdigit(*(szTime+1)) || *(szTime+2) !=
 ':' ||
        !isdigit(*(szTime+3)) || !isdigit(*(szTime+4)) || *(szTime+5)
 != ':' ||
        !isdigit(*(szTime+6)) || !isdigit(*(szTime+7)) || *(szTime+8)
 != ':' ||
        !isdigit(*(szTime+9)) || !isdigit(*(szTime+10)) ||
 !isdigit(*(szTime+11)) )
        return TRUE;

    pTime->dtm_hour = atoi(szTime);
    pTime->dtm_min = atoi(szTime+3);
    pTime->dtm_sec = atoi(szTime+6);
    pTime->wMilliseconds = atoi(szTime+9);

    if ( pTime->dtm_hour > 23 || pTime->dtm_hour < 0 ||
        pTime->dtm_min > 59 || pTime->dtm_min < 0 ||
        pTime->dtm_sec > 59 || pTime->dtm_sec < 0 ||
        pTime->wMilliseconds < 0 )
        return TRUE;

    if ( pTime->wMilliseconds > 999 )
    {
        pTime->dtm_sec += (pTime->wMilliseconds/1000);
        pTime->wMilliseconds = pTime->wMilliseconds % 1000;
    }

    return FALSE;
}

/* FUNCTION: void ErrorMessage(int iError)
 *
 * PURPOSE: This function displays an error message in the delivery
 executable's console window.
 *
 * ARGUMENTS: int   iError   error id to be displayed
 *
 * RETURNS:   None
 *
 * COMMENTS:  None
 */

static void ErrorMessage(int iError)
{
    int i;

    static SERRORMSG errorMsgs[] =
    {
        { ERR_SUCCESS,          "Success, no error."
        },
        { ERR_CANNOT_OPEN_RESULTS_FILE,    "Cannot open delivery
results file delilog."
        },
        { ERR_READING_LOGFILE,            "Reading delivery log file,
Delivery item format incorrect."
        },
        { ERR_INSUFFICIENT_MEMORY,        "insufficient memory to
process 90th percentile report."
        },
        { 0,                            ""
        }
    };

    for(i=0; errorMsgs[i].szMsg[0]; i++)
    {
        if ( iError == errorMsgs[i].iError )
        {
            printf("\nError(%d): %s\n", iError, errorMsgs[i].szMsg);
            return;
        }
    }
    printf("Error(%d): %s", errorMsgs[0].szMsg);
    return;
}

/* FUNCTION: BOOL GetParameters(int argc, char *argv[])
 *
 * PURPOSE: This function parses the command line passed in to the
 delivery executable, initializing
 *          and filling in global variable parameters.
 *
 * ARGUMENTS: int   argc   number of command line arguments passed
 to delivery

```

```

 *          char *argv[] array of command line argument pointers
 *
 * RETURNS:   BOOL   FALSE parameter read successfull
 *              TRUE  user has requested parameter information screen
 be displayed.
 *
 * COMMENTS:  None
 */

static BOOL GetParameters(int argc, char *argv[])
{
    int i;
    DelTime startTime;
    DelTime endTime;

    iStartTime = 0;
    iEndTime = 0;
    iReport = 4;

    for(i=0; i<argc; i++)
    {
        if ( argv[i][0] == '-' || argv[i][0] == '/' )
        {
            switch(argv[i][1])
            {
                case 'S':
                case 's':
                    if ( ParseTime(argv[i]+2, &startTime) )
                        return TRUE;
                    iStartTime = (startTime.dtm_hour * 3600000) +
(startTime.dtm_min * 60000) + (startTime.dtm_sec * 1000)
+ startTime.wMilliseconds;
                    break;
                case 'E':
                case 'e':
                    if ( ParseTime(argv[i]+2, &endTime) )
                        return TRUE;
                    iEndTime = (endTime.dtm_hour * 3600000) +
(endTime.dtm_min * 60000) + (endTime.dtm_sec * 1000) +
endTime.wMilliseconds;
                    if (iStartTime > iEndTime)
                        OverMidnight=1;
                    break;
                case 'R':
                case 'r':
                    iReport = atoi(argv[i]+2);
                    if ( iReport > 4 || iReport < 1 )
                        iReport = 4;
                    break;
                case '?':
                    return TRUE;
            }
        }
    }
    return FALSE;
}

/* FUNCTION: void PrintParameters(void)
 *
 * PURPOSE: This function displays the supported command line
 flags.
 *
 * ARGUMENTS: None
 *
 * RETURNS:   None
 *
 * COMMENTS:  None
 */

static void PrintParameters(void)
{
    printf("DELIRPT:\n\n");
    printf("Parameter
Default\n");
    printf("-----\n");
    printf("-S Start Time HH:MM:SS:MMM
All \n");
    printf("-E End Time HH:MM:SS:MMM
All \n");
    printf("-R 1)Average Response, 2)90th 3) Skipped 4) All
All \n");
    printf("-? This help screen\n\n");
    printf("Note: Command line switches are NOT case sensitive.\n");

    return;
}

/* FUNCTION: void cls(void)
 *
 * PURPOSE: This function clears the console window
 *
 * ARGUMENTS: None
 *
 * RETURNS:   None
 *
 * COMMENTS:  None
 */

```

```

*/
static void cls(void)
{
    system("clear");

    return;
}

/* FUNCTION: BOOL IsNumeric(char *ptr)
 *
 * PURPOSE: This function determines if a string is numeric. It
 * fails if any characters other
 * than numeric and null terminator are present.
 *
 * ARGUMENTS: char *ptr pointer to string to check.
 *
 * RETURNS:  BOOL FALSE if string is not all numeric
 *          TRUE  if string contains only numeric characters i.e.
 *          '0' - '9'
 *
 * COMMENTS: A comma is counted as a valid delimiter.
 */

static BOOL IsNumeric(char *ptr)
{
    if ( *ptr == 0 )
        return FALSE;

    while( *ptr && isdigit(*ptr) )
        ptr++;
    if ( !*ptr || *ptr == ',' )
        return TRUE;
    else
        return FALSE;
}

*****
logfile_mod.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 tpc benchmark.
 *
 * Author: W Carr
 *
 * Creation Date: October 1997
 *
 *
 * Modification history:
 *
 *
 *      08/01/2002      Andrew Bond, HP
 *
 *                      - Conversion to run under Linux and Apache
 *
 */

#include <stdio.h>
#include <stdarg.h>
#include <time.h>
#include <sys/time.h>
#include <errno.h>
#include <unistd.h>

#include "apr_thread_mutex.h"

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

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

#include <tpcc.h>

static FILE *LogFile;

static char t1[1];
static apr_thread_mutex_t * ErrCriticalSection;
static apr_thread_mutex_t * LogCriticalSection;

/* FUNCTION: void TPCCOpenLog( void )

```

```

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

    apr_thread_mutex_create(&LogCriticalSection, 0, pool);

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

    if (LogFile = fopen( szFile, "a" )) {
        apr_thread_mutex_create(&ErrCriticalSection, 0, pool);
        return TRUE;
    }
    else
    {
        return FALSE;
    }
}

/* FUNCTION: void TPCCcloseLog( void )
*
* PURPOSE: This function closes the log file.
*
* ARGUMENTS: None
*
* RETURNS: None
*
* COMMENTS: None
*/
BOOL
TPCCcloseLog( void )
{
    fclose( LogFile );

    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    szArg[4096];
    struct timezone    tz;
    struct timeval    tv;
    struct tm    systemTime;
    struct tm    *pst;
    int    len, ret;

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

    pst=&systemTime;
    ret=gettimeofday(&tv, &tz);

    apr_thread_mutex_lock( LogCriticalSection );

    pst=localtime(&tv.tv_sec);

    len = fprintf( stderr,
        "[%ld] %2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d\t%s\r\n",
        getpid(),
        1900+pst->tm_year, pst->tm_mon+1, pst->tm_mday,
        pst->tm_hour, pst->tm_min, pst->tm_sec,
        szArg );
    apr_thread_mutex_unlock( LogCriticalSection );
}

void
TPCCerrInternal( char *szTmp, int len )
{
    int    dwWriteLen;
    FILE    *ErrFile;

```

```

char      szFile[FILENAME_SIZE];

apr_thread_mutex_lock( ErrCriticalSection );

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

ErrFile = fopen( szFile, "a" );

if (ErrFile) {
    len = fprintf( ErrFile, "%s\n", szTmp );
    fclose( ErrFile );
}

apr_thread_mutex_unlock( ErrCriticalSection );
}

void
TPCCErr( char *fmt, ... )
{
    va_list  marker;
    char      szTmp[4096];
    char      szArg[4096];
    struct timezone tz;
    struct timeval tv;
    struct tm  systemTime;
    struct tm  *pst;
    int      len, ret;

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

    pst=&systemTime;
    ret=gettimeofday(&tv, &tz);
    pst=localtime(&tv.tv_sec);

    len = sprintf( szTmp,
        "%2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d\t%s\r\n",
        1900+pst->tm_year, pst->tm_mon+1, pst->tm_mday,
        pst->tm_hour, pst->tm_min, pst->tm_sec,
        pConn->w_id, pConn->ld_id, pConn->pCC,
        pConn->status, pConn->dbstatus,
        szArg );

    TPCCErrInternal( szTmp, len );
}

void
TPCCTransactionErr( pConnData pConn, char *fmt, ... )

```

```

{
    va_list  marker;
    char      szTmp[4096];
    char      szArg[4096];
    struct timezone tz;
    struct timeval tv;
    struct tm  systemTime;
    struct tm  *pst;
    int      len, ret;

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

    pst=&systemTime;
    ret=gettimeofday(&tv, &tz);
    pst=localtime(&tv.tv_sec);

    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",
        1900+pst->tm_year, pst->tm_mon+1, pst->tm_mday,
        pst->tm_hour, pst->tm_min, pst->tm_sec,
        pConn->w_id, pConn->ld_id, pConn->pCC,
        pConn->status, pConn->dbstatus,
        szArg );

    TPCCErrInternal( szTmp, len );
}

*****
logfile_tux.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
*
*
* Modification history:
*
*
* 08/01/2002 Andrew Bond, HP
*
* - Conversion to run under Linux and Apache
*
*/

#include <stdio.h>
#include <stdarg.h>
#include <time.h>
#include <sys/time.h>

#include <tpccstruct.h>

static FILE *LogFile;

void
TPCCErr( char *fmt, ... )
{
    va_list marker;
    char szTmp[4096];
    char szArg[4096];
    struct timezone tz;
    struct timeval tv;
    struct tm systemTime;

```

```

struct tm *pst;
int len, ret;

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

pst=&systemTime;
ret=gettimeofday(&tv, &tz);
pst=localtime(&tv.tv_sec);

len = userlog( "%2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d\t%s\r\n",
1900+pst->tm_year, pst->tm_mon+1, pst->tm_mday,
pst->tm_hour, pst->tm_min, pst->tm_sec,
szArg );

if (len < 0)
printf("TPCCErr: Error writing to Tuxedo userlog\n");

}

*****
mod_tpcc.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
 *
 *
 * Modification history:
 *
 *
 * 08/01/2002 Andrew Bond, HP
 *
 * - Conversion to run under Linux and Apache
 *
 * - Additions by Joe Orton to support Apache 2.0
 */
#include "httpd.h"
#include "http_config.h"
#include "http_protocol.h"
#include "ap_config.h"
#include "ap_mpm.h"
#include "apr_thread_mutex.h"

#include <stdio.h>
#include <stdarg.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

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

#define MOD_TPCC_C
#include <tpccerr.h>
#include <tpccstruct.h>
#include <oracle_db8.h>
#include <tpccapi.h>

#include <tpcc.h>
#include <mod_tpcc.h>

```

```

#ifdef FFE_DEBUG
# include <crtdbg.h>
static int tmpDbgFlag;
static _HFILE hMemFile;
#endif

int tpcc_handler(request_rec *req);

static int tpcc_post_config(apr_pool_t *p, apr_pool_t *plog,
apr_pool_t *ptemp, server_rec *s);

static void tpcc_child_init(apr_pool_t *p, server_rec *s);

static apr_status_t tpcc_child_exit(void *data);

#define FORMMAXSIZE 4096

#define MYFILE "/etc/httpd/logs/tpcc.log"
#define BOGUS "Bogus File!"
#define GOOD "Good File!"

int LogFD;
int myerrno;
int max_threads;

static void tpcc_register_hooks(apr_pool_t *p)
{
    fprintf(stderr, "register()");

    ap_hook_handler(tpcc_handler, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_post_config(tpcc_post_config, NULL, NULL,
APR_HOOK_MIDDLE);
    /*
    ap_hook_child_init(tpcc_child_init, NULL, NULL,
APR_HOOK_MIDDLE);
    */
}

/* Dispatch list for API hooks */
module AP_MODULE_DECLARE_DATA tpcc_module = {
    STANDARD20_MODULE_STUFF,
    /* NULL,          /* create per-dir    config structures
    *
    * NULL,          /* merge per-dir    config structures
    *
    * NULL,          /* create per-server config structures
    *
    * NULL,          /* merge per-server config structures
    *
    * NULL,          /* table of config file commands
    *
    tpcc_register_hooks /* register hooks
    */
};

#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)\
{\
    apr_thread_mutex_lock( gpForms->critSec[type] );\
    FFE_ASSERT( gpForms->iNextFreeForm[type] <= gpForms->iMaxIndex[type] );\
    szForm = gpForms->index[gpForms->iFirstFormIndex[type] +\
        gpForms->iNextFreeForm[type]++];\
    apr_thread_mutex_unlock( gpForms->critSec[type] );\
}

#define UNRESERVE_FORM(type,szForm)\
{\
    apr_thread_mutex_lock( gpForms->critSec[type] );\
    FFE_ASSERT( gpForms->iNextFreeForm[type] > 0 );\
    gpForms->index[gpForms->iFirstFormIndex[type] +\
        --gpForms->iNextFreeForm[type]] = szForm;\
    apr_thread_mutex_unlock( gpForms->critSec[type] );\
}

#define RESERVE_RESPONSE(type,szResponse)\
{\
    apr_thread_mutex_lock( gpResponses->critSec[type] );\
    FFE_ASSERT(gpResponses->iNextFreeResponse[type]<=gpResponses->iMaxIndex[type]);\
    szResponse = gpResponses->index[gpResponses->iFirstResponseIndex[type] +\
        gpResponses->iNextFreeResponse[type]++];\
    apr_thread_mutex_unlock( gpResponses->critSec[type] );\
}

#define UNRESERVE_RESPONSE(type,szResponse)\
{\
    apr_thread_mutex_lock( gpResponses->critSec[type] );\

```

```

    FFE_ASSERT(gpResponses->iNextFreeResponse[type] > 0 );\
    gpResponses->index[gpResponses->iFirstResponseIndex[type] +\
        --gpResponses->iNextFreeResponse[type]] = szResponse;\
    apr_thread_mutex_unlock( gpResponses->critSec[type] );\
}

#define RESERVE_PANIC_FORM(szForm)\
{\
    apr_thread_mutex_lock( gpPanicForms->critSec );\
    FFE_ASSERT( gpPanicForms->iNextFree <= gpPanicForms->iMaxIndex );\
    szForm = gpPanicForms->index[gpPanicForms->iNextFree++];\
    apr_thread_mutex_unlock( gpPanicForms->critSec );\
}

#define UNRESERVE_PANIC_FORM(szForm)\
{\
    apr_thread_mutex_lock( gpPanicForms->critSec );\
    FFE_ASSERT( gpPanicForms->iNextFree > 0 );\
    gpPanicForms->index[--gpPanicForms->iNextFree] = szForm;\
    apr_thread_mutex_unlock( 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)+FILENAME_SIZE];
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>"
"</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=5> <INPUT NAME=B SIZE=6>"
"                <INPUT NAME=C SIZE=1><BR>"
" <INPUT NAME=D SIZE=5> <INPUT NAME=E SIZE=6>"
"                <INPUT NAME=F SIZE=1><BR>"
" <INPUT NAME=G SIZE=5> <INPUT NAME=H SIZE=6>"
"                <INPUT NAME=I SIZE=1><BR>"
" <INPUT NAME=J SIZE=5> <INPUT NAME=K SIZE=6>"
"                <INPUT NAME=L SIZE=1><BR>"
" <INPUT NAME=M SIZE=5> <INPUT NAME=N SIZE=6>"
"                <INPUT NAME=O SIZE=1><BR>"
" <INPUT NAME=P SIZE=5> <INPUT NAME=Q SIZE=6>"
"                <INPUT NAME=R SIZE=1><BR>"
" <INPUT NAME=S SIZE=5> <INPUT NAME=T SIZE=6>"
"                <INPUT NAME=U SIZE=1><BR>"
" <INPUT NAME=V SIZE=5> <INPUT NAME=W SIZE=6>"
"                <INPUT NAME=X SIZE=1><BR>"
" <INPUT NAME=a SIZE=5> <INPUT NAME=b SIZE=6>"
"                <INPUT NAME=c SIZE=1><BR>"
" <INPUT NAME=d SIZE=5> <INPUT NAME=e SIZE=6>"
"                <INPUT NAME=f SIZE=1><BR>"
" <INPUT NAME=g SIZE=5> <INPUT NAME=h SIZE=6>"
"                <INPUT NAME=i SIZE=1><BR>"
" <INPUT NAME=j SIZE=5> <INPUT NAME=k SIZE=6>"
"                <INPUT NAME=l SIZE=1><BR>"
" <INPUT NAME=m SIZE=5> <INPUT NAME=n SIZE=6>"
"                <INPUT NAME=o SIZE=1><BR>"
" <INPUT NAME=p SIZE=5> <INPUT NAME=q SIZE=6>"
"                <INPUT NAME=r SIZE=1><BR>"
" <INPUT NAME=s SIZE=5> <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>"
"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>"

```



```

"<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[] =
"###\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( request_rec *req );
void CheckpointCmd( request_rec *req, int w_id, int ld_id );
void CheckpointStartupCmd( request_rec *req, int w_id, int ld_id );
void ClearCmd( request_rec *req );
void ExitCmd( request_rec *req );
void MenuCmd( request_rec *req, int w_id, int ld_id );
void SubmitCmd( request_rec *req, int *w_id, int *ld_id );
void MemoryCheckCmd( request_rec *req, 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( int dwResponseSize, apr_pool_t *p );
void MakeTemplatePool( int dwFormSize, int dwResponseSize,
apr_pool_t *p);
void MakeTransactionPool( int dwTransactionPoolSize, apr_pool_t
*p);
void DeletePanicPool( void );
void DeleteTemplatePool( void );
void DeleteTransactionPool( void );

int ProcessDeliveryQuery( request_rec *req,
char *the_request,
int w_id, int ld_id );
int ProcessNewOrderQuery( request_rec *req,
char *the_request,
int w_id, int ld_id );
int ProcessOrderStatusQuery( request_rec *req,
char *the_request,
int w_id, int ld_id );
int ProcessPaymentQuery( request_rec *req,
char *the_request,
int w_id, int ld_id );
int ProcessStockLevelQuery( request_rec *req,
char *the_request,
int w_id, int ld_id );

int ProcessQueryString( request_rec *req);

void PutNumeric( int iInt, int iFieldSize, char *pChar );
void SendErrorResponse( request_rec *req, int iError,
int iErrorType, char *szMsg, int w_id, int ld_id,
pConnData pConn );
void SendMainMenuForm( request_rec *req,
int w_id, int ld_id, char *szStatus );
void SendResponse(request_rec *req, char *szStr, int iStrLen);
void SendWelcomeForm(request_rec *req);
#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
{

```



```

    apr_thread_mutex_t * critSec;
#ifdef FFE_DEBUG
    int iMaxIndex;
#endif
int iNextFree;
char *index[1];
char forms[APANIC_FORM_SIZE];
} PanicStruct, *pPanicStruct;

typedef struct
{
    apr_thread_mutex_t * critSec[NUMBER_POOL_FORM_TYPER];
#ifdef FFE_DEBUG
    int iMaxIndex[NUMBER_POOL_FORM_TYPER];
#endif
    int iNextFreeForm[NUMBER_POOL_FORM_TYPER];
    int iFirstFormIndex[NUMBER_POOL_FORM_TYPER];
    char *index[1];
    char forms[1];
} FormStruct, *pFormStruct;

typedef struct
{
    apr_thread_mutex_t * critSec[NUMBER_POOL_RESPONSE_TYPER];
#ifdef FFE_DEBUG
    int iMaxIndex[NUMBER_POOL_RESPONSE_TYPER];
#endif
    int iNextFreeResponse[NUMBER_POOL_RESPONSE_TYPER];
    int iFirstResponseIndex[NUMBER_POOL_RESPONSE_TYPER];
    char *index[1];
    char responses[1];
} ResponseStruct, *pResponseStruct;

/* global variables */
static int iInitStatus = FALSE;

static apr_thread_mutex_t * startupspinlock;
static BOOL startupFlag = FALSE;

static pPanicStruct gpPanicForms = NULL;
static int giPanic = 0;
static pFormStruct gpForms = 0;
static int giFormLen[NUMBER_POOL_FORM_TYPER] = { 0 };
static pResponseStruct gpResponses = 0;
static int giResponseLen[NUMBER_POOL_RESPONSE_TYPER] = { 0 };

/* FUNCTION: BOOL APIENTRY DllMain(HANDLE hModule, int
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
* int ul_reason_for_call reason for call
* LPVOID lpReserved reserved for future use
*
* RETURNS: BOOL Always TRUE Errors in initialization
* are presented at the first
* screen to the user.
* COMMENTS: None
*
*/

static int tpcc_post_config(apr_pool_t *p, apr_pool_t *plog,
apr_pool_t *ptemp, server_rec *s)
{
    if (iInitStatus == FALSE) {
        apr_thread_mutex_create(&startupspinlock, 0, p);

        LogFD=open(MYFILE, O_CREAT|O_RDWR);
        myerrno=errno;
        MyLogFile=fdopen(LogFD, "a+");
        if (LogFD == -1)
        {
            printf("Bad file open, errno=%d\n", myerrno);
        }

        iInitStatus=TRUE;

        TPCCOpenLog(s->process->pool);

        ap_mpm_query(AP_MPMQ_MAX_THREADS, &max_threads);

#ifdef (DEBUG == 1)
        fprintf(MyLogFile, "tpcc_post_config, pid=%d\n", getpid());
        fprintf(MyLogFile, "s->path: %s\n", s->path);
        fprintf(MyLogFile, "s->port: %d\n", s->port);
        fprintf(MyLogFile, "s->server_hostname: %s\n", s-
>server_hostname);
        fprintf(MyLogFile, "s->error_fname: %s\n", s->error_fname);
        fprintf(MyLogFile, "Max threads = %d\n", max_threads);
        fflush(MyLogFile);
#endif
    }

    return OK;
}

static void tpcc_child_init(apr_pool_t *p, server_rec *s)

```

```

{

#if (DEBUG == 1)
    fprintf(MyLogFile, "In tpcc_child_init\n");
    fflush(MyLogFile);
#endif

}

static apr_status_t tpcc_child_exit(void *data)
{
#if (DEBUG == 1)
    fprintf(MyLogFile, "In tpcc_child_exit\n");
    fflush(MyLogFile);
#endif

    TPCCShutdown( );

    DeleteTransactionPool( );
    DeleteTemplatePool( );
    DeletePanicPool( );

    TPCCCloseLog( );
}

/* FUNCTION: int tpcc_handler(request_rec *req)
 *
 * PURPOSE: This function is the main entry point for the TPCC DLL.
 * The internet service calls this function passing in the
 * http string.
 *
 * ARGUMENTS: request_rec *req structure ptr containing the
 * internet service information.
 *
 * RETURNS: int HSE_STATUS_SUCCESS connection can be dropped if
 * error
 * HSE_STATUS_SUCCESS_AND_KEEP_CONN keep connect valid
 * comment sent
 *
 * COMMENTS: None
 *
 */

int tpcc_handler(request_rec *req)
{
    int status;
    int dbstatus;

    /* TPCCLog("now in handler"); */

```

```

if ( ! startupFlag ) {
    apr_thread_mutex_lock( startupspinlock );
    if ( ! startupFlag ) {

#if (DEBUG == 1)
        fprintf(MyLogFile, "tpcc_handler: Startup Section\n");
#endif

        if ( ERR_SUCCESS != ( iInitStatus = ReadRegistrySettings( )) )
            MakePanicPool( 50, req->pool ); /* make room for error
            messages */
        else {
            dbstatus = TPCCStartup( );
            if( ERR_DB_SUCCESS != dbstatus ) {
                iInitStatus = dbstatus;
            }
        }

        {
            apr_pool_t *ppool = req->server->process->pool;

            strcpy(szModName, req->uri);

            MakeTemplatePool(max_threads, max_threads, ppool);
            MakePanicPool(max_threads, ppool);
            MakeTransactionPool(max_threads, ppool);
        }

        startupFlag = TRUE;
    }
    apr_thread_mutex_unlock( startupspinlock );
}

#if (DEBUG == 1)
    fprintf(MyLogFile, "tpcc_handler: iInitStatus=%d\n",
iInitStatus);
#endif

if( ERR_SUCCESS != iInitStatus )
{
    SendErrorResponse(req, iInitStatus, ERR_TYPE_WEBDLL, NULL, -1,
-1, NULL);
    return TRUE;
}

#if (DEBUG == 1)
    fprintf(MyLogFile, "req->the_request: %s\n", req-
>the_request);
    fprintf(MyLogFile, "req->unparsed_uri: %s\n", req-
>unparsed_uri);
    fprintf(MyLogFile, "req->uri: %s\n", req->uri);
    fprintf(MyLogFile, "req->filename: %s\n", req->filename);

```

```

        fprintf(MyLogFile, "req->args: %s\n", req->args);
        fflush(MyLogFile);
    #endif

    /* process http query */
    status = ProcessQueryString(req);

    /* finish up with status returned by Processing functions */
    return OK;
}

/* FUNCTION: void SendErrorResponse( request_rec *req, int iError,
 *
 *      int iErrorType, char *szMsg,
 *
 *      int w_id, int ld_id )
 *
 *
 * PURPOSE: This function displays an error form in the client
browser.
 *
 * ARGUMENTS: request_rec *req  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( request_rec *req, 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;

    #if (DEBUG == 1)
        fprintf(MyLogFile, "Entering SendErrorResponse\n");
        fflush(MyLogFile);
    #endif

    RESERVE_PANIC_FORM( szForm );

    #if (DEBUG == 1)
        fprintf(MyLogFile, "After Reserve Form\n");
        fflush(MyLogFile);
    #endif

    if( ERR_TYPE_WEBDLL == iErrorType )
    {
        ii = 0;
        while( '\0' != errorMsgs[ii].szMsg[0] && iError !=
errorMsgs[ii].iError )
            ii++;
    #if (DEBUG == 1)
        fprintf(MyLogFile, "After while\n");
        fflush(MyLogFile);
    #endif

        if ( '\0' == errorMsgs[ii].szMsg[0] )
            ii = 1; /* ERR_NO_MESSAGE */
        szErrorTypeMsg = "TPCCWEB";
        szErrorMsg = errorMsgs[ii].szMsg;
    }
    else if( ERR_TYPE_DBLIB == iErrorType )
    {
        szErrorTypeMsg = "DBLIB";
        szErrorMsg = szMsg;
    }
    #if (DEBUG == 1)
        fprintf(MyLogFile, "After Reserve Form\n");
        fflush(MyLogFile);
    #endif

    /*
    if( NULL != pConn )
        TPCCTransactionErr( pConn, "%s(%d): %s\r\n",
            szErrorTypeMsg, iError, szErrorMsg );
    else
    */
    TPCCErr( "%s(%d): %s\r\n", szErrorTypeMsg, iError, szErrorMsg
);
    #if (DEBUG == 1)
        fprintf(MyLogFile, "szErrorMsg=%s\n", szErrorMsg);
        fflush(MyLogFile);
    #endif

```

```

#endif

    iStrLen = sprintf( szForm, szErrorFormTemplate, req->uri,
        WDID(w_id,ld_id), iError, szErrorTypeMsg, szErrorMsg );

    #if (DEBUG == 1)
        fprintf(MyLogFile, "szForm=%s\n", szForm);
        fflush(MyLogFile);
    #endif

    #if (DEBUG == 1)
        fprintf(MyLogFile, "SendErrorResponse: Before
SendResponse\n");
        fflush(MyLogFile);
    #endif

    SendResponse(req, szForm, iStrLen);

    #if (DEBUG == 1)
        fprintf(MyLogFile, "SendErrorResponse: After
SendResponse\n");
        fflush(MyLogFile);
    #endif

    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(request_rec *req, 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:
 *
 * request_rec *req      Server context structure.
 * char      *szForm     form to pass to browser.
 * int      iStrLen     length of form excluding null.
 *
 * RETURNS:
 *
 * None
 *

```

```

 * COMMENTS:
 */

void
SendResponse(request_rec *req, char *szForm, int iStrLen)
{
    int    lpbSize, numpad;
    char   szHeader1[10];
    char   headerpad[5];

    lpbSize = iStrLen;

    #if (DEBUG == 1)
        fprintf(MyLogFile, "Entering SendResponse\n");
        fflush(MyLogFile);
    #endif

    sprintf(szHeader1, "%d\0", lpbSize);
    apr_table_setn(req->headers_out, "Keep-Alive", "1");
    /*
    apr_table_setn(req->headers_out, "Content-Length", szHeader1);
    */

    numpad=MAXPAD-(strlen(szHeader1));

    #if (DEBUG == 1)
        fprintf(MyLogFile, "Header Pad = %s\n", szHeader1);
        fprintf(MyLogFile, "numpad = %d\n", numpad);
        fflush(MyLogFile);
    #endif

    if (numpad > 0)
    {
        sprintf(headerpad, "%s\0", "P");
        while (--numpad > 0)
            strcat(headerpad, (char *)"P");
    }

    apr_table_set(req->headers_out, "PRTE PAD", headerpad);
    #if (DEBUG == 1)
        fprintf(MyLogFile, "Header Pad = %s\n", headerpad);
        fflush(MyLogFile);
    #endif

    req->content_type = "text/html";
    /*
    apr_send_http_header(req);
    */

```

```

    ap_rputs(szForm, req);
}

/* 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( request_rec *req,
 *      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: request_rec *req 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( request_rec *req, 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(req, deliveryForm, giFormLen[DELIVERY_FORM]);

    UNRESERVE_FORM( DELIVERY_FORM, deliveryForm );
}

```

```

}

/* FUNCTION: void SendNewOrderForm( request_rec *req,
 *      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: request_rec *req 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( request_rec *req, 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(req, newOrderForm, giFormLen[NEW_ORDER_FORM]);

    UNRESERVE_FORM( NEW_ORDER_FORM, newOrderForm );
}

/* FUNCTION: void SendPaymentForm(request_rec *req,
 *      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:
 *      request_rec *req pointer to structure passed in
 *      the internet
 *      int w_id warehouse id
 *      int ld_id login district id

```

```

*
* RETURNS: None
*
* COMMENTS: None
*
*/

void
SendPaymentForm( request_rec *req, 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(req, paymentForm, giFormLen[PAYMENT_FORM]);

    UNRESERVE_FORM( PAYMENT_FORM, paymentForm );
}

/* FUNCTION: void SendOrderStatusForm(request_rec *req,
*     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: request_rec *req ptr to structure passed in the
*     internet.
*     int w_id warehouse id
*     int ld_id login district id
*
* RETURNS: None
*
* COMMENTS: None
*
*/

void
SendOrderStatusForm( request_rec *req, 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(req, orderStatusForm, giFormLen[ORDER_STATUS_FORM]);

    UNRESERVE_FORM( ORDER_STATUS_FORM, orderStatusForm );
}

/* FUNCTION: void SendStockLevelForm(request_rec *req,
*     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: request_rec *req 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( request_rec *req, 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(req, stockLevelForm, giFormLen[STOCK_LEVEL_FORM]);

UNRESERVE_FORM( STOCK_LEVEL_FORM, stockLevelForm );
}

/* FUNCTION: void SendMainMenuForm(request_rec *req,
 *      int w_id, int ld_id, char *szStatus)
 *
 * PURPOSE: This function sends the main menu form to the browser.
 *
 * ARGUMENTS: request_rec *req 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( request_rec *req,
      int w_id, int ld_id, char *szStatus )
{
    char *szForm;
    int iStrLen;
    static char *szNoStatus = "";
    char *pszStatus;

    pszStatus = ( NULL == szStatus ) ? szNoStatus : szStatus;

    #if (DEBUG == 1)
        fprintf(MyLogFile, "Before RESERVE_PANIC_FORM\n");
        fflush(MyLogFile);
    #endif

    RESERVE_PANIC_FORM( szForm );

    #if (DEBUG == 1)
        fprintf(MyLogFile, "Before SendMainMenuForm\n");
        fflush(MyLogFile);
    #endif

    iStrLen = sprintf( szForm, szMainMenuFormTemplate,
        req->uri, WDID(w_id,ld_id), pszStatus );

    SendResponse(req, szForm, iStrLen);

```

```

UNRESERVE_PANIC_FORM( szForm );
}

/* FUNCTION: void SendWelcomeForm(request_rec *req)
 *
 * PURPOSE: This function sends the welcome form to the browser.
 *
 * ARGUMENTS: None
 *
 * RETURNS: None
 *
 * COMMENTS: The welcome form is generated on initialization.
 */

void
SendWelcomeForm(request_rec *req)
{
    char *mod_name;

    #if (DEBUG == 1)
        fprintf(MyLogFile, "SendWelcomeForm 1\n");
        fflush(MyLogFile);
    #endif

    mod_name = strrchr( req->uri, '/' );
    if( NULL != mod_name )
        mod_name++;
    else
    {
        fprintf(MyLogFile, "SendWelcomeForm: Null mod_name\n");
        return;
    }

    iWelcomeFormLen = sprintf(szWelcomeForm, szWelcomeFormTemplate,
        mod_name);

    #if (DEBUG == 1)
        fprintf(MyLogFile, "SendWelcomeForm 2\n");
        fflush(MyLogFile);
    #endif

    SendResponse( req, szWelcomeForm, iWelcomeFormLen );
}

/* FUNCTION: int ProcessQueryString(request_rec *req)
 *
 * PURPOSE: This function extracts the relevent information out
 * of the http command passed in from the browser.
 *
 * ARGUMENTS: request_rec *req IIS context structure pointer
 *      unique to this connection.

```

```

*
* RETURNS: int      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.
*/

int
ProcessQueryString(request_rec *req)
{
    static char *beginptr = "Begin";
    char *ptr;
    char *cmdptr;
    int cFormID;
    int w_id;
    int ld_id;
    int status;
    int retcode;

    w_id = 0;
    ld_id = 0;

    #if (DEBUG == 1)
        fprintf(MyLogFile, "Starting QueryString 1\n");
        fprintf(MyLogFile, "&ptr=%x\n", &ptr);
        fflush(MyLogFile);
    #endif
    if ( GetCharKeyValuePtr( req->args, '3', &ptr ) )
    {
        cFormID = *ptr++;
        if ( !GetWDID( ptr, &w_id, &ld_id, &ptr ) )
        #if (DEBUG == 1)
            fprintf(MyLogFile, "Calling SendErrorResponse\n");
            fflush(MyLogFile);
        #endif
        SendErrorResponse( req, ERR_W_ID_INVALID, ERR_TYPE_WEBDLL,
            NULL,
            w_id, ld_id, NULL );
        return TRUE;
    }
    else
        cFormID = '\0';

    /* now figure out what command we have and execute it */
    if ( !GetCharKeyValuePtr( ptr, '0', &cmdptr ) )
    {
        if( req->args == NULL ) {
            cmdptr = beginptr;

```

```

}
    else {
        SendErrorResponse( req, ERR_COMMAND_UNDEFINED,
            ERR_TYPE_WEBDLL,
            NULL, w_id, ld_id, NULL );
        return TRUE;
    }
}

if( '\0' == cFormID && !MATCHES_BEGIN( cmdptr ) ) {
    SendErrorResponse( req, ERR_INVALID_FORM_AND_CMD_NOT_BEGIN,
        ERR_TYPE_WEBDLL, NULL, w_id, ld_id, NULL );
    return TRUE;
}

status = TRUE;
if( MATCHES_PROCESS( cmdptr ) )
{
    #if (DEBUG == 1)
        fprintf(MyLogFile, "Matches Process\n");
        fflush(MyLogFile);
    #endif

    if( 'N' == cFormID )
        retcode = ProcessNewOrderQuery( req, ptr, w_id, ld_id );
    else if( 'P' == cFormID )
        retcode = ProcessPaymentQuery( req, ptr, w_id, ld_id );
    else if( 'D' == cFormID )
        retcode = ProcessDeliveryQuery( req, ptr, w_id, ld_id );
    else if( 'O' == cFormID )
        retcode = ProcessOrderStatusQuery( req, ptr, w_id, ld_id );
    else if( 'S' == cFormID )
        retcode = ProcessStockLevelQuery( req, ptr, w_id, ld_id );
    else {
        SendErrorResponse( req, ERR_INVALID_FORM, ERR_TYPE_WEBDLL,
            NULL,
            w_id, ld_id, NULL );
        return TRUE;
    }

    if( ERR_DB_PENDING == retcode )
        status = TRUE;
    else if( ERR_DB_SUCCESS != retcode ) {
        #if (DEBUG == 1)
            fprintf(MyLogFile, "Here We Are Again!!!\n");
            fflush(MyLogFile);
        #endif
        #if (!apr_table_get(req->headers_out, "PRTE PAD"))
        {
            SendErrorResponse( req, retcode, ERR_TYPE_WEBDLL, NULL,
                w_id, ld_id, NULL );
        }
    }
}

```

```

        return TRUE;
    }
}

else if( MATCHES_BEGIN( cmdptr ))
    BeginCmd( req );
else if( MATCHES_NEWORDER( cmdptr ))
    SendNewOrderForm( req, w_id, ld_id );
else if( MATCHES_PAYMENT( cmdptr ))
    SendPaymentForm( req, w_id, ld_id );
else if( MATCHES_ORDERSTATUS( cmdptr ))
    SendOrderStatusForm( req, w_id, ld_id );
else if( MATCHES_STOCKLEVEL( cmdptr ))
    SendStockLevelForm( req, w_id, ld_id );
else if( MATCHES_DELIVERY( cmdptr ))
    SendDeliveryForm( req, w_id, ld_id );
else if( MATCHES_SUBMIT( cmdptr ))
    SubmitCmd( req, &w_id, &ld_id );
else if( MATCHES_MENU( cmdptr ))
    MenuCmd( req, w_id, ld_id );
else if( MATCHES_EXIT( cmdptr ))
    ExitCmd( req );
else if( MATCHES_CLEAR( cmdptr ))
    ClearCmd( req );
else
    SendErrorResponse( req, 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[] = "*****";
    double dtmp;

    pChar += (iFieldSize - 1);

    dtmp=(dVal*100.0);

    if(0 > dVal)
    {
        bSignFlag = FALSE;
        iInt = abs((int)( dtmp ));
    }
    else
    {
        iInt = (int)( dtmp );
    }
    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
 * compiled into the template. The routine scans all input
 * strings to
 * determine if any contain special characters 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. */
                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( request_rec *req,
 *      int retcode,

```

```

*      DeliveryData *deliveryData )
*
* PURPOSE: This function fills in the values and returns the
* response form to the browser.
*
* ARGUMENTS: request_rec *req
* 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;
    request_rec *req;

    req = pDelivery->pCC;

    if ( ERR_DB_PENDING == retcode )
    {
        return;
    }
    else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
    {
        SendErrorResponse( req, ERR_DELIVERY_NOT_PROCESSED,
                          ERR_TYPE_WEBDLL, NULL,
                          pDelivery->w_id, pDelivery->ld_id,
                          (pConnData)pDelivery );

        return;
    }
    else if ( ERR_DB_SUCCESS != retcode )
    {
        SendErrorResponse( req, 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]);

UNRESERVE_TRANSACTION_STRUCT( DELIVERY_TRANS, pDelivery );

PUT_STRING(NULL, 0, 0, StrStruct[ssCnt]);

PutHTMLStrings(StrStruct, deliveryForm,
               giResponseLen[DELIVERY_RESPONSE],
               &szOutput, &iOutputLen);

SendResponse(req, szOutput, iOutputLen);

UNRESERVE_RESPONSE( DELIVERY_RESPONSE, deliveryForm );

if( szOutput != deliveryForm )
    UNRESERVE_PANIC_FORM( szOutput );
}

/* FUNCTION: void TPCCNewOrderResponse(request_rec *req,
*      int retcode,
*      NewOrderData *newOrderData )
*
* PURPOSE: This function fills in the values and returns the
* response form to the browser.
*
* ARGUMENTS: request_rec *req 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];
request_rec *req;

req = pNewOrder->pCC;

if ( ERR_DB_PENDING == retcode )
{
    return;
}
else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
{
    SendErrorResponse( req, 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( req, 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 committed.";
}
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)].iStartIndex-1],
        szDollar, 1);
        PutFloat2(pNewOrder->o_ol[i].i_price,
            newOrderResponseIndexes[NO_PRICE+(i*8)].iLen,
&newOrderForm[newOrderResponseIndexes[NO_PRICE+(i*8)].iStartIndex]);
memcpy(&newOrderForm[newOrderResponseIndexes[NO_AMT+(i*8)].iStartIndex-1],
        szDollar, 1);
        PutFloat2(pNewOrder->o_ol[i].ol_amount,
            newOrderResponseIndexes[NO_AMT+(i*8)].iLen,
&newOrderForm[newOrderResponseIndexes[NO_AMT+(i*8)].iStartIndex]);
    }

    /* 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].iStartIndex],
                szBlanks, newOrderResponseIndexes[jj].iLen);
            jj++;
        }
        /* blank out the '$' for the blank $values */
        for(mm=0; mm<2; mm++)
        {
            memcpy(&newOrderForm[newOrderResponseIndexes[jj].iStartIndex-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].iStartIndex],
      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 $*/
for(mm=0; mm<6; mm++)
{
memcpy(&newOrderForm[newOrderResponseIndexes[jj].iStartIndex],
      szBlanks, newOrderResponseIndexes[jj].iLen);
jj++;
}
/* blank out the '$' for the blank $values */
for(mm=0; mm<2; mm++)
{
memcpy(&newOrderForm[newOrderResponseIndexes[jj].iStartIndex-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(req, szOutput, iOutputLen);

UNRESERVE_RESPONSE( NEW_ORDER_RESPONSE, newOrderForm );

if( szOutput != newOrderForm )
UNRESERVE_PANIC_FORM( szOutput );
}

/* FUNCTION: void TPCCPaymentResponse(request_rec *req,
* int retcode,
* PaymentData *paymentData)
*
* PURPOSE: This function fills in the values and returns the
* response form to the browser.
*
* ARGUMENTS: request_rec *req 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 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;
request_rec *req;

req = pPayment->pCC;

if ( ERR_DB_PENDING == retcode )
{
return;
}
else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
{
SendErrorResponse( req, 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( req, ERR_PAYMENT_INVALID_CUSTOMER,
ERR_TYPE_WEBDLL, NULL,
pPayment->w_id, pPayment->ld_id,
(pConnData)pPayment );
return;
}
else if ( ERR_DB_SUCCESS != retcode )
{
SendErrorResponse( req, 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].iStartIndex],
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].iStartIndex],
            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(req, 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: request_rec *req 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;
    request_rec *req;

    req = pOrderStatus->pCC;

    if ( ERR_DB_PENDING == retcode )
    {
        return;
    }

    else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
    {
        SendErrorResponse( req, 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( req, ERR_NOSUCH_CUSTOMER,
        ERR_TYPE_WEBDLL, NULL,
        pOrderStatus->w_id, pOrderStatus->ld_id,
        (pConnData)pOrderStatus );
    return;
}
else if ( ERR_DB_SUCCESS != retcode )
{
    SendErrorResponse( req, 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].iStartIndex],
    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)].iStartIndex]);
    PutNumeric(pOrderStatus->s_ol[i].ol_i_id,
        orderStatusResponseIndexes[OS_IID+(i*5)].iLen,
&orderStatusForm[orderStatusResponseIndexes[OS_IID+(i*5)].iStartIndex]);
    PutNumeric(pOrderStatus->s_ol[i].ol_quantity,
        orderStatusResponseIndexes[OS_QTY+(i*5)].iLen,
&orderStatusForm[orderStatusResponseIndexes[OS_QTY+(i*5)].iStartIndex]);
    memcpy(&orderStatusForm[orderStatusResponseIndexes[OS_AMT+(i*5)].iStartIndex-1],
        szDollar, 1);
    PutFloat2(pOrderStatus->s_ol[i].ol_amount,
        orderStatusResponseIndexes[OS_AMT+(i*5)].iLen,
&orderStatusForm[orderStatusResponseIndexes[OS_AMT+(i*5)].iStartIndex]);
    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(req, 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: request_rec *req 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;
    request_rec *req;

    req = pStockLevel->pCC;

    if ( ERR_DB_PENDING == retcode )
    {
        return;
    }
    else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
    {
        SendErrorResponse( req, ERR_STOCKLEVEL_NOT_PROCESSED,
            ERR_TYPE_WEBDLL, NULL,
            pStockLevel->w_id, pStockLevel->ld_id,
            (pConnData)pStockLevel );
        return;
    }
    else if ( ERR_DB_SUCCESS != retcode )
    {
        SendErrorResponse( req, 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(req, stockLevelForm,
             giResponseLen[STOCK_LEVEL_RESPONSE]);

UNRESERVE_RESPONSE( STOCK_LEVEL_RESPONSE, stockLevelForm );
}

/* FUNCTION: int ProcessDeliveryQuery( request_rec *req,
 *
 * 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: request_rec *req ptr to the structure
 * containing the internet server
 * information.
 *
 * RETURNS: int status
 *
 * COMMENTS: None
 *
 */

int
ProcessDeliveryQuery( request_rec *req, char *the_request,
                    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 = req;

PARSE_QUERY_STRING(the_request, MAXDELIVERYVALS,
                  deliveryStrs, deliveryVals);

if ( !GetValuePtr(deliveryVals, QUEUE_TIME, &ptr) )
return ERR_DELIVERY_MISSING_QUEUE_TIME_KEY;

if ( !GetNumeric(ptr, &pDelivery->queue_time) )
return ERR_DELIVERY_QUEUE_TIME_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 );

#ifdef FFE_DEBUG
_ASSERT(VALID_DB_ERR(retcode));
pDelivery->iStage |= CALLING_RESP;
#endif

TPCCDeliveryResponse( retcode, pDelivery, CompletedDeliveries );

return retcode;
}

/* FUNCTION: int ProcessNewOrderQuery( request_rec *req,
 *
 * 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: request_rec *req ptr to structure containing
 * internet server info
 *
 * RETURNS: int status

```

```

*
* COMMENTS: None
*
*/

int
ProcessNewOrderQuery( request_rec *req, char *the_request,
                    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 = req;

    if ( ERR_SUCCESS != ( retcode = ParseNewOrderQuery( the_request,
                                                         pNewOrder )))
        return retcode;

#ifdef FFE_DEBUG
    pNewOrder->iStage |= CALLING_LH;
#endif

    retcode = TPCCNewOrder( pNewOrder );

    if (pNewOrder->status > 0)
    {
        retcode=pNewOrder->status;
    }

#ifdef FFE_DEBUG
    _ASSERT(VALID_DB_ERR(retcode));
    pNewOrder->iStage |= CALLING_RESP;
#endif

    TPCCNewOrderResponse( retcode, pNewOrder );

    return retcode;
}

/* FUNCTION: int ProcessOrderStatusQuery( request_rec *req,
*
* 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: request_rec *req ptr to structure that contains
* the internet server info.
*

```

```

* RETURNS: int status
*
* COMMENTS: None
*
*/

int
ProcessOrderStatusQuery( request_rec *req, char *the_request,
                        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 = req;

    if( ERR_SUCCESS != ( retcode = ParseOrderStatusQuery(
the_request,
                                                         pOrderStatus )))
        return retcode;

#ifdef FFE_DEBUG
    pOrderStatus->iStage |= CALLING_LH;
#endif

    retcode = TPCCOrderStatus( pOrderStatus );

    if (pOrderStatus->status > 0)
        retcode=ERR_DB_ERROR;

#ifdef FFE_DEBUG
    _ASSERT(VALID_DB_ERR(retcode));
    pOrderStatus->iStage |= CALLING_RESP;
#endif

    TPCCOrderStatusResponse( retcode, pOrderStatus );

    return retcode;
}

/* FUNCTION: int ProcessPaymentQuery( request_rec *req,
*
* 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: request_rec *req ptr to structure that contains
* the internet server info.
*

```

```

*
* RETURNS: int status
*
* COMMENTS: None
*
*/

int
ProcessPaymentQuery( request_rec *req, char *the_request,
                    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 = req;

    if( ERR_SUCCESS != ( retcode = ParsePaymentQuery( the_request,
                                                    pPayment )) )
        return retcode;

#ifdef FFE_DEBUG
    pPayment->iStage |= CALLING_LH;
#endif

    retcode = TPCCPayment( pPayment );

    if (pPayment->status > 0)
        retcode=ERR_DB_ERROR;

#ifdef FFE_DEBUG
    _ASSERT(VALID_DB_ERR(retcode));
    pPayment->iStage |= CALLING_RESP;
#endif

    TPCCPaymentResponse( retcode, pPayment );

    return retcode;
}

/* FUNCTION: int ProcessStockLevelQuery( request_rec *req,
*
* 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: request_rec *req ptr to structure that contains

```

```

* the internet server info.
* int iSyncId client browser sync id
*
* RETURNS: int status
*
* COMMENTS: None
*
*/

int
ProcessStockLevelQuery( request_rec *req, char *the_request,
                      int w_id, int ld_id )
{
    char *ptr;
    int retcode;
    char *stockLevelVals[MAXSTOCKLEVELVALS];
    StockLevelData *pStockLevel;

#ifdef DEBUG == 1
    fprintf(MyLogFile, "Entering ProcessStockLevelQuery\n");
    fflush(MyLogFile);
#endif

    RESERVE_TRANSACTION_STRUCT( STOCK_LEVEL_TRANS, pStockLevel );

    pStockLevel->w_id = w_id;
    pStockLevel->ld_id = ld_id;
    pStockLevel->pCC = req;

    PARSE_QUERY_STRING(the_request, MAXSTOCKLEVELVALS,
                      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 );

    if (pStockLevel->status > 0)
        retcode=ERR_DB_ERROR;

```



```

#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( int 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( int dwResponseSize, apr_pool_t *p )
{
    int iMallocSize;
    char *pForm;
    int 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));

    #if (DEBUG == 1)

```

```

        fprintf(MyLogFile, "gpPanicForms malloc=%d\n",
iMallocSize);
        fflush(MyLogFile);
    #endif

    gpPanicForms = malloc( iMallocSize );
    apr_thread_mutex_create( &gpPanicForms->critSec, 0, p );
    #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 )
{
    free( gpPanicForms );
}

/* FUNCTION: void MakeTemplatePool( int dwFormSize, int
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( int dwFormSize, int dwResponseSize, apr_pool_t
*p)
{
    char szDeliveryForm[sizeof(szFormTemplate)+FILENAME_SIZE+
        sizeof(szDeliveryFormTemp2i)];
    char szNewOrderForm[sizeof(szFormTemplate)+FILENAME_SIZE+
        sizeof(szNewOrderFormTemp2i)];
    char szOrderStatusForm[sizeof(szFormTemplate)+FILENAME_SIZE+
        sizeof(szOrderStatusFormTemp2i)];
    char szPaymentForm[sizeof(szFormTemplate)+FILENAME_SIZE+
        sizeof(szPaymentFormTemp2i)];
    char szStockLevelForm[sizeof(szFormTemplate)+FILENAME_SIZE+
        sizeof(szStockLevelFormTemp2i)];
    char szDeliveryResponse[sizeof(szFormTemplate)+FILENAME_SIZE+
        sizeof(szDeliveryFormTemp2p)];
    char szNewOrderResponse[sizeof(szFormTemplate)+FILENAME_SIZE+
        sizeof(szNewOrderFormTemp2p)];
    char szOrderStatusResponse[sizeof(szFormTemplate)+FILENAME_SIZE+
        sizeof(szOrderStatusFormTemp2p)];
    char szPaymentResponse[sizeof(szFormTemplate)+FILENAME_SIZE+
        sizeof(szPaymentFormTemp2p)];
    char szStockLevelResponse[sizeof(szFormTemplate)+FILENAME_SIZE+
        sizeof(szStockLevelFormTemp2p)];
    int iFormLen[NUMBER_POOL_FORM_TYPES];
    int iResponseLen[NUMBER_POOL_RESPONSE_TYPES];
    int iMallocSize;
    int iRowSize;
    int 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])));
    #if (DEBUG == 1)
        fprintf(MyLogFile, "gpForms malloc=%d\n", iMallocSize);
        fflush(MyLogFile);
    #endif
    gpForms = malloc( iMallocSize );

    for( jj = 0; jj < NUMBER_POOL_FORM_TYPES; jj++ )
    {
        apr_thread_mutex_create( &gpForms->critSec[jj], 0, p );
        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]));
#if (DEBUG == 1)
    fprintf(MyLogFile, "gpResponses malloc=%d\n", iMallocSize);
    fflush(MyLogFile);
#endif
gpResponses = malloc( iMallocSize );

for( jj = 0; jj < NUMBER_POOL_RESPONSE_TYPES; jj++ )
{
    apr_thread_mutex_create( &gpResponses->critSec[jj], 0, p );
#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 )
{
    free( gpResponses );

    free( gpForms );

    free( gpPanicForms );
}

```

```

}

/* FUNCTION: void MakeTransactionPool( int 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( int dwTransactionPoolSize , apr_pool_t *p)
{
    int iMaxSize;
    int iSize;
    char *data;
    int 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));

    #if 1
        iSize = (((char *)&gpTransactionPool->index - (char
        *)&gpTransactionPool) +
            ((char *)&gpTransactionPool->data - (char *)&gpTransactionPool-
            >index)
            * dwTransactionPoolSize ) +
            (sizeof( char ) * iMaxSize * dwTransactionPoolSize );
    #else
        iSize = (((char *)&gpTransactionPool->index - (char
        *)&gpTransactionPool) +
            ((char *)&gpTransactionPool->data - (char *)&gpTransactionPool-
            >index)
            * dwTransactionPoolSize ) +
            (sizeof( char ) * iMaxSize * dwTransactionPoolSize );
    #endif

    #if (DEBUG == 1)
        fprintf(MyLogFile, "gpTransaction malloc=%d\n", iSize);
        fflush(MyLogFile);
    #endif

    gpTransactionPool = malloc( iSize );

```

```

    apr_thread_mutex_create( &gpTransactionPool->critSec, 0, p );
#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 )
{
    free( gpTransactionPool );
}

/* FUNCTION: void BeginCmd( request_rec *req )
 *
 * PURPOSE: This routine is executed in response to the browser
 * query
 *
 * 'CMD=Begin&Server=??????.
 *
 * ARGUMENTS: request_rec *req IIS context structure pointer
 *
 * unique to this connection.
 *
 * at login.
 *
 * RETURNS: None
 *
 * COMMENTS: Specification of a server machine is required.

```

```

*/

void
BeginCmd( request_rec *req )
{
    SendWelcomeForm(req);
}

/* FUNCTION: void ClearCmd(request_rec *req)
*
* PURPOSE: This resets all terminals and resets the log file.
*
* ARGUMENTS: request_rec *req 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(request_rec *req)
{
    if ( bLog )
    {
        TPCCCloseLog( );
        TPCCOpenLog( req->server->process->pool);
    }

    SendWelcomeForm(req);
}

/* FUNCTION: void ExitCmd(request_rec *req,
*
* PURPOSE: This function deallocates the terminal associated with
*           the browser and presents the login screen.
*
* ARGUMENTS: request_rec *req IIS context structure pointer
*             unique to this connection.
*
* RETURNS: None
*
* COMMENTS: None
*
*/

void
ExitCmd( request_rec *req )

```

```

{
/*
    TPCCDisconnect( req );
*/

    SendWelcomeForm( req );
}

/* FUNCTION: void MenuCmd( request_rec *req,
*
* PURPOSE: This function displays the main menu.
*
* ARGUMENTS: request_rec *req IIS context structure pointer
*             unique to this connection.
*
* RETURNS: None
*
* COMMENTS: None
*/

void
MenuCmd( request_rec *req, int w_id, int ld_id )
{
    SendMainMenuForm(req, w_id, ld_id, NULL);
}

/* FUNCTION: void SubmitCmd( request_rec *req )
*
* PURPOSE: This function assigns a unique terminal id to the
*           calling
*           browser.
*
* ARGUMENTS: request_rec *req 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( request_rec *req, int *w_id, int *ld_id )
{
    int iStatus;
    LoginData login;
    char *ptr;

```

```

if ( !GetCharKeyValuePtr( req->args, '4', &ptr ) ||
    ( 0 == ( *w_id = atoi( ptr ) ) ||
      ( *w_id < 0 ) )
{
    SendErrorResponse( req, ERR_W_ID_INVALID, ERR_TYPE_WEBDLL,
                      NULL, *w_id, -1, NULL );
    goto SubmitError;
}

if ( !GetCharKeyValuePtr( req->args, '5', &ptr ) ||
    ( 0 == ( *ld_id = atoi( ptr ) ) ||
      ( *ld_id > 10 ) ||
      ( *ld_id < 0 ) )
{
    SendErrorResponse( req, 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 = req;
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( req, iStatus, ERR_TYPE_WEBDLL,
                      NULL, *w_id, *ld_id, NULL );
    goto SubmitError;
}

SendMainMenuForm(req, *w_id, *ld_id, NULL);
return;

SubmitError:
return;
}

/* 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 cKey pointer to key to find.
* char **pszOPtr pointer to value.
*
* RETURNS: BOOL FALSE if key is not found.
* TRUE if key is found.
*

```

```

* 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;

    if (szIPtr == NULL)
        return 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 = (int) (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)
    {
        _ASSERT(!_CRTCHECKMEMORY());
        Sleep(1000);
    }

    return 0;
}

#endif

*****
mod_tpcc.h
*****

#ifndef MOD_TPCC_H
#define MOD_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
*
*
*
* Modification history:
*
*
*
* 08/01/2002 Andrew Bond, HP
*
* - Conversion to run under Linux and Apache
*
*/

/* 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 QUEUE TIME 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 MOD_TPCC_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 /* MOD_TPCC_C */
GLOBAL(char szModName[FILENAME_SIZE], { 0 });
#endif /* MOD_TPCC_H */

*****
mod_tpcc_template.c
*****

/*
** mod_tpcc.c -- Apache sample tpcc module
** [Autogenerated via `apxs -n tpcc -g`]
**
** To play with this sample module, first compile it into a
** DSO file and install it into Apache's libexec directory
** by running:
**
**   $ apxs -c -i mod_tpcc.c
**
** Then activate it in Apache's httpd.conf file, for instance
** for the URL /tpcc, as follows:
**
**   # httpd.conf
**   LoadModule tpcc_module libexec/mod_tpcc.so
**   <Location /tpcc>
**     SetHandler tpcc
**   </Location>
**
** Then after restarting Apache via
**
**   $ apachectl restart
**
** you immediately can request the URL /%NAME and watch for the
** output of this module. This can be achieved for instance via:
**
**   $ lynx -mime_header http://localhost/tpcc

```

```

**
** The output should be similar to the following one:
**
** HTTP/1.1 200 OK
** Date: Tue, 31 Mar 1998 14:42:22 GMT
** Server: Apache/1.3.4 (Unix)
** Connection: close
** Content-Type: text/html
**
** The sample page from mod_tpcc.c
*/

#include "httpd.h"
#include "http_config.h"
#include "http_protocol.h"
#include "ap_config.h"

/* The sample content handler */
static int tpcc_handler(request_rec *r)
{
    r->content_type = "text/html";
    ap_send_http_header(r);
    if (!r->header_only)
        ap_rputs("The sample page from mod_tpcc.c\n", r);
    return OK;
}

/* Dispatch list of content handlers */
static const handler_rec tpcc_handlers[] = {
    { "tpcc", tpcc_handler },
    { NULL, NULL }
};

/* Dispatch list for API hooks */
module MODULE_VAR_EXPORT tpcc_module = {
    STANDARD_MODULE_STUFF,
    NULL, /* module initializer
*/
    NULL, /* create per-dir config structures
*/
    NULL, /* merge per-dir config structures
*/
    NULL, /* create per-server config structures
*/
    NULL, /* merge per-server config structures
*/
    NULL, /* table of config file commands
*/
    tpcc_handlers, /* [#8] MIME-typed-dispatched handlers */
    NULL, /* [#1] URI to filename translation
*/
    NULL, /* [#4] validate user id from request
*/
    NULL, /* [#5] check if the user is ok_here_
*/
    NULL, /* [#3] check access by host address
*/
    NULL, /* [#6] determine MIME type
*/
    NULL, /* [#7] pre-run fixups
*/
    NULL, /* [#9] log a transaction
*/
    NULL, /* [#2] header parser
*/
    NULL, /* child_init
*/
    NULL, /* child_exit
*/
    NULL /* [#0] post read-request
*/
#ifdef EAPI
    ,NULL, /* EAPI: add_module
*/
    NULL, /* EAPI: remove_module
*/
    NULL, /* EAPI: rewrite_command
*/
    NULL /* EAPI: new_connection
*/
#endif
};

*****
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 <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/timeb.h>
#include <asm/atomic.h>
#include <linux/spinlock.h>

#include <oci.h>

```

```

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

#define ORACLE_DB_C

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

#define DEADLOCKRETRIES 6

static int bTpcExit; /* exit delivery disconnect loop as dll
exiting. */

static spinlock_t 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
TPCCerr("tkvuopen() fnam: %s, mode: %s\n", fnam, mode);
#endif

fd = fopen((char *)fnam, (char *)mode);
if (!fd){
TPCCerr(" fopen on %s failed %d\n",fnam,fd);
/* exit(-1); */
}
return(fd);
}

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

#ifdef DEBUG

```

```

TPCCerr("sqlfile() fname: %s, linebuf: %#x\n", fname, linebuf);
#endif

fd = fopen(fname, "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()
{
#ifdef DEBUG_TPCCSTARTUPDB
_ASSERT(FALSE);
#endif

spin_lock_init(&ErrorLogCriticalSection);

return ERR_DB_SUCCESS;
}

int TPCCShutdownDB(void)
{
bTpccExit = TRUE;

/* Add Oracle specific code */

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 RECOVER;
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:
TPCCerr("Warning Deadlock, being retried");
return RECOVER;

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

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

```

```

/* vmm313   TPCCDisconnectDB(p); */
/* vmm313   exit(1); */
    break;
case OCI_INVALID_HANDLE:
    sprintf(errbuf, "Module %s Line %d\r\n", fname, lineno);
    strcat(errbuf, "Error - OCI_INVALID_HANDLE\r\n");
    TPCCErr("%s", errbuf);
    TPCCDisconnectDB(p, NULL);
    return IRRECERR;
/*   terminate(-1); */
/*   exit(-1); */
    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;
}
TPCCErr("%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);

OCIEnvCreate(&(p->tpcenv), OCI_DEFAULT | OCI_OBJECT, NULL, NULL,
NULL, NULL, (size_t) 0, NULL);

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, OCI_SERVER_ATTACH(p->tpcsrv, p->errhp,
(text *)0, 0, OCI_DEFAULT))))
/* return IRRECERR; */

```



```

return ERR_DB_ERROR;

/*
OCIERROR(p, OCI_SERVER_ATTACH(p->tpcsrv, p->errhp,
userstr, strlen(userstr),
OCI_DEFAULT));*/

/*
{
return IRRECERR;
}
*/
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 (RECOVER != OCIERROR(p, OCI_SESSION_BEGIN(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 (RECOVER != OCIERROR(p, OCI_STMT_EXECUTE(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 (RECOVER != OCIERROR(p, OCI_STMT_EXECUTE(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 (tkvodinit (&(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->ptx));
        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) {
        tkvcdone(&(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_SUCCESS 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 = RECOVER;
        tries < DEADLOCKRETRIES && status == RECOVER; tries++) {

        status = tkvcs(dbproc);
    }

    pStockLevel->low_stock = dbproc-
>bindvars.info.stockLevel.low_stock;

    if (status == RECOVER) 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(&ntemp->cr_date); */
    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 = the size of entry_date in newtemp struct */
datebufsize=21;
/* datebufsize=sizeof(ntemp->entry_date); */
/* OCIDateToText(datecvterrhp, &ntemp->cr_date,(text *) "DD-MM-
YYYY HH:MM:SS", 19, (text *) 0, 0, &datebufsize, &ntemp-
>entry_date); */
/* cvtdmyhms(ntemp->cr_date, ntemp->entry_date); */
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;
    float ftmp;
    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. */
    ftmp=pbindvars->h_amount*100;
    ptemp->h_amount = (int)(ftmp);

    for ( tries = 0, status = RECOVER;
    tries < DEADLOCKRETRIES && status == RECOVER; 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 = the size of c_since_str in paytemp struct */
    datebufsize=11;

    /* convert date format */

    /* OCIDateToText(datecvterrhp, &ptemp->customer_sdate,(text *) 0,
10, (text *) 0, 0, &datebufsize, &ptemp->c_since_str); */

    OCIDateToText(datecvterrhp, &ptemp->customer_sdate,(text *) "DD-
MM-YYYY", 10, (text *) 0, 0, &datebufsize, &ptemp->c_since_str);

    /* cvtdmy(ptemp->customer_sdate, ptemp->c_since_str); */

    /* datebufsize = the size of h_date string in paytemp struct */
    datebufsize=DATE_SIZ;

    /* OCIDateToText(datecvterrhp, &ptemp->cr_date,(text *) "DD-MM-
YYYY.HH24:MI:SS", 21, (text *) 0, 0, &datebufsize, &ptemp->h_date);
*/

    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 == RECOVER) return ERR_DB_DEADLOCK_LIMIT;
    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 successful
*
* COMMENTS: None
*
*/

int TPCCOrderStatusDB(OraContext *dbproc, pOrderStatusData
pOrderStatus)
{

int tries,status;
int ii;

OrderStatusData *pbindvars = &(dbproc-
>bindvars.info.orderStatus);

ordtemp *otemp = &(dbproc->tempvars.ord);
OCIErr *datecvterrhp = dbproc->datecvterrhp;

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

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

    status = tkvcd(pDeliveryData, dbproc);
}

if(status == RECOVER) 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 checkpoint 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 (RECOVER != OCIERROR(dbproc,
OCIStmtExecute(dbproc->tpcenv, dbproc->curi,
dbproc->errhp, 1, 0, 0, 0,
OCI_DEFAULT)))
return (ERR_DB_ERROR);

OCIHandleFree(dbproc->curi, OCI_HTYPE_STMT);

return ERR_DB_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.
-*/

/*+
-*/

/*+
-*/

/*+
-*/

*****_*/

/*
*
*
* Modification history:
*
*
* 08/01/2002 Andrew Bond, HP
* Conversion to run under Linux and Apache
*
*/

#ifndef ORACLE_DB_H
#define ORACLE_DB_H

#endif 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 */

```

```

#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];
    /* float amt[NDISTS]; Changed to int */
    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

    OCISmt *curd0;
    OCISmt *curd1;
    OCISmt *curd2;
    OCISmt *curd3;
    OCISmt *curd4;
    OCISmt *curd5;
    OCISmt *curd6;
    OCISmt *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];

/* changed to OCIDate unsigned char null_date[NITEMS][7]; */
OCIDate null_date[NITEMS]; /* base date for null date entry */
/* not needed for Oracle 8.1.5 OCISmt *curn; */
OCISmt *curn1;
OCISmt *curn2;
OCISmt *curn3[10];
OCISmt *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 {
    OCISstmt *curpi;
    OCISstmt *curp0;
    OCISstmt *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_bpl;
sb2 w_street_1_ind;
ub2 w_street_1_len;
ub2 w_street_1_rc;

OCIBind *w_street_2_bp;
OCIBind *w_street_2_bpl;
sb2 w_street_2_ind;
ub2 w_street_2_len;
ub2 w_street_2_rc;

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

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

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

OCIBind *d_street_1_bp;
OCIBind *d_street_1_bpl;
sb2 d_street_1_ind;
ub2 d_street_1_len;
ub2 d_street_1_rc;

OCIBind *d_street_2_bp;
OCIBind *d_street_2_bpl;
sb2 d_street_2_ind;
ub2 d_street_2_len;
ub2 d_street_2_rc;

OCIBind *d_city_bp;
OCIBind *d_city_bpl;
sb2 d_city_ind;
ub2 d_city_len;
ub2 d_city_rc;
```

```
OCIBind *d_state_bp;
OCIBind *d_state_bpl;
sb2 d_state_ind;
ub2 d_state_len;
ub2 d_state_rc;

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

OCIBind *c_first_bp;
OCIBind *c_first_bpl;
sb2 c_first_ind;
ub2 c_first_len;
ub2 c_first_rc;

OCIBind *c_middle_bp;
OCIBind *c_middle_bpl;
sb2 c_middle_ind;
ub2 c_middle_len;
ub2 c_middle_rc;

OCIBind *c_street_1_bp;
OCIBind *c_street_1_bpl;
sb2 c_street_1_ind;
ub2 c_street_1_len;
ub2 c_street_1_rc;

OCIBind *c_street_2_bp;
OCIBind *c_street_2_bpl;
sb2 c_street_2_ind;
ub2 c_street_2_len;
ub2 c_street_2_rc;

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

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

OCIBind *c_zip_bp;
OCIBind *c_zip_bpl;
```

```

sb2 c_zip_ind;
ub2 c_zip_len;
ub2 c_zip_rc;

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

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

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

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

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

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

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

OCIBind *h_date_bp;
OCIBind *h_date_bpl;
sb2 h_date_ind;
ub2 h_date_len;

```

```

ub2 h_date_rc;

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

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

/* temporary structures needed since oracle binds to some vars
differently
    than we store in our tpcc structures from tpccstruct.h */

typedef struct _deltemp {
    char cvtor_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];

```

```

double c_discount;
double w_tax;
double 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;
    double c_discount;
    int h_amount;
    int c_credit_lim;
    int p_retry;
} paytemp;

typedef struct _oracontext {
    /* V8 handles for talking to Oracle */
    OCIEnv *tpcenv;
    OCIError *errhp;
    OCIError *datecvterrhp;
    OCISvcCtx *tpcsvc;
    OCISession *tpcusr;
    OCISmt *curi;
    /* other V8 additions */
    dvoid *xmem;
    /* are these really needed since we do not malloc and therefore
    do not
        need to free in *txn*done ???*/
    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;
stctx sctx;
defctx cbctx;
amtctx actx;
/* temporary data areas for cursor data - oracle stores/binds
differently than tpcc */
union {
    deltemp del;
    newtemp new;
    ordtemp ord;
    paytemp pay;
} tempvars;
} OraContext;

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

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

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

#define
OCIENDRAD(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
OCIENDR(stmp,bndp,p,sqlvar,progvl,ftype,indp,alen,arcode) \
    ocierror(__FILE__,__LINE__,(p), \

```

```

OCIBindByName((stmp), &(bndp), (p->errhp), (text
*)(sqlvar), strlen((sqlvar)), \

(prog), (progv), (ftype), (indp), (alen), (arcodes), 0, 0, OCI_DEFAULT))

#define
OCIBNDRAA(stmp, bndp, p, sqlvar, progv, progv1, ftype, indp, alen, arcodes, ms
, cu) \
    ocierror(__FILE__, __LINE__, (p), \
        OCIBindByName((stmp), &(bndp), (p->errhp), \
            (text *) (sqlvar), strlen((sqlvar)), \
            (prog), (progv), (ftype), (indp), (alen), (arcodes), \
            (ms), (cu), OCI_DEFAULT))

#define OCIDEFINE(stmp, dfnp, errp, pos, progv, progv1, ftype) \
OCIDEFINE(stmp, dfnp, errp, pos, progv, progv1, ftype) \
OCIDefineByPos((stmp), &(dfnp), (errp), (pos), (prog), (progv), (ftype)
, \
    0, 0, 0, OCI_DEFAULT)

#define OCIDEF(stmp, dfnp, errp, pos, progv, progv1, ftype) \
OCIDefineByPos((stmp), &(dfnp), (errp), (pos), (prog), (progv), \
    (ftype), NULL, NULL, NULL, OCI_DEFAULT)

#define
OCIDFNRA(stmp, dfnp, errp, pos, progv, progv1, ftype, indp, alen, arcodes) \
OCIDefineByPos((stmp), &(dfnp), (errp), (pos), (prog), \
    (progv), (ftype), (indp), (alen), \
    (arcodes), OCI_DEFAULT)

#define
OCIDFNDRY(stmp, dfnp, errp, pos, progv, progv1, ftype, indp, ctp, 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), (prog),
        (progv1), (ftype), \
            (indp), NULL, NULL,
OCI_DYNAMIC_FETCH)); \
    ocierror(__FILE__, __LINE__, (errp), \
        OCIDefineDynamic((dfnp), (errp), (ctp), (cbf_data)));

/* old defines for v7 */
/*****

#define OBNDRV(lda, cursor, sqlvar, progv, progv1, ftype) \

```

```

if
(obndrv((cursor), (text*) (sqlvar), NA, (ub1*) (prog), (progv), (ftype),
NA, \
    (sb2 *) 0, (text *) 0, NA, NA)) \
    {ErrRpt(lda, cursor->rc); return( ERR_DB_ERROR); } \
else \
    DISCARD 0

#define
OBNDRA(lda, cursor, sqlvar, progv, progv1, ftype, indp, alen, arcodes) \
if
(obndra((cursor), (text*) (sqlvar), NA, (ub1*) (prog), (progv), (ftype),
NA, \
    (indp), (alen), (arcodes), (ub4) 0, (ub4*) 0, (text*) 0, NA, NA)) \
    {ErrRpt(lda, cursor->rc); return( ERR_DB_ERROR); } \
else \
    DISCARD 0

#define
OBNDRAA(lda, cursor, sqlvar, progv, progv1, ftype, indp, alen, arcodes, ms, cs
) \
if
(obndraa((cursor), (text*) (sqlvar), NA, (ub1*) (prog), (progv), (ftype),
NA, \
    (indp), (alen), (arcodes), (ub4) (ms), (ub4*) (cs), (text*) 0, NA, NA)) \
    {ErrRpt(lda, cursor->rc); return( ERR_DB_ERROR); } \
else \
    DISCARD 0

#define
ODEFIN(lda, cursor, pos, buf, bufl, ftype, scale, indp, fmt, fmt1, fmtt, rlen,
rcode) \
if
(odefin((cursor), (pos), (ub1*) (buf), (bufl), (ftype), (scale), (indp), \
    (text*) (fmt), (fmt1), (fmtt), (rlen), (rcode))) \
    {ErrRpt(lda, cursor->rc); return( ERR_DB_ERROR); } \
else \
    DISCARD 0

#define
OEXFET(lda, cursor, nrows, cancel, exact) \
if (oexfet((cursor), (nrows), (cancel), (exact))) \
    {if ((cursor->rc) == 1403) DISCARD 0; \
    else if (ErrRpt(lda, cursor->rc) == RECOVER) \
        {orol(lda); return(RECOVER); } \
    else {orol(lda); return( ERR_DB_ERROR); }} \
else \
    DISCARD 0

#define
OOPEN(lda, cursor) \
if (oopen((cursor), (lda), (text*) 0, NA, NA, (text*) 0, NA)) \
    {ErrRpt(lda, cursor->rc); return( ERR_DB_ERROR); } \
else \
    DISCARD 0

#define
OPARSE(lda, cursor, sqlstm, sql1, defflg, lngflg) \

```



```

if
(oparse((cursor),(sqlstm),(sb4)(sql1),(defflg),(ub4)(lngflg)))\
    {ErrRpt(lda,cursor->rc);return(ERR_DB_ERROR);}\
else\
    DISCARD 0

#define OFEN(lda,cursor,nrows)\
if (ofen((cursor),(nrows))\
    {if (ErrRpt(lda,cursor->rc)==RECOVERR) \
        {orol(lda);return(RECOVERR);} \
    else{orol(lda);return(ERR_DB_ERROR);}}\
else\
    DISCARD 0

#define OEXEC(lda,cursor)\
if (oexec((cursor))\
    {if (ErrRpt(lda,cursor->rc)==RECOVERR) \
        {orol(lda);return(RECOVERR);} \
    else{orol(lda);return(ERR_DB_ERROR);}}\
else\
    DISCARD 0

#define OCOM(lda,cursor)\
if (ocom((lda)) \
    {ErrRpt(lda,cursor->rc);orol(lda);return(-1);}\
else\
    DISCARD 0

#define OEXN(lda,cursor,itors,rowoff)\
if (oexn((cursor),(itors),(rowoff)) \
    {if (ErrRpt(lda,cursor->rc)==RECOVERR) \
        {orol(lda);return(RECOVERR);} \
    else{orol(lda);return(-1);}}\
else\
    DISCARD 0

*****

/* 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 tkvcoodone (ordctx *pctx);

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

extern int tkvcs (OraContext *p);

extern void tkvcsdone (stocctx *psctx);

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

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

extern void tkvoddone (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
pdel.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 anonymous 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 v8 database - for the tpc benchmark.

```

```

*
*
* Modification history:
*
*
*   08/01/2002   Andrew Bond, HP Corporation
*
*               - Conversion to run under Linux
*
*/

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

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

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

#include <tpcc.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\0",day,month,year); */
    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\0", */
    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;

tempub2 = nctx->i_price_ind[i];
nctx->i_price_ind[i] = nctx->i_price_ind[j];
nctx->i_price_ind[j] = tempub2;
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;

tempub2 = nctx->i_name_ind[i];
nctx->i_name_ind[i] = nctx->i_name_ind[j];
nctx->i_name_ind[j] = tempub2;
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);

tempub2 = nctx->i_data_ind[i];
nctx->i_data_ind[i] = nctx->i_data_ind[j];
nctx->i_data_ind[j] = tempub2;
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);

tempub2 = nctx->s_quantity_ind[i];
nctx->s_quantity_ind[i] = nctx->s_quantity_ind[j];
nctx->s_quantity_ind[j] = tempub2;
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;

tempub2 = nctx->s_dist_info_ind[i];
nctx->s_dist_info_ind[i] = nctx->s_dist_info_ind[j];
nctx->s_dist_info_ind[j] = tempub2;
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);

tempub2 = nctx->s_data_ind[i];
nctx->s_data_ind[i] = nctx->s_data_ind[j];
nctx->s_data_ind[j] = tempub2;

```

```

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 = &(amp;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 = &(amp;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->cur3)[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 == RECOVER))
        {
            /* In case of NO_DATA this should NOT return, but simply fall
            through */
            OCITransRollback(p->tpcsvc, p->errhp, OCI_DEFAULT);
            return (RECOVER);
        }
    }
    else
    {
        OCITransRollback(p->tpcsvc, p->errhp, OCI_DEFAULT);
        return (IRRECERR);
    }
}

/* mark invalid items */
OCIAttrGet((nctx->cur3)[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 more than one item is wrong, this is not an simulated */
/* error and we'll blow off */

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 */
}
/**

```

```

TPCCErr("TPC-C server %d: reordering items and stocks\n",
proc_no);   ***/

/* 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 (zpc3);
}

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 = OCISmtExecute(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 == RECOVER)) {
return (RECOVER);
    } else {
        return (IRRECERR);
    }
}
OCIAAttrGet(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,OCI_DEFAULT);
if(execstatus != OCI_SUCCESS) {
    OCITransRollback(p->tpcsvc,p->errhp,OCI_DEFAULT);
    errcode = OCIERROR(p,execstatus);
    if(errcode == NOT_SERIALIZABLE) {
return (RECOVER);
    } else if (errcode == RECOVER) {
return (RECOVER);
    } 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 SQLXTTEST "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 (RECOVER);
} else if (errcode == RECOVER) {
return (RECOVER);
} else if (errcode == SNAPSHOT_TOO_OLD) {
return (RECOVER);
} else {
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;"

tkvcppinit (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 = 7; */
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->h_date_bp, p,":h_date",pTemp-
>h_date,
    SIZ(pTemp->h_date),
    SQLT_STR, &pctx->h_date_ind, &pctx->h_date_len, &pctx->
>h_date_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->curpl, pctx->w_id_bpl, 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->curpl, pctx->d_id_bpl, 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);
OCIBNDR(pctx->curpl, pctx->c_w_id_bpl, p,":c_w_id",ADR(pPay->
c_w_id),
        SIZ(pPay->c_w_id),
        SQLT_INT);
OCIBNDR(pctx->curpl, pctx->c_d_id_bpl, p,":c_d_id",ADR(pPay->
c_d_id),
        SIZ(pPay->c_d_id),
        SQLT_INT);
OCIBNDR(pctx->curpl, pctx->c_id_bpl, 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->curpl, pctx->h_amount_bpl, p,":h_amount",
        ADR(ptemp->h_amount),
        SIZ(int),SQLT_INT, &pctx->h_amount_ind, &pctx->
h_amount_len,
        &pctx->h_amount_rc);
OCIBNDR(pctx->curpl, pctx->c_last_bpl, p,":c_last",pPay->c_last,
        SIZ(pPay->c_last),
        SQLT_STR);
OCIBNDR(pctx->curpl, pctx->w_street_1_bpl, 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->curpl, pctx->w_street_2_bpl, 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->curpl, pctx->w_city_bpl, 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->curpl, pctx->w_state_bpl, 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->curpl, pctx->w_zip_bpl, 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->curpl, pctx->d_street_1_bpl, 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->curpl, pctx->d_street_2_bpl, 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->curpl, pctx->d_city_bpl, 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->curpl, pctx->d_state_bpl, 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->curpl, pctx->d_zip_bpl, 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->curpl, pctx->c_first_bpl, 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->curpl, pctx->c_middle_bpl, p,":c_middle",
        pPay->c_middle,2,
        SQLT_AFC, &pctx->c_middle_ind, &pctx->c_middle_len,
        &pctx->c_middle_rc);
OCIBNDR(pctx->curpl, pctx->c_street_1_bpl, 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->curpl, pctx->c_street_2_bpl, 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->curpl, pctx->c_city_bpl, 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->curpl, pctx->c_state_bpl, 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->curpl, pctx->c_zip_bpl, 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->curpl, pctx->c_phone_bpl, 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->curpl, pctx->c_since_bpl, 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->curpl, pctx->c_credit_bpl, 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->curpl, pctx->c_credit_lim_bpl, 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->curpl, pctx->c_discount_bpl, 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->curpl, pctx->c_balance_bpl, 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->curpl, pctx->c_data_bpl, 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->curpl, pctx->h_date_bpl, p,":h_date",
        ptemp->h_date,SIZ(ptemp->h_date),
        SQLT_STR, &pctx->h_date_ind, &pctx->h_date_len, &pctx->
>h_date_rc); */
OCIBNDR(pctx->curpl, pctx->retries_bpl, p,":retry",
        ADR(ptemp->p_retry),SIZ(int), SQLT_INT,
        &pctx->retries_ind, &pctx->retries_len, &pctx->retries_rc);
OCIBNDR(pctx->curpl, pctx->cr_date_bpl, 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);

}

```

```
tkvcv (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(ptemp->cr_date); */

```

```

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 = 7; */
pctx->cr_date_len = sizeof(ptime->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(RECOVER);
    } else if (errcode == RECOVER) {
        return(RECOVER);
    } else if (errcode == SNAPSHOT_TOO_OLD) {
        return(RECOVER);
    } else {

```

```

        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 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 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_delivery_d_len[i] = sizeof(pOrd->s_ol-
        >ol_delivery_d); */
    }

    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));

/** sth OCIDFNDDYN( octx->curo0,
    octx->c_rowid_dp,
    p,
    1,
    octx->c_rowid_ptr,
    sizeof(OCIRowid*),
    SQLT_RDD,
    octx->c_rowid_ind,
    p,
    rid_data);
**/

    OCIBND(octx->curo1,octx->c_rowid_bp,p,":cust_rowid",
        &octx->middle_cust,sizeof(octx->middle_cust),SQLT_RDD);

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

    OCIDF(octx->curo1,octx->c_last_dp1,p->errhp,5,pOrd->c_last,
        SIZ(pOrd->c_last)-1, SSQLT_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 v8 */
/* octx->o_ol_cnt_ind = NA;*/
/* pOrd->o_ol_cnt = 0;*/
if(pOrd->byname)
{
    cbctx->reexec = FALSE;
    execstatus=OCIStmtExecute(p->tpcsvc,octx->curo0,p->errhp,
        100,0,0,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 {
            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 {
            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 RECOVER;
    } else {
        return ERR_DB_ERROR;
    }
}
octx->cust_idx=0;
}
}

octx->middle_cust=octx->c_rowid_ptr{octx->cust_idx};
execstatus = OCISmtExecute(p->tpcsvc,octx->curol,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 == RECOVER) ||
        (errcode == SNAPSHOT_TOO_OLD))
    {
        return RECOVER;
    } else {
        return ERR_DB_ERROR;
    }
}
}
else
{
    /* execstatus = OCISmtExecute(p->tpcsvc,octx->curo2,p->errhp,
        1,0,0,0,OCI_DEFAULT); */
    execstatus = OCISmtExecute(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 == RECOVER)
            || (errcode == SNAPSHOT_TOO_OLD))
        {
            return RECOVER;
        } else {
            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 = OCISmtExecute(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 == RECOVER)
            || (errcode == SNAPSHOT_TOO_OLD))
        {
            return RECOVER;
        } else {
            return ERR_DB_ERROR;
        }
    }
}

/*
#ifdef NOTMORE
    OCIERROR(errhp,
        OCITransCommit(tpcsvc,errhp,OCI_DEFAULT));
#endif
*/

    /* 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
            /* cvtdmy(otemp->loc_ol_delivery_date[i], otemp-
>ol_delivery_date_str[i]); */
            {
                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 */
    /* cvtdmyhms(otemp->entry_date, otemp->entry_date_str); */
    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 *pctx)
{
    ordctx octx = *pctx;

    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 */ ord1 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 */

tkvcodinit (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, SQLT4);
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, SQLT6);
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);

#ifdef ISO || defined(ISO5) || defined(ISO6) || defined(ISO8)
    int hasno;
    int rread;
    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], &temp->cr_date, sizeof(OCIDate));

    actx->o1_cnt[i]=0;
}

/* array select from new_order and orders tables */

execstatus=OCISmtExecute(p->tpcsvc, dctx->curdl, 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(RECOVER);
}
else if (errcode == RECOVER)
{
    return(RECOVER);
}
else if (errcode == SNAPSHOT_TOO_OLD)
{
    return(RECOVER);
}
else
{
    return (ERR_DB_ERROR);
}
}

/* mark districts with no new order */
OCIAttrGet(dctx-
>curdl, 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
        {
            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
        {
            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,rc,0,0,0,
                                OCI_DEFAULT);
#else
    execstatus=OCIStmtExecute(p->tpcsvc,dctx->curd6,p-
>errhp,rc,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 (RECOVER);
        }
        else if (errcode == RECOVER)
        {
            return (RECOVER);
        }
        else if (errcode == SNAPSHOT_TOO_OLD)
        {
            return (RECOVER);
        }
        else
        {
            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(:idxlarr); 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_NTIV_SYNTAX, OCI_DEFAULT));

/* bind variables */

OCIBNDR(nctx->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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->curnl, 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_NT_V_SYNTAX, OCI_DEFAULT));

/* execute second cursor to init newinit package */
{
    int idxlarr[NITEMS];
    OCIBind *idxlarr_bp;
    ub2 idxlarr_len[NITEMS];
    ub2 idxlarr_rcode[NITEMS];
    sb2 idxlarr_ind[NITEMS];
    ub4 idxlarr_count;
    ub2 idx;

    for (idx=0;idx<NITEMS;idx++)
    {
        idxlarr[idx] = idx + 1;
        idxlarr_ind[idx] = TRUE;
        idxlarr_len[idx] = sizeof(int);
    }
    idxlarr_count=NITEMS;
    pNew->o_ol_cnt=NITEMS;

/* Bind array */
    OCIENDRAA(nctx-
>curn2,idxlarr_bp,p,":idxlarr",idxlarr,SIZ(int),SQLT_INT,
        idxlarr_ind,idxlarr_len,idxlarr_rcode,NITEMS,&idxlarr_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);
        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 */
/* this section of code is needed for cprime, but we have
    already calculated the number of order lines back in web_ui
    no need to recalculate. */
/* pNew->o_ol_cnt = NITEMS;
pNew->o_all_local = 1;
for (i = 0; i < NITEMS; i++) {
    if (ntemp->nol_i_id[i] == 0) {
        pNew->o_ol_cnt = i;
        break;
    }
}
*/

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;

```

/* following not relevant */

```

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

/* 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);
}

if(execstatus != OCI_SUCCESS) {
    OCITransRollback(p->tpcsvc,p->errhp,OCI_DEFAULT);
    errcode = OCIERROR(p,execstatus);
    if(errcode == NOT_SERIALIZABLE)
    {
        retries++;
        return (RECOVER);
    }
    else if (errcode == RECOVER)
    {
        retries++;
        return (RECOVER);
    }
    else if (errcode == SNAPSHOT_TOO_OLD)
    {
        retries++;
        return (RECOVER);
    }
    else
    {
        return (ERR_DB_ERROR);
    }
}

```

```

#ifdef IS01
    else {
        OCITransCommit(p->tpcsvc, p->errhp, OCI_DEFAULT);
    }
#endif

#if defined(IS01) || defined(IS07)
    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 *= ((double)(1-ntemp->c_discount)) *
(double)(1.0 + ((double)(ntemp->d_tax))+((double)(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.
*
*
*
*
*
*
*****
*****/

/*
*
*
* Modification history:
*
*
*
* 08/01/2002 Andrew Bond, HP
*
* - Conversion to run under Linux and Apache
*
*/

#include <stdio.h>
#include <stdarg.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#include "apr_thread_mutex.h"

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

#define TPCC_C

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

#include <tpcc.h>
#include <mod_tpcc.h>

#define _strupr(x) { \
\
    int strupr_pos; \
\
    for (strupr_pos=0; strupr_pos <
strlen(x);strupr_pos++) \
\
        x[strupr_pos] = toupper(x[strupr_pos]); \
\
}

```

```

/* 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 Linux TPCC configuration file
for
 * startup parameters.
 *
 * 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)
{
    char szTmp[FILENAME_SIZE];
    int status;
    int iTmp;

```

```

status = GetConfigValue("PATH", (char *)&szTmp);
if ( status != ERROR_SUCCESS )
    return ERR_CANT_FIND_PATH_VALUE;
strcpy(szTpccLogPath, szTmp);

status = GetConfigValue("Server", (char *)&szTmp);
if ( status != ERROR_SUCCESS )
    /* required */
    return ERR_CANT_FIND_SERVER_VALUE;
strcpy(gszServer, szTmp);

status = GetConfigValue("Database", (char *)&szTmp);
if ( status != ERROR_SUCCESS )
    /* required */
    return ERR_CANT_FIND_DATABASE_VALUE;
strcpy(gszDatabase, szTmp);

status = GetConfigValue("User", (char *)&szTmp);
if ( status != ERROR_SUCCESS )
    /* required */
    return ERR_CANT_FIND_USER_VALUE;
strcpy(gszUser, szTmp);

status = GetConfigValue("Password", (char *)&szTmp);
if ( status != ERROR_SUCCESS )
    /* required */
    return ERR_CANT_FIND_PASSWORD_VALUE;
strcpy(gszPassword, szTmp);

status = GetConfigValue("LOG", (char *)&szTmp);
if ( status == ERROR_SUCCESS && 0 == strcmp(szTmp, "ON" ) )
    bLog = TRUE;

status = GetConfigValue("MaxConnections", (char *)&szTmp);
if ( status == ERROR_SUCCESS && 0 != (iTmp = atoi(szTmp)) )
    iMaxConnections = iTmp;

return ERR_SUCCESS;
}

*****
tpcc.h
*****

#ifdef 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
*
*
* Modification history:
*
*
* 08/01/2002 Andrew Bond, HP
*
* Conversion to run under Linux and Apache
*
*/

#define ERROR_SUCCESS 1
#define FILENAMESIZE 256

```

```

#define DEBUG 0
#define MAXPAD 6

#define itoa(x,y) sprintf(y, "%d", x)

#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 MOD_TPCC_C
#define GLOBAL(thing,initializer) thing = initializer
#else
#define GLOBAL(thing,initializer) extern thing
#endif /* 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],"oracle");
GLOBAL(char gszPassword[32],{'\0'});
GLOBAL(pTransactionPoolStruct gpTransactionPool,{0});
GLOBAL(FILE *MyLogFile, {0});

#endif /* TPCC_H */

*****
tpccapi.h
*****

#ifdef 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
**
*
* Modification history:
*
*
* 08/01/2002      Andrew Bond, HP
*
*      Conversion to run under Linux and Apache

```

```

*
*/

#define DELIVERY_RESPONSE_COUNT 2

int TPCCGetTransportData( pTransportData pTransport );

int TPCCStartup( );
int TPCCStartupDB( );

int TPCCConnect( pLoginData pLogin );
int TPCCConnectDB( OraContext **dbproc, pLoginData pLogin );

int TPCCDelivery( pDeliveryData pDelivery);
int TPCCDeliveryDeferred( pDeliveryData ppDelivery );
int TPCCDeliveryDB( OraContext *dbproc, pDeliveryData pDelivery );

int TPCCNewOrder( pNewOrderData pNewOrder );
int TPCCNewOrderDB( OraContext *dbproc, pNewOrderData pNewOrder );

int TPCCOrderStatus( pOrderStatusData pOrderStatus );
int TPCCOrderStatusDB( OraContext *dbproc, pOrderStatusData
pOrderStatus );

int TPCCPayment( pPaymentData pPayment );
int TPCCPaymentDB( OraContext *dbproc, pPaymentData pPayment );

int TPCCStockLevel( pStockLevelData pStockLevel );
int TPCCStockLevelDB( OraContext *dbproc, pStockLevelData
pStockLevel );

int TPCCCheckpoint( pCheckpointData pCheckpoint );
int TPCCCheckpointDB( OraContext *dbproc, pCheckpointData
pCheckpoint );

int TPCCDisconnect( pCallersContext pCC );
int TPCCDisconnectDB( OraContext *dbproc, 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 *pszMesage );

int TPCCGetTransportErrorString( int iErrorCode, int iBufSize, char
*pBuffer );
int TPCCGetDBErrorString( int iErrorCode, int iBufSize, char
*pBuffer );

BOOL TPCCOpenLog( apr_pool_t *pool );

BOOL TPCCCloseLog( void );

void TPCCLog( char *fmt, ... );

void TPCCerr( char *fmt, ... );

void TPCCTransactionErr( pConnData pConn, char *fmt, ... );

int GetConfigValue(char *option, char *value);

#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@percom.enet.dec.com
*
* Modification history:
*
*/

```

```

*
*
*/

#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_CID_INVALID 1051
#define ERR_PAYMENT_CID_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_FRCINIT_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_QUEUEUETIME_KEY 1105
#define ERR_DELIVERY_QUEUEUETIME_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 corresponding Item.Id." },
    { ERR_NEWORDER_QTY_WITHOUT_ITEMID, "New Order Qty entered without a corresponding 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_QUEUEUETIME_KEY, "Delivery queue time
missing from query." },

{ ERR_DELIVERY_QUEUEUETIME_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
*****

#ifdef TPCCSTRUCT_H
#define TPCCSTRUCT_H

#include "apr_thread_mutex.h"

/*****
*****
***** tpcstruct.h
*****
*****/

/*
** tpcstruct.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)
**

```



```

*
* Modification history:
*
*
* 08/01/2002 Andrew Bond, HP
*
* - Conversion to run under Linux and Apache
*
*/

#include <time.h>

/*
#include <sys/types.h>
*/

#define BOOLEAN int
#define BOOL int
#define VMS 0
#define LINEMAX 256
#define FALSE 0
#ifdef TRUE
#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;\
int dwThreadId;\
int 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)\
apr_thread_mutex_lock( gpTransactionPool->critSec );\
pData = gpTransactionPool->index[gpTransactionPool->iNextFree];\
INIT_TRANSACTION(type,pData);\
gpTransactionPool->iNextFree++;\
apr_thread_mutex_unlock( gpTransactionPool->critSec );

#define UNRESERVE_TRANSACTION_STRUCT(type,pData)\
apr_thread_mutex_lock( gpTransactionPool->critSec );\
CHECK_TRANSACTION(type,pData);\
gpTransactionPool->index[--gpTransactionPool->iNextFree] =
pData;\
apr_thread_mutex_unlock( gpTransactionPool->critSec );

typedef struct
{
apr_thread_mutex_t * critSec;
int iNextFree;
#ifdef FFE_DEBUG
int iMaxIndex;
int iTransactionSize;
int iHistoryId;
struct
{

```

```

int iOpCode;
int iFailure;
int iReserveHistoryId;
int iUnreserveHistoryId;
int iType;
int dwThreadId;
int 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
consistent with */
/* the io_delivery struct. Note also that the input portion of the
delivery */
/* data can be simply memcpied from the input to the input/output
struct. */
#define I_DELIVERY \
    CONN_DATA\
    time_t    queue_time;\
    int    delta_time;      /* in milliseconds */\
    struct timeval tbegin;\
    struct timeval tend;\
    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];
    double c_discount;
    double w_tax;
    double 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;
double 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 {
```

```

BOOLEAN asynchronous;
BOOLEAN generic;
int    num_gc;
int    num_dy;
int    num_no;
int    num_os;
int    num_pt;
int    num_sl;
BOOLEAN 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 /* TPCSTRUCT_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
*
*****
*****/

/*
*
*

```

```

* Modification history:
*
*
*      08/01/2002      Andrew Bond, HP
*
*                      - Conversion to run under Linux
*
*/

#include <stdlib.h> /* stg - added for change to Tuxedo */
#include <string.h>
#include <stdio.h>
#include <sys/time.h>

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

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

#include <tpcc.h>

#include <pthread.h>

/* tuxedo include files */
#include <atmi.h>

#ifdef FFE_DEBUG
# include <crtdbg.h>
#endif

#define TOTAL_ADMIN_CONNECTIONS 1

#define FILENAMESIZE 256

static pthread_key_t initkey;

static pthread_once_t initkey_once = PTHREAD_ONCE_INIT;

static void doinit(void)
{
    pthread_key_create(&initkey, NULL);
}

/* Returns non-zero if thread has been initialized already. */
static int IsInited(void)
{
    void *p;

```

```

pthread_once(&initkey_once, doinit);
p = pthread_getspecific(initkey);
return (p == NULL);
}

static void NowInited(void)
{
pthread_setspecific(initkey, (void *)1); /* non-NULL value. */
}

/* 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 retcode = -1;
int count = 0;
static int num_tpinit = 0;

#if (DEBUG == 1)
fprintf(MyLogFile, "Entering IsTuxInit\n");
fflush(MyLogFile);
#endif
if(IsInited())
{
while(count < 20)
{
if(NULL == (tpinitbuf = (TPINIT *) tmalloc("TPINIT", NULL,
sizeof(TPINIT))))
{
TPCCerr("error with tmalloc - %d - %d", tperrno, count);
}
else
{
tpinitbuf->flags |= TPMULTICONTEXTS;
itoa(++num_tpinit, tpinitbuf->cltname);
retcode = tpinit(tpinitbuf);
}
}
}
#if (DEBUG == 1)
fprintf(MyLogFile, "Back from tpinit, retcode=%d\n",
retcode);
fflush(MyLogFile);
#endif
if(-1 != retcode)
{

```

```

NowInited();
tpfree((char*)tpinitbuf);
break;
}
else
{
TPCCerr("error with TPINIT - %s (%d) - %d\n\t\t\t..%s..",
tpsterror(tperrno),
tperrno,
count,
tpsterrordetail( tperrorid( 0 ), 0 ));
tpfree((char*)tpinitbuf);
}
}

count++;
if(count > 50)
{
retcode = -1;
TPCCerr("exceeded 50 trys in TPINIT");
}

sleep(10);
}
/*
sleep(50);
*/
if( -1 != retcode)
return ERR_DB_SUCCESS;
else
return(retcode);
}

return ERR_DB_SUCCESS;
}

/* stg - end IsTuxInit function */

/* 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;
}

int TPCCDelivery( pDeliveryData pDelivery)
{
    int                retcode;
    struct timezone    tz;

    time( &pDelivery->queue_time );

    gettimeofday(&pDelivery->tbegin, &tz);

    retcode = TPCCDeliveryDeferred(pDelivery);

    if ( ERR_DB_PENDING != retcode )
    {
        if( ERR_DB_SUCCESS != retcode)
        {
            /* send a flag to the reducer to mark an error on the
            delivery */
            pDelivery->queue_time = 1;
            DELIErrorMessage(retcode);
        }
    }

    return ERR_DB_SUCCESS;
}

/* 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);

    #if (DEBUG == 1)
        fprintf(MyLogFile, "Entering TPCCDeliveryDeferred\n");
        fflush(MyLogFile);
    #endif

    /* 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) tmalloc("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,TPNOREPLY|TPSIGRSTRT|TPNOTIME);
    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);

    #if (DEBUG == 1)
        fprintf(MyLogFile, "Entering TPCCNewOrder\n");
        fflush(MyLogFile);
    #endif

    /* 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) tmalloc("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, (long *)&nosiz, TPSIGRSTRT|TPNOTIME);

    if( -1 == retcode )
    {
        TPCCerr("tpcall - new order: %d", tperrno);
        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);

    #if (DEBUG == 1)
        fprintf(MyLogFile, "Entering TPCCOrderStatus\n");
        fflush(MyLogFile);
    #endif

    /* 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) tmalloc("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, (long *)&ossiz, TPSIGRSTRT|TPNOTIME);
    #if (DEBUG == 1)
        fprintf(MyLogFile, "TPCCOrderStatus:tpcall returned %d\n",
retcode);
        fflush(MyLogFile);
    #endif
    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);

#ifdef (DEBUG == 1)
        fprintf(MyLogFile, "Entering TPCCPayment\n");
        fflush(MyLogFile);
#endif

    /* 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) tmalloc("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, TPSIGRSTRT|TPNOTIME);
    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);

#ifdef (DEBUG == 1)
        fprintf(MyLogFile, "Entering TPCCStockLevel\n");
        fflush(MyLogFile);
#endif

    /* 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) tmalloc("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, (long *)&slsiz, TPSIGRSTRT|TPNOTIME);
    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 */

/*
***+
** FUNCTION NAME: TPCCStartup
**--
*/
int
TPCCStartup()
{
    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;

    /* shut down the servers listed in the TUXCONFIG file (ubb* file)
    */
    retcode = system("tmsshutdown -y");
    if (retcode != 0)
    {
        TPCCErr("Error shutting the tuxedo servers down.");
        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.
*
*
*
*****_*/

/*
*
*
* Modification history:
*
*
* 08/01/2002 Andrew Bond, HP
*
* - Conversion to run under Linux
*
*/

#include <errno.h>
#include <unistd.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

```

```

#include <sys/types.h>
#include <sys/time.h>

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

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

static FILE *fpLog = NULL;          /* pointer to log file
*/

FILE *LogFile;
FILE *MyLogFile;

#define MAXNUMDIGITS 10

char      szTpccLogPath[FILENAME_SIZE];
char      szNumber[MAXNUMDIGITS];

/* 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;

    pDelivery->delta_time = ((pDelivery->tend.tv_sec - pDelivery-
>tbegin.tv_sec) * 1000) + (int)ceil((pDelivery->tend.tv_usec -
pDelivery->tbegin.tv_usec)/1000);

    memcpy( &start, localtime( &pDelivery->tbegin.tv_sec), sizeof(
start ));

    memcpy( &end, localtime( &pDelivery->tend.tv_sec), sizeof( end
));

    fprintf( fpLog,
            "%4.4d/%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",
            1900+start.tm_year, start.tm_mon+1, start.tm_mday,
            start.tm_hour, start.tm_min, start.tm_sec, pDelivery-
>tbegin.tv_usec/1000,
            end.tm_hour, end.tm_min, end.tm_sec, pDelivery-
>tend.tv_usec/1000,
            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] );

    fflush(fpLog);

    return;
}

```

```

/*
***+
** FUNCTION NAME: tpsvrinit
**--
*/
int
tpsvrinit( int argc, char *argv[] )
{
    BOOL    bLog;

    /* stg next two lines not needed for v6 web ora tux app code
    StartupData  Startup;
    pStartupData  pStartup = &Startup; */

    int status;

    char  szTmp[FILENAME_SIZE];
    LoginData  login;

    /* to avoid compiler errors */
    argc = argc;
    argv = argv;

    /* used for debugging the server code */
    /*
    sleep(30000);
    */

    userlog("Starting tpcc server");

    /* Get login data from file settings */
    status = GetConfigValue("Server", (char *)&szTmp);
    if ( status != ERROR_SUCCESS )
        return ERR_CANT_FIND_SERVER_VALUE;
    strcpy(login.szServer, szTmp);

    status = GetConfigValue("Database", (char *)&szTmp);
    if ( status != ERROR_SUCCESS )
        return ERR_CANT_FIND_DATABASE_VALUE;
    strcpy(login.szDatabase, szTmp);

    status = GetConfigValue("User", (char *)&szTmp);
    if ( status != ERROR_SUCCESS )
        return ERR_CANT_FIND_USER_VALUE;
    strcpy(login.szUser, szTmp);

    status = GetConfigValue("Password", (char *)&szTmp);
    if ( status != ERROR_SUCCESS )
        return ERR_CANT_FIND_PASSWORD_VALUE;

```

```

strcpy(login.szPassword, szTmp);

/* Get Path registry value */
status = GetConfigValue("PATH", (char *)&szTmp);
if (status != ERROR_SUCCESS )
    return ERR_CANT_FIND_PATH_VALUE;
strcpy (szTpccLogPath, szTmp);

if (ERROR_SUCCESS == status)
{
    /* set application name */
    /* strcpy( pStartup->Login.databaseLogin.szApplication,
    "TUX_SRV" ); */

    TPCCStartupDB();

    /* 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';

    strcpy(szTmp, szTpccLogPath);
    strcat(szTmp, "delilog");
    itoa(getpid(), szNumber);
    strcat(szTmp, szNumber);
    fpLog = fopen(szTmp, "wb");
    if ( NULL == fpLog )
        return ERR_CANNOT_CREATE_RESULTS_FILE;

    status = TPCCConnectDB( (OraContext **)&DBC, &login );

    if(ERR_DB_SUCCESS != status)
    {
        TPCCErr( "tpsvrinit : Error logging into db." );
        return ERR_DB_ERROR;
    }
    TPCCErr( "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 )
{
    struct timeval tend;
    struct timezone tz;

    pDeliveryData ptr;

    ptr = (pDeliveryData)dy_wksp->data;

    ptr->status = TPCCDeliveryDB( DBC, ptr );

    gettimeofday(&ptr->tend, &tz);

    /* update log */
    DELILog( 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(TPPFAIL, ptr->status, sl_wksp->data, 0L, 0);
}

*****
util.c
*****

/*
 *
 *      08/01/2002      Andrew Bond, HP
 *                      - Configuration values are stored in a
 *                      filesystem file under Linux
 *                      rather than the Windows registry.
 */

#include <stdio.h>

#define MAXCFGLINE 255
#define CONFIGFILENAME "/usr/local/etc/tpcc.conf"

/* FUNCTION: int GetConfigValue(char *option, char *value)
 * Read the Linux tpcc configuration file
 */
int GetConfigValue(char *option, char *value)
{
    FILE *cfFD;
    char line[MAXCFGLINE];
    char optname[MAXCFGLINE];
    char *poptname, *tmpValue, *linep;
    int full_len, half_len, len;
    short notfound=1;

    poptname=(char *)&optname;

    cfFD=fopen(CONFIGFILENAME, "r");

    if (cfFD == NULL)
    {
        printf("Error opening file\n");
        return -1;
    }
    linep=(char *)&line;

    while ((fgets(linep, MAXCFGLINE, cfFD) != NULL) && (notfound))
    {
        tmpValue=(char *)index(linep, '=');

        if (tmpValue==NULL)
        {
            printf("Equals sign not found\n");
            continue;
        }

        full_len=strlen(linep);
        half_len=strlen(tmpValue);

        strncpy(poptname,linep, full_len-half_len);
        optname[full_len-half_len] = '\0';
        tmpValue++;

        if (!strcmp(optname, option))

```

```

{
    len=strlen(tmpValue);
    strncpy(value, tmpValue, len-1);
    value[len-1] = '\0';
    notfound=0;
}
}

fclose(cfFD);

if (notfound)
    return(0);
else
    return(1);
}

*****
Makefile
*****

##
## Makefile -- Build procedure for sample tpcc Apache module
## Autogenerated via `apxs -n tpcc -O2`.
##

builddir=.
top_srcdir=/usr/src/redhat/BUILD/httpd-2.0.36
top_builddir=/usr/src/redhat/BUILD/httpd-2.0.36
include /usr/src/redhat/BUILD/httpd-2.0.36/build/special.mk

# the used tools
#APXS=/usr/sbin/apxs
APXS=/usr/local/ap2/sbin/apxs
APACHECTL=/usr/sbin/apachectl
TUXDIR=/home/bea/tuxedo8.0
ORAHOME=/home/oracle/OraHome1

# additional user defines, includes and libraries
#DEF=-Dmy_define=my_value
#LIB=-Lmy/lib/dir -lmylib
APACHEINC=-I/usr/local/ap2/include/apache
INC=-I. $(APACHEINC) $(ORAINC) $(TUXINC)
DEF=-Wall
TUXINC=-I/home/bea/tuxedo8.0/include
ORAINC=-I/home/oracle/OraHome1/rdbms/demo -
I/home/oracle/OraHome1/rdbms/public

AP_LIBS = $(top_builddir)/lib/libapr.a
TUX_LIBS = $(TUXDIR)/lib/libtux.a \
$(TUXDIR)/lib/libbuft.a \
$(TUXDIR)/lib/libengine.a \
$(TUXDIR)/lib/libtrpc.a \
$(TUXDIR)/lib/libfml.a \
$(TUXDIR)/lib/libfml32.a

LINUX_LIBS = /usr/lib/libpthread.a \
/usr/lib/libdl.a \
/usr/lib/libm.a

ORA_LIBS = $(ORAHOME)/lib/libclient9.a \
$(ORAHOME)/lib/libcore9.a \
$(ORAHOME)/lib/libgeneric9.a \
$(ORAHOME)/lib/libcommon9.a \
$(ORAHOME)/lib/libnls9.a

TUX_SRV_OBJS = tux_srv.o \
oracle_db8.o \
oracle_txns8.o \
logfile_tux.o \
util.o

MOD_TPCC_OBJS = mod_tpcc.o \
logfile_mod.o \
tpcc.o \
tux_cli.o \
util.o

# the default target
tpcc: local-shared-build

# compile the DSO file
mod_tpcc.so: $(MOD_TPCC_OBJS)
$(APXS) -Wc,-O2 -c $(DEF) $(INC) $(LIB) -L$(TUXDIR)/lib
$(MOD_TPCC_OBJS) -ltux -lbuft -lfml -lfml32 -lengine -ldl -lpthread

mod_tpcc.o: mod_tpcc.c
gcc -O2 -c -DEAPI $(DEF) $(INC) $(LIB) mod_tpcc.c

logfile_mod.o: logfile_mod.c
gcc -O2 -c $(DEF) $(INC) $(LIB) logfile_mod.c

logfile_tux.o: logfile_tux.c
gcc -O2 -c $(DEF) $(INC) $(LIB) logfile_tux.c

tpcc.o: tpcc.c
gcc -O2 -c $(DEF) $(INC) $(LIB) tpcc.c

```

```

util.o: util.c
gcc -O2 -c $(DEF) $(INC) $(LIB) util.c

tux_cli.o: tux_cli.c
gcc -O2 -c $(DEF) $(INC) $(LIB) tux_cli.c

oracle_db8.o: oracle_db8.c
gcc -O2 -c $(DEF) $(INC) $(LIB) oracle_db8.c

oracle_txns8.o: oracle_txns8.c
gcc -O2 -c $(DEF) $(INC) $(LIB) oracle_txns8.c

tux_srv.o: tux_srv.c
gcc -O2 -c $(DEF) $(INC) $(LIB) tux_srv.c

delirpt: delirpt.c
gcc -O2 -o delirpt delirpt.c

#tuxora: $(TUX_SRV_OBJS)
# gcc $(TUX_SRV_OBJS) $(TUX_LIBS) -Wl,-rpath $(TUXDIR)/lib
$(ORAHOME)/lib/libclntst9.a $(LINUX_LIBS) -o tuxora

BS-7dc9.o: BS-7dc9.c
gcc -c -I$(TUXDIR)/include BS-7dc9.c

BS-deli.o: BS-deli.c
gcc -c -I$(TUXDIR)/include BS-deli.c

BS-pay.o: BS-pay.o.c
gcc -c -I$(TUXDIR)/include BS-pay.o.c

BS-ordo.o: BS-ordo.c
gcc -c -I$(TUXDIR)/include BS-ordo.c

BS-stoo.o: BS-stoo.c
gcc -c -I$(TUXDIR)/include BS-stoo.c

BS-newo.o: BS-newo.c
gcc -c -I$(TUXDIR)/include BS-newo.c

BS-tpcc.o: BS-tpcc.c
gcc -c -I$(TUXDIR)/include BS-tpcc.c

tuxora: $(TUX_SRV_OBJS)
gcc -o tuxora -L$(TUXDIR)/lib $(TUX_SRV_OBJS) BS-7dc9.o -ltux -
lbuft -lfml -lfml32 -lengine -ldl -lpthread /usr/lib/libcrypt.a
$(ORAHOME)/lib/libclntst9.a $(LINUX_LIBS)

tpccora: $(TUX_SRV_OBJS) BS-tpcc.o
gcc -o tpccora -L$(TUXDIR)/lib $(TUX_SRV_OBJS) BS-tpcc.o -ltux -
lbuft -lfml -lfml32 -lengine -ldl -lpthread /usr/lib/libcrypt.a
$(ORAHOME)/lib/libclntst9.a $(LINUX_LIBS)

deliora: $(TUX_SRV_OBJS) BS-deli.o
gcc -o deliora -L$(TUXDIR)/lib $(TUX_SRV_OBJS) BS-deli.o -ltux -
lbuft -lfml -lfml32 -lengine -ldl -lpthread /usr/lib/libcrypt.a
$(ORAHOME)/lib/libclntst9.a $(LINUX_LIBS)

stoora: $(TUX_SRV_OBJS) BS-stoo.o
gcc -o stoora -L$(TUXDIR)/lib $(TUX_SRV_OBJS) BS-stoo.o -ltux -
lbuft -lfml -lfml32 -lengine -ldl -lpthread /usr/lib/libcrypt.a
$(ORAHOME)/lib/libclntst9.a $(LINUX_LIBS)

ordora: $(TUX_SRV_OBJS) BS-ordo.o
gcc -o ordora -L$(TUXDIR)/lib $(TUX_SRV_OBJS) BS-ordo.o -ltux -
lbuft -lfml -lfml32 -lengine -ldl -lpthread /usr/lib/libcrypt.a
$(ORAHOME)/lib/libclntst9.a $(LINUX_LIBS)

payora: $(TUX_SRV_OBJS) BS-pay.o
gcc -o payora -L$(TUXDIR)/lib $(TUX_SRV_OBJS) BS-pay.o -ltux -
lbuft -lfml -lfml32 -lengine -ldl -lpthread /usr/lib/libcrypt.a
$(ORAHOME)/lib/libclntst9.a $(LINUX_LIBS)

newora: $(TUX_SRV_OBJS) BS-newo.o
gcc -o newora -L$(TUXDIR)/lib $(TUX_SRV_OBJS) BS-newo.o -ltux -
lbuft -lfml -lfml32 -lengine -ldl -lpthread /usr/lib/libcrypt.a
$(ORAHOME)/lib/libclntst9.a $(LINUX_LIBS)

tpccora:
# install the shared object file into Apache
install: install-modules

replace:
cp .libs/mod_tpcc.so /etc/httpd/modules
cp tuxora $(TUXDIR)

# cleanup
clean:
-rm -f mod_tpcc.o mod_tpcc.so

# simple test
test: reload
lynx -mime_header http://localhost/tpcc

# reload the module by installing and restarting Apache
reload: install restart

# the general Apache start/restart/stop procedures
start:
$(APACHECTL) start

```

```

restart:
$(APACHECTL) restart
stop:
$(APACHECTL) stop

*****
modules.mk
*****

mod_tpcc.la: mod_tpcc.slo
$(SH_LINK) -rpath $(libexecdir) -module -avoid-version
mod_tpcc.lo
DISTCLEAN_TARGETS = modules.mk
shared = mod_tpcc.la

*****
linux_8.pr
*****

echo

#####
#####
#
#
#
# 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 0.005
set type_rate 0.0

echo

```



```

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

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

#####
#
# FRONT END NETWORK VARIABLES #
#
#####
# FE_NAMES          A comma separated list of the front end network
# node names.
#
# FE_USER_COUNTS A comma separated 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 separated list of FEs on which the
# aides will
# operated.

set network_variable FE_NAMES
lc1,lc2,lc3,lc4,lc5,lc6,lc7,lc8,lc9,lc10,lc11,lc12,lc13,lc14,lc15,lc16
c16
set network_variable FE_USER_COUNTS
7200,7200,7200,7200,7200,7200,7200,7200,7200,7200,7200,7200,7200,7200,7200,7200,7200,7200,7200,7200
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)

```

```

#
# users for                    8 - user sut data logging (required by web
#                               error checking)
#
# In general, leave this at 12 for web clients
# doing binary                data reduction, and 13 for web clients doing
# asci data                    reduction.
#
# TPCC_USER_FLUSH_LOG is whether or not to flush every write to the
# log.
#
# DURABILITY_LOGGING is whether or not to parse new order response
# pages for                    durability data (to be sent to reducer). This
# variable
# is a boolean so legal values are 0,f,F and 1,t,T.
#
# C_LAST is the constant value used for customer last names.
# This value must be chosen with care. It must be based on
# the value you used when populating your database.

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

#####
#
# CONFIGURATION NETWORK VARIABLES #
#
#####
#
# CGI_SCRIPT_NAME is the name of the application to run on the
# front ends.
#
# LOAD_DLL_TIMEOUT is how long master should wait (in seconds) for
# the dll
# to initially load before timing out.
#
#set network_variable CGI_SCRIPT_NAME /webacmsxploop.dll
#set network_variable CGI_SCRIPT_NAME /webacmsxploop1500.dll
#set network_variable CGI_SCRIPT_NAME /webacmsxpورا84.dll
#set network_variable CGI_SCRIPT_NAME /webacmsxpورا8.dll
set network_variable CGI_SCRIPT_NAME /tpcc
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 separate 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
# resume) but
# pause, so the act of logging in is just a
# 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 /usr/users/tpcc/results/
set network_variable TEST_RESULTS_DIR /results/

#set network_variable WARMUP_TIME 300.0
#set network_variable STEADY_STATE_TIME 120.0
#set network_variable MEASUREMENT_INTERVAL 120.0
#set network_variable COOLDOWN_TIME 120.0

set network_variable WARMUP_TIME 3600.0
set network_variable STEADY_STATE_TIME 10800.0
set network_variable MEASUREMENT_INTERVAL 7500.0
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 w2k
set network_variable RESUME_DELAY 1.50

```

```

set network_variable STOP_DELAY 0.05
# 100 5000
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 30.0
#set network_variable NO_THINK_TIME 12.02
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
variable 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
# mllog.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 ral 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 DATABASE_TYPE MsSql
#set network_variable DATABASE_USERNAME tpcc
#set network_variable DATABASE_PASSWORD tpcc

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

```

```

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

echo

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

disable stop

#start

*****
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/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);
-- 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 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;

      IF SQL%NOTFOUND
      THEN
        raise NO_DATA_FOUND;
      END IF;

      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;

*****
tkvcpnw.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',

```

```

'B'))
        DECODE(instr(s_data,'ORIGINAL'), 0, 'G',
        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);

-- copying :d_id in local variable is important - lots of
instr.
            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;
            END IF;

            ROLLBACK;
            END IF;

            EXIT;

        EXCEPTION

```

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

Appendix B: Database Design

```
*****
addtempts.sh
*****

#!/bin/sh

sqlplus 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
*****

#!/bin/sh

sqlplus 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 ;

!

*****
c_stat.sql
*****

rem
rem
rem=====
rem
rem      Copyright (c) 1997 Oracle Corp, Redwood Shores, CA
rem
rem      All Rights Reserved
rem=====
rem
rem FILENAME
rem      cs_tpcc.sql
rem DESCRIPTION
rem      Create tables for saving TPC-C results.
rem=====
rem
rem Usage: sqlplus user/password @cs_tpcc.sql
rem spool log/cs_tpcc.log

connect tpcc/tpcc;
set echo on

DROP TABLE tpcc_run_desc;
DROP TABLE tpcc_run_int;
DROP TABLE bench_run_int;
DROP TABLE tpcc_back_res;
DROP TABLE tpcc_user_res;
DROP TABLE bench_user_res;
DROP TABLE tpcc_tpm;
DROP TABLE tpcc_new_res;
DROP TABLE bench_new_res;
DROP TABLE tpcc_pay_res;
DROP TABLE bench_pay_res;
DROP TABLE tpcc_ord_res;
DROP TABLE bench_ord_res;
DROP TABLE tpcc_del_res;
DROP TABLE bench_del_res;
```

```
DROP TABLE tpcc_sto_res;
DROP TABLE bench_sto_res;

rem
rem description of a run
rem
CREATE TABLE tpcc_run_desc
(
  run_name      VARCHAR2(20),
  rundate       DATE,
  time          NUMBER,
  rampup        NUMBER,
  rampdown      NUMBER,
  warehouses    NUMBER,
  customers     NUMBER,
  users         NUMBER,
  driver        VARCHAR2(40),
  commnt       VARCHAR2(80)
);

rem
rem throughput of new order transactions
rem
CREATE TABLE tpcc_run_int
(
  run_name      VARCHAR2(20),
  interval      NUMBER,
  interval_count NUMBER,
  response_time NUMBER,
  think_time    NUMBER
);

rem
rem throughput of new order transactions
rem
CREATE TABLE bench_run_int
(
  run_name      VARCHAR2(20),
  proc_no       NUMBER,
  interval      NUMBER,
  interval_count NUMBER,
  response_time NUMBER,
  think_time    NUMBER
);

rem
rem Results from delivery servers
rem
CREATE TABLE tpcc_back_res
(
  run_name      VARCHAR2(20),
  in_timing_int NUMBER,
  fast          NUMBER,
  resp_time    NUMBER,
  retries       NUMBER
);

rem
rem Aggregate results for all generators.
rem These results are from the measurement interval only.
rem These results are used to calculate the TPS rate over
rem the measurement interval.
rem
CREATE TABLE tpcc_user_res
(
  run_name      VARCHAR2(20),
  no_men        NUMBER,
  fast_men      NUMBER,
  in_flight_men NUMBER,
  retry_men     NUMBER,
  min_time_men  NUMBER,
  max_time_men  NUMBER,
  sum_time_men  NUMBER,
  ninety_per_men NUMBER,
  think_min_men NUMBER,
  think_max_men NUMBER,
  think_sum_men NUMBER,
  key_min_men   NUMBER,
  key_max_men   NUMBER,
  key_sum_men   NUMBER,
  no_new        NUMBER,
  fast_new      NUMBER,
  in_flight_new NUMBER,
  retry_new     NUMBER,
  min_time_new  NUMBER,
  max_time_new  NUMBER,
  sum_time_new  NUMBER,
  ninety_per_new NUMBER,
  think_min_new NUMBER,
  think_max_new NUMBER,
  think_max_new NUMBER
);
```



```

think_sum_new NUMBER,
  key_min_new NUMBER,
  key_max_new NUMBER,
key_sum_new NUMBER,
remote_new NUMBER,
rollback_new NUMBER,
sum_ol_new NUMBER,
remote_ol_new NUMBER,
  allrollback_new NUMBER,
no_pay NUMBER,
fast_pay NUMBER,
in_flight_pay NUMBER,
  retry_pay NUMBER,
min_time_pay NUMBER,
max_time_pay NUMBER,
sum_time_pay NUMBER,
  ninety_per_pay NUMBER,
  think_min_pay NUMBER,
  think_max_pay NUMBER,
think_sum_pay NUMBER,
  key_min_pay NUMBER,
  key_max_pay NUMBER,
key_sum_pay NUMBER,
remote_pay NUMBER,
bylast_pay NUMBER,
no_ord NUMBER,
fast_ord NUMBER,
in_flight_ord NUMBER,
  retry_ord NUMBER,
min_time_ord NUMBER,
max_time_ord NUMBER,
sum_time_ord NUMBER,
  ninety_per_ord NUMBER,
  think_min_ord NUMBER,
  think_max_ord NUMBER,
think_sum_ord NUMBER,
  key_min_ord NUMBER,
  key_max_ord NUMBER,
key_sum_ord NUMBER,
bylast_ord NUMBER,
no_del NUMBER,
fast_del NUMBER,
in_flight_del NUMBER,
  retry_del NUMBER,
min_time_del NUMBER,
max_time_del NUMBER,
sum_time_del NUMBER,
  ninety_per_del NUMBER,
  think_min_del NUMBER,
  think_max_del NUMBER,
think_sum_del NUMBER,
  key_min_del NUMBER,
  key_max_del NUMBER,
key_sum_del NUMBER,
no_sto NUMBER,
fast_sto NUMBER,
in_flight_sto NUMBER,
  retry_sto NUMBER,
min_time_sto NUMBER,
max_time_sto NUMBER,
sum_time_sto NUMBER,
  ninety_per_sto NUMBER,
  think_min_sto NUMBER,
  think_max_sto NUMBER,
think_sum_sto NUMBER,
  key_min_sto NUMBER,
  key_max_sto NUMBER,
key_sum_sto NUMBER,
cpu_time NUMBER,
  deadlocks NUMBER
);

rem
rem Results from individual generators.
rem These results are from the measurement interval only.
rem These results are used to calculate the TPS rate over
rem the measurement interval.
rem
rem CREATE TABLE bench_user_res
rem (
rem   run_name      VARCHAR2(20),
rem   audit_str    VARCHAR2(10),
rem   proc_no      NUMBER,
rem   hid          NUMBER,
rem   no_men       NUMBER,
rem   fast_men     NUMBER,
rem   in_flight_men NUMBER,
rem   retry_men    NUMBER,
rem   min_time_men NUMBER,
rem   max_time_men NUMBER,
rem   sum_time_men NUMBER,
rem   ninety_per_men NUMBER,
rem   think_min_men NUMBER,
rem   think_max_men NUMBER,
rem   think_sum_men NUMBER,
rem   key_min_men  NUMBER,
rem   key_max_men  NUMBER,
rem   key_sum_men  NUMBER,
rem   no_new       NUMBER,
rem   fast_new     NUMBER,
rem   in_flight_new NUMBER,

```

```

  retry_new     NUMBER,
  min_time_new NUMBER,
  max_time_new NUMBER,
  sum_time_new NUMBER,
  ninety_per_new NUMBER,
  think_min_new NUMBER,
  think_max_new NUMBER,
  think_sum_new NUMBER,
  key_min_new  NUMBER,
  key_max_new  NUMBER,
  key_sum_new  NUMBER,
  remote_new   NUMBER,
  rollback_new NUMBER,
  sum_ol_new   NUMBER,
  remote_ol_new NUMBER,
  allrollback_new NUMBER,
  no_pay       NUMBER,
  fast_pay     NUMBER,
  in_flight_pay NUMBER,
  retry_pay    NUMBER,
  min_time_pay NUMBER,
  max_time_pay NUMBER,
  sum_time_pay NUMBER,
  ninety_per_pay NUMBER,
  think_min_pay NUMBER,
  think_max_pay NUMBER,
  think_sum_pay NUMBER,
  key_min_pay  NUMBER,
  key_max_pay  NUMBER,
  key_sum_pay  NUMBER,
  remote_pay   NUMBER,
  bylast_pay   NUMBER,
  no_ord       NUMBER,
  fast_ord     NUMBER,
  in_flight_ord NUMBER,
  retry_ord    NUMBER,
  min_time_ord NUMBER,
  max_time_ord NUMBER,
  sum_time_ord NUMBER,
  ninety_per_ord NUMBER,
  think_min_ord NUMBER,
  think_max_ord NUMBER,
  think_sum_ord NUMBER,
  key_min_ord  NUMBER,
  key_max_ord  NUMBER,
  key_sum_ord  NUMBER,
  bylast_ord   NUMBER,
  no_del       NUMBER,
  fast_del     NUMBER,
  in_flight_del NUMBER,
  retry_del    NUMBER,
  min_time_del NUMBER,
  max_time_del NUMBER,
  sum_time_del NUMBER,
  ninety_per_del NUMBER,
  think_min_del NUMBER,
  think_max_del NUMBER,
  think_sum_del NUMBER,
  key_min_del  NUMBER,
  key_max_del  NUMBER,
  key_sum_del  NUMBER,
  no_sto       NUMBER,
  fast_sto     NUMBER,
  in_flight_sto NUMBER,
  retry_sto    NUMBER,
  min_time_sto NUMBER,
  max_time_sto NUMBER,
  sum_time_sto NUMBER,
  ninety_per_sto NUMBER,
  think_min_sto NUMBER,
  think_max_sto NUMBER,
  think_sum_sto NUMBER,
  key_min_sto  NUMBER,
  key_max_sto  NUMBER,
  key_sum_sto  NUMBER,
  cpu_time     NUMBER,
  deadlocks    NUMBER
);

rem
rem Aggregate results for generators on each host.
rem These results are from the measurement interval only.
rem These results are used to calculate the TPM rate over
rem the measurement interval.
rem
rem CREATE TABLE tpcc_tpm
rem (
rem   run_name      VARCHAR2(20),
rem   hid          NUMBER,
rem   no_new       NUMBER
rem );

rem
rem Aggregate results for new order transactions.
rem These results are from the measurement interval only.
rem
rem CREATE TABLE tpcc_new_res
rem (
rem   run_name      VARCHAR2(20),
rem   repl         NUMBER,
rem   repl2        NUMBER,

```

```

rep3      NUMBER,
rep4      NUMBER,
rep5      NUMBER,
rep6      NUMBER,
rep7      NUMBER,
rep8      NUMBER,
rep9      NUMBER,
rep10     NUMBER,
rep11     NUMBER,
rep12     NUMBER,
rep13     NUMBER,
rep14     NUMBER,
rep15     NUMBER,
rep16     NUMBER,
rep17     NUMBER,
rep18     NUMBER,
rep19     NUMBER,
rep20     NUMBER,
rep21     NUMBER,
rep22     NUMBER,
rep23     NUMBER,
rep24     NUMBER,
rep25     NUMBER,
rep26     NUMBER,
rep27     NUMBER,
rep28     NUMBER,
rep29     NUMBER,
rep30     NUMBER,
rep31     NUMBER,
rep32     NUMBER,
rep33     NUMBER,
rep34     NUMBER,
rep35     NUMBER,
rep36     NUMBER,
rep37     NUMBER,
rep38     NUMBER,
rep39     NUMBER,
rep40     NUMBER,
rep41     NUMBER,
rep42     NUMBER,
rep43     NUMBER,
rep44     NUMBER,
rep45     NUMBER,
rep46     NUMBER,
rep47     NUMBER,
rep48     NUMBER,
rep49     NUMBER,
rep50     NUMBER,
rep51     NUMBER,
rep52     NUMBER,
rep53     NUMBER,
rep54     NUMBER,
rep55     NUMBER,
rep56     NUMBER,
rep57     NUMBER,
rep58     NUMBER,
rep59     NUMBER,
rep60     NUMBER,
rep61     NUMBER,
rep62     NUMBER,
rep63     NUMBER,
rep64     NUMBER,
rep65     NUMBER,
rep66     NUMBER,
rep67     NUMBER,
rep68     NUMBER,
rep69     NUMBER,
rep70     NUMBER,
rep71     NUMBER,
rep72     NUMBER,
rep73     NUMBER,
rep74     NUMBER,
rep75     NUMBER,
rep76     NUMBER,
rep77     NUMBER,
rep78     NUMBER,
rep79     NUMBER,
rep80     NUMBER,
rep81     NUMBER,
rep82     NUMBER,
rep83     NUMBER,
rep84     NUMBER,
rep85     NUMBER,
rep86     NUMBER,
rep87     NUMBER,
rep88     NUMBER,
rep89     NUMBER,
rep90     NUMBER,
rep91     NUMBER,
rep92     NUMBER,
rep93     NUMBER,
rep94     NUMBER,
rep95     NUMBER,
rep96     NUMBER,
rep97     NUMBER,
rep98     NUMBER,
rep99     NUMBER,
rep100    NUMBER,
thk1      NUMBER,
thk2      NUMBER,
thk3      NUMBER,

```

```

thk4      NUMBER,
thk5      NUMBER,
thk6      NUMBER,
thk7      NUMBER,
thk8      NUMBER,
thk9      NUMBER,
thk10     NUMBER,
thk11     NUMBER,
thk12     NUMBER,
thk13     NUMBER,
thk14     NUMBER,
thk15     NUMBER,
thk16     NUMBER,
thk17     NUMBER,
thk18     NUMBER,
thk19     NUMBER,
thk20     NUMBER,
thk21     NUMBER,
thk22     NUMBER,
thk23     NUMBER,
thk24     NUMBER,
thk25     NUMBER,
key1      NUMBER,
key2      NUMBER,
key3      NUMBER,
key4      NUMBER,
key5      NUMBER,
key6      NUMBER,
key7      NUMBER,
key8      NUMBER,
key9      NUMBER,
key10     NUMBER
);

```

```

rem
rem Results for new order transactions.
rem These results are from the measurement interval only.
rem

```

```

CREATE TABLE bench_new_res
(
  run_name      VARCHAR2(20),
  audit_str     VARCHAR2(10),
  proc_no      NUMBER,
  rep1          NUMBER,
  rep2          NUMBER,
  rep3          NUMBER,
  rep4          NUMBER,
  rep5          NUMBER,
  rep6          NUMBER,
  rep7          NUMBER,
  rep8          NUMBER,
  rep9          NUMBER,
  rep10         NUMBER,
  rep11         NUMBER,
  rep12         NUMBER,
  rep13         NUMBER,
  rep14         NUMBER,
  rep15         NUMBER,
  rep16         NUMBER,
  rep17         NUMBER,
  rep18         NUMBER,
  rep19         NUMBER,
  rep20         NUMBER,
  rep21         NUMBER,
  rep22         NUMBER,
  rep23         NUMBER,
  rep24         NUMBER,
  rep25         NUMBER,
  rep26         NUMBER,
  rep27         NUMBER,
  rep28         NUMBER,
  rep29         NUMBER,
  rep30         NUMBER,
  rep31         NUMBER,
  rep32         NUMBER,
  rep33         NUMBER,
  rep34         NUMBER,
  rep35         NUMBER,
  rep36         NUMBER,
  rep37         NUMBER,
  rep38         NUMBER,
  rep39         NUMBER,
  rep40         NUMBER,
  rep41         NUMBER,
  rep42         NUMBER,
  rep43         NUMBER,
  rep44         NUMBER,
  rep45         NUMBER,
  rep46         NUMBER,
  rep47         NUMBER,
  rep48         NUMBER,
  rep49         NUMBER,
  rep50         NUMBER,
  rep51         NUMBER,
  rep52         NUMBER,
  rep53         NUMBER,
  rep54         NUMBER,
  rep55         NUMBER,
  rep56         NUMBER,
  rep57         NUMBER,
  rep58         NUMBER,

```

```

rep59      NUMBER,
rep60      NUMBER,
rep61      NUMBER,
rep62      NUMBER,
rep63      NUMBER,
rep64      NUMBER,
rep65      NUMBER,
rep66      NUMBER,
rep67      NUMBER,
rep68      NUMBER,
rep69      NUMBER,
rep70      NUMBER,
rep71      NUMBER,
rep72      NUMBER,
rep73      NUMBER,
rep74      NUMBER,
rep75      NUMBER,
rep76      NUMBER,
rep77      NUMBER,
rep78      NUMBER,
rep79      NUMBER,
rep80      NUMBER,
rep81      NUMBER,
rep82      NUMBER,
rep83      NUMBER,
rep84      NUMBER,
rep85      NUMBER,
rep86      NUMBER,
rep87      NUMBER,
rep88      NUMBER,
rep89      NUMBER,
rep90      NUMBER,
rep91      NUMBER,
rep92      NUMBER,
rep93      NUMBER,
rep94      NUMBER,
rep95      NUMBER,
rep96      NUMBER,
rep97      NUMBER,
rep98      NUMBER,
rep99      NUMBER,
rep100     NUMBER,
thk1       NUMBER,
thk2       NUMBER,
thk3       NUMBER,
thk4       NUMBER,
thk5       NUMBER,
thk6       NUMBER,
thk7       NUMBER,
thk8       NUMBER,

thk9       NUMBER,
thk10      NUMBER,
thk11      NUMBER,
thk12      NUMBER,
thk13      NUMBER,
thk14      NUMBER,
thk15      NUMBER,
thk16      NUMBER,
thk17      NUMBER,
thk18      NUMBER,
thk19      NUMBER,
thk20      NUMBER,
thk21      NUMBER,
thk22      NUMBER,
thk23      NUMBER,
thk24      NUMBER,
thk25      NUMBER,
key1       NUMBER,
key2       NUMBER,
key3       NUMBER,
key4       NUMBER,
key5       NUMBER,
key6       NUMBER,
key7       NUMBER,
key8       NUMBER,
key9       NUMBER,
key10      NUMBER
);

rem
rem Aggregate results for payment transactions.
rem These results are from the measurement interval only.
rem
CREATE TABLE tpcc_pay_res
(
  run_name    VARCHAR2(20),
  rep1        NUMBER,
  rep2        NUMBER,
  rep3        NUMBER,
  rep4        NUMBER,
  rep5        NUMBER,
  rep6        NUMBER,
  rep7        NUMBER,
  rep8        NUMBER,
  rep9        NUMBER,
  rep10       NUMBER,
  rep11       NUMBER,
  rep12       NUMBER,
  rep13       NUMBER,
  rep14       NUMBER,

```

```

rep15      NUMBER,
rep16      NUMBER,
rep17      NUMBER,
rep18      NUMBER,
rep19      NUMBER,
rep20      NUMBER,
rep21      NUMBER,
rep22      NUMBER,
rep23      NUMBER,
rep24      NUMBER,
rep25      NUMBER,
rep26      NUMBER,
rep27      NUMBER,
rep28      NUMBER,
rep29      NUMBER,
rep30      NUMBER,
rep31      NUMBER,
rep32      NUMBER,
rep33      NUMBER,
rep34      NUMBER,
rep35      NUMBER,
rep36      NUMBER,
rep37      NUMBER,
rep38      NUMBER,
rep39      NUMBER,
rep40      NUMBER,
rep41      NUMBER,
rep42      NUMBER,
rep43      NUMBER,
rep44      NUMBER,
rep45      NUMBER,
rep46      NUMBER,
rep47      NUMBER,
rep48      NUMBER,
rep49      NUMBER,
rep50      NUMBER,
rep51      NUMBER,
rep52      NUMBER,
rep53      NUMBER,
rep54      NUMBER,
rep55      NUMBER,
rep56      NUMBER,
rep57      NUMBER,
rep58      NUMBER,
rep59      NUMBER,
rep60      NUMBER,
rep61      NUMBER,
rep62      NUMBER,
rep63      NUMBER,
rep64      NUMBER,
rep65      NUMBER,
rep66      NUMBER,
rep67      NUMBER,
rep68      NUMBER,
rep69      NUMBER,
rep70      NUMBER,
rep71      NUMBER,
rep72      NUMBER,
rep73      NUMBER,
rep74      NUMBER,
rep75      NUMBER,
rep76      NUMBER,
rep77      NUMBER,
rep78      NUMBER,
rep79      NUMBER,
rep80      NUMBER,
rep81      NUMBER,
rep82      NUMBER,
rep83      NUMBER,
rep84      NUMBER,
rep85      NUMBER,
rep86      NUMBER,

rep87      NUMBER,
rep88      NUMBER,
rep89      NUMBER,
rep90      NUMBER,
rep91      NUMBER,
rep92      NUMBER,
rep93      NUMBER,
rep94      NUMBER,
rep95      NUMBER,
rep96      NUMBER,
rep97      NUMBER,
rep98      NUMBER,
rep99      NUMBER,
rep100     NUMBER,
thk1       NUMBER,
thk2       NUMBER,
thk3       NUMBER,
thk4       NUMBER,
thk5       NUMBER,
thk6       NUMBER,
thk7       NUMBER,
thk8       NUMBER,
thk9       NUMBER,
thk10      NUMBER,
thk11      NUMBER,
thk12      NUMBER,
thk13      NUMBER,
thk14      NUMBER,

```

```

thk15          NUMBER,
thk16          NUMBER,
thk17          NUMBER,
thk18          NUMBER,
thk19          NUMBER,
thk20          NUMBER,
thk21          NUMBER,
thk22          NUMBER,
thk23          NUMBER,
thk24          NUMBER,
thk25          NUMBER,
key1           NUMBER,
key2           NUMBER,
key3           NUMBER,
key4           NUMBER,
key5           NUMBER,
key6           NUMBER,
key7           NUMBER,
key8           NUMBER,
key9           NUMBER,
key10          NUMBER
);

rem
rem Results for payment transactions.
rem These results are from the measurement interval only.
rem
rem CREATE TABLE bench_pay_res
rem (
rem   run_name      VARCHAR2(20),
rem   audit_str    VARCHAR2(10),
rem   proc_no      NUMBER,
rem   rep1          NUMBER,
rem   rep2          NUMBER,
rem   rep3          NUMBER,
rem   rep4          NUMBER,
rem   rep5          NUMBER,
rem   rep6          NUMBER,
rem   rep7          NUMBER,
rem   rep8          NUMBER,
rem   rep9          NUMBER,
rem   rep10         NUMBER,
rem   rep11         NUMBER,
rem   rep12         NUMBER,
rem   rep13         NUMBER,
rem   rep14         NUMBER,
rem   rep15         NUMBER,
rem   rep16         NUMBER,
rem   rep17         NUMBER,
rem   rep18         NUMBER,
rem   rep19         NUMBER,
rem   rep20         NUMBER,
rem   rep21         NUMBER,
rem   rep22         NUMBER,
rem   rep23         NUMBER,
rem   rep24         NUMBER,
rem   rep25         NUMBER,
rem   rep26         NUMBER,
rem   rep27         NUMBER,
rem   rep28         NUMBER,
rem   rep29         NUMBER,
rem   rep30         NUMBER,
rem   rep31         NUMBER,
rem   rep32         NUMBER,
rem   rep33         NUMBER,
rem   rep34         NUMBER,
rem   rep35         NUMBER,
rem   rep36         NUMBER,
rem   rep37         NUMBER,
rem   rep38         NUMBER,
rem   rep39         NUMBER,
rem   rep40         NUMBER,
rem   rep41         NUMBER,
rem   rep42         NUMBER,
rem   rep43         NUMBER,
rem   rep44         NUMBER,
rem   rep45         NUMBER,
rem   rep46         NUMBER,
rem   rep47         NUMBER,
rem   rep48         NUMBER,
rem   rep49         NUMBER,
rem   rep50         NUMBER,
rem   rep51         NUMBER,
rem   rep52         NUMBER,
rem   rep53         NUMBER,
rem   rep54         NUMBER,
rem   rep55         NUMBER,
rem   rep56         NUMBER,
rem   rep57         NUMBER,
rem   rep58         NUMBER,
rem   rep59         NUMBER,
rem   rep60         NUMBER,
rem   rep61         NUMBER,
rem   rep62         NUMBER,
rem   rep63         NUMBER,
rem   rep64         NUMBER,
rem   rep65         NUMBER,
rem   rep66         NUMBER,
rem   rep67         NUMBER,
rem   rep68         NUMBER,
rem   rep69         NUMBER,

```

```

rep70          NUMBER,
rep71          NUMBER,
rep72          NUMBER,
rep73          NUMBER,
rep74          NUMBER,
rep75          NUMBER,
rep76          NUMBER,
rep77          NUMBER,
rep78          NUMBER,
rep79          NUMBER,
rep80          NUMBER,
rep81          NUMBER,
rep82          NUMBER,
rep83          NUMBER,
rep84          NUMBER,
rep85          NUMBER,
rep86          NUMBER,
rep87          NUMBER,
rep88          NUMBER,
rep89          NUMBER,
rep90          NUMBER,
rep91          NUMBER,
rep92          NUMBER,
rep93          NUMBER,
rep94          NUMBER,
rep95          NUMBER,
rep96          NUMBER,
rep97          NUMBER,
rep98          NUMBER,
rep99          NUMBER,
rep100         NUMBER,
thk1           NUMBER,
thk2           NUMBER,
thk3           NUMBER,
thk4           NUMBER,
thk5           NUMBER,
thk6           NUMBER,
thk7           NUMBER,
thk8           NUMBER,
thk9           NUMBER,
thk10          NUMBER,
thk11          NUMBER,
thk12          NUMBER,
thk13          NUMBER,
thk14          NUMBER,
thk15          NUMBER,
thk16          NUMBER,
thk17          NUMBER,
thk18          NUMBER,
thk19          NUMBER,
thk20          NUMBER,
thk21          NUMBER,
thk22          NUMBER,
thk23          NUMBER,
thk24          NUMBER,
thk25          NUMBER,
key1           NUMBER,
key2           NUMBER,
key3           NUMBER,
key4           NUMBER,
key5           NUMBER,
key6           NUMBER,
key7           NUMBER,
key8           NUMBER,
key9           NUMBER,
key10          NUMBER
);

rem
rem Aggregate results for order status transactions.
rem These results are from the measurement interval only.
rem
rem CREATE TABLE tpcc_ord_res
rem (
rem   run_name      VARCHAR2(20),
rem   repl         NUMBER,
rem   rep2         NUMBER,
rem   rep3         NUMBER,
rem   rep4         NUMBER,
rem   rep5         NUMBER,
rem   rep6         NUMBER,
rem   rep7         NUMBER,
rem   rep8         NUMBER,
rem   rep9         NUMBER,
rem   rep10        NUMBER,
rem   rep11        NUMBER,
rem   rep12        NUMBER,
rem   rep13        NUMBER,
rem   rep14        NUMBER,
rem   rep15        NUMBER,
rem   rep16        NUMBER,
rem   rep17        NUMBER,
rem   rep18        NUMBER,
rem   rep19        NUMBER,
rem   rep20        NUMBER,
rem   rep21        NUMBER,
rem   rep22        NUMBER,
rem   rep23        NUMBER,
rem   rep24        NUMBER,
rem   rep25        NUMBER,
rem   rep26        NUMBER,

```

```

rep27      NUMBER,
rep28      NUMBER,
rep29      NUMBER,
rep30      NUMBER,
rep31      NUMBER,
rep32      NUMBER,
rep33      NUMBER,
rep34      NUMBER,
rep35      NUMBER,
rep36      NUMBER,
rep37      NUMBER,
rep38      NUMBER,
rep39      NUMBER,
rep40      NUMBER,
rep41      NUMBER,
rep42      NUMBER,
rep43      NUMBER,
rep44      NUMBER,
rep45      NUMBER,
rep46      NUMBER,
rep47      NUMBER,
rep48      NUMBER,
rep49      NUMBER,
rep50      NUMBER,
rep51      NUMBER,
rep52      NUMBER,
rep53      NUMBER,
rep54      NUMBER,
rep55      NUMBER,
rep56      NUMBER,
rep57      NUMBER,

rep58      NUMBER,
rep59      NUMBER,
rep60      NUMBER,
rep61      NUMBER,
rep62      NUMBER,
rep63      NUMBER,
rep64      NUMBER,
rep65      NUMBER,
rep66      NUMBER,
rep67      NUMBER,
rep68      NUMBER,
rep69      NUMBER,
rep70      NUMBER,
rep71      NUMBER,
rep72      NUMBER,
rep73      NUMBER,
rep74      NUMBER,
rep75      NUMBER,
rep76      NUMBER,
rep77      NUMBER,
rep78      NUMBER,
rep79      NUMBER,
rep80      NUMBER,
rep81      NUMBER,
rep82      NUMBER,
rep83      NUMBER,
rep84      NUMBER,
rep85      NUMBER,
rep86      NUMBER,
rep87      NUMBER,
rep88      NUMBER,
rep89      NUMBER,
rep90      NUMBER,
rep91      NUMBER,
rep92      NUMBER,
rep93      NUMBER,
rep94      NUMBER,
rep95      NUMBER,
rep96      NUMBER,
rep97      NUMBER,
rep98      NUMBER,
rep99      NUMBER,
rep100     NUMBER,
thk1       NUMBER,
thk2       NUMBER,
thk3       NUMBER,
thk4       NUMBER,
thk5       NUMBER,
thk6       NUMBER,
thk7       NUMBER,
thk8       NUMBER,
thk9       NUMBER,
thk10      NUMBER,
thk11      NUMBER,
thk12      NUMBER,
thk13      NUMBER,
thk14      NUMBER,
thk15      NUMBER,
thk16      NUMBER,
thk17      NUMBER,
thk18      NUMBER,
thk19      NUMBER,
thk20      NUMBER,
thk21      NUMBER,
thk22      NUMBER,
thk23      NUMBER,
thk24      NUMBER,
thk25      NUMBER,
key1       NUMBER,

```

```

key2       NUMBER,
key3       NUMBER,
key4       NUMBER,
key5       NUMBER,
key6       NUMBER,
key7       NUMBER,
key8       NUMBER,
key9       NUMBER,
key10      NUMBER
);

rem
rem Results for order status transactions.
rem These results are from the measurement interval only.
rem
CREATE TABLE bench_ord_res
(
  run_name  VARCHAR2(20),
  audit_str VARCHAR2(10),
  proc_no   NUMBER,
  repl      NUMBER,
  rep2      NUMBER,
  rep3      NUMBER,
  rep4      NUMBER,
  rep5      NUMBER,
  rep6      NUMBER,
  rep7      NUMBER,
  rep8      NUMBER,
  rep9      NUMBER,
  rep10     NUMBER,
  rep11     NUMBER,
  rep12     NUMBER,
  rep13     NUMBER,
  rep14     NUMBER,
  rep15     NUMBER,
  rep16     NUMBER,
  rep17     NUMBER,
  rep18     NUMBER,
  rep19     NUMBER,
  rep20     NUMBER,
  rep21     NUMBER,
  rep22     NUMBER,
  rep23     NUMBER,
  rep24     NUMBER,
  rep25     NUMBER,
  rep26     NUMBER,
  rep27     NUMBER,
  rep28     NUMBER,
  rep29     NUMBER,
  rep30     NUMBER,
  rep31     NUMBER,
  rep32     NUMBER,
  rep33     NUMBER,
  rep34     NUMBER,
  rep35     NUMBER,
  rep36     NUMBER,
  rep37     NUMBER,
  rep38     NUMBER,
  rep39     NUMBER,
  rep40     NUMBER,
  rep41     NUMBER,
  rep42     NUMBER,
  rep43     NUMBER,
  rep44     NUMBER,
  rep45     NUMBER,
  rep46     NUMBER,
  rep47     NUMBER,
  rep48     NUMBER,
  rep49     NUMBER,
  rep50     NUMBER,

  rep51     NUMBER,
  rep52     NUMBER,
  rep53     NUMBER,
  rep54     NUMBER,
  rep55     NUMBER,
  rep56     NUMBER,
  rep57     NUMBER,
  rep58     NUMBER,
  rep59     NUMBER,
  rep60     NUMBER,
  rep61     NUMBER,
  rep62     NUMBER,
  rep63     NUMBER,
  rep64     NUMBER,
  rep65     NUMBER,
  rep66     NUMBER,
  rep67     NUMBER,
  rep68     NUMBER,
  rep69     NUMBER,
  rep70     NUMBER,
  rep71     NUMBER,
  rep72     NUMBER,
  rep73     NUMBER,
  rep74     NUMBER,
  rep75     NUMBER,
  rep76     NUMBER,
  rep77     NUMBER,
  rep78     NUMBER,
  rep79     NUMBER,
  rep80     NUMBER,

```

```

rep81      NUMBER,
rep82      NUMBER,
rep83      NUMBER,
rep84      NUMBER,
rep85      NUMBER,
rep86      NUMBER,
rep87      NUMBER,
rep88      NUMBER,
rep89      NUMBER,
rep90      NUMBER,
rep91      NUMBER,
rep92      NUMBER,
rep93      NUMBER,
rep94      NUMBER,
rep95      NUMBER,
rep96      NUMBER,
rep97      NUMBER,
rep98      NUMBER,
rep99      NUMBER,
rep100     NUMBER,
thk1       NUMBER,
thk2       NUMBER,
thk3       NUMBER,
thk4       NUMBER,
thk5       NUMBER,
thk6       NUMBER,
thk7       NUMBER,
thk8       NUMBER,
thk9       NUMBER,
thk10      NUMBER,
thk11      NUMBER,
thk12      NUMBER,
thk13      NUMBER,
thk14      NUMBER,
thk15      NUMBER,
thk16      NUMBER,
thk17      NUMBER,
thk18      NUMBER,
thk19      NUMBER,
thk20      NUMBER,
thk21      NUMBER,
thk22      NUMBER,
thk23      NUMBER,
thk24      NUMBER,
thk25      NUMBER,
key1       NUMBER,
key2       NUMBER,
key3       NUMBER,
key4       NUMBER,
key5       NUMBER,
key6       NUMBER,
key7       NUMBER,
key8       NUMBER,
key9       NUMBER,
key10      NUMBER
);

rem
rem Aggregate results for delivery transactions.
rem These results are from the measurement interval only.
rem
CREATE TABLE tpcc_del_res
(
  run_name      VARCHAR2(20),
  rep1          NUMBER,
  rep2          NUMBER,
  rep3          NUMBER,
  rep4          NUMBER,
  rep5          NUMBER,
  rep6          NUMBER,
  rep7          NUMBER,
  rep8          NUMBER,
  rep9          NUMBER,
  rep10         NUMBER,
  rep11         NUMBER,
  rep12         NUMBER,
  rep13         NUMBER,
  rep14         NUMBER,
  rep15         NUMBER,
  rep16         NUMBER,
  rep17         NUMBER,
  rep18         NUMBER,
  rep19         NUMBER,
  rep20         NUMBER,
  rep21         NUMBER,
  rep22         NUMBER,
  rep23         NUMBER,
  rep24         NUMBER,
  rep25         NUMBER,
  rep26         NUMBER,
  rep27         NUMBER,
  rep28         NUMBER,
  rep29         NUMBER,
  rep30         NUMBER,
  rep31         NUMBER,
  rep32         NUMBER,
  rep33         NUMBER,
  rep34         NUMBER,
  rep35         NUMBER,
  rep36         NUMBER,
  rep37         NUMBER,

```

```

rep38      NUMBER,
rep39      NUMBER,
rep40      NUMBER,
rep41      NUMBER,
rep42      NUMBER,
rep43      NUMBER,
rep44      NUMBER,
rep45      NUMBER,
rep46      NUMBER,
rep47      NUMBER,
rep48      NUMBER,
rep49      NUMBER,
rep50      NUMBER,
rep51      NUMBER,
rep52      NUMBER,
rep53      NUMBER,
rep54      NUMBER,
rep55      NUMBER,
rep56      NUMBER,
rep57      NUMBER,
rep58      NUMBER,
rep59      NUMBER,
rep60      NUMBER,
rep61      NUMBER,
rep62      NUMBER,
rep63      NUMBER,
rep64      NUMBER,
rep65      NUMBER,
rep66      NUMBER,
rep67      NUMBER,
rep68      NUMBER,
rep69      NUMBER,
rep70      NUMBER,
rep71      NUMBER,
rep72      NUMBER,
rep73      NUMBER,
rep74      NUMBER,
rep75      NUMBER,
rep76      NUMBER,
rep77      NUMBER,
rep78      NUMBER,
rep79      NUMBER,
rep80      NUMBER,
rep81      NUMBER,
rep82      NUMBER,
rep83      NUMBER,
rep84      NUMBER,
rep85      NUMBER,
rep86      NUMBER,
rep87      NUMBER,
rep88      NUMBER,
rep89      NUMBER,
rep90      NUMBER,
rep91      NUMBER,
rep92      NUMBER,
rep93      NUMBER,
rep94      NUMBER,
rep95      NUMBER,
rep96      NUMBER,
rep97      NUMBER,
rep98      NUMBER,
rep99      NUMBER,
rep100     NUMBER,
thk1       NUMBER,
thk2       NUMBER,
thk3       NUMBER,
thk4       NUMBER,
thk5       NUMBER,
thk6       NUMBER,
thk7       NUMBER,
thk8       NUMBER,
thk9       NUMBER,
thk10      NUMBER,
thk11      NUMBER,
thk12      NUMBER,
thk13      NUMBER,
thk14      NUMBER,
thk15      NUMBER,
thk16      NUMBER,
thk17      NUMBER,
thk18      NUMBER,
thk19      NUMBER,
thk20      NUMBER,
thk21      NUMBER,
thk22      NUMBER,
thk23      NUMBER,
thk24      NUMBER,
thk25      NUMBER,
key1       NUMBER,
key2       NUMBER,
key3       NUMBER,
key4       NUMBER,
key5       NUMBER,
key6       NUMBER,
key7       NUMBER,
key8       NUMBER,
key9       NUMBER,
key10      NUMBER
);

rem

```

```

rem Results for delivery transactions.
rem These results are from the measurement interval only.
rem

```

```

CREATE TABLE bench_del_res
(
  run_name      VARCHAR2(20),
  audit_str     VARCHAR2(10),
  proc_no       NUMBER,
  rep1          NUMBER,
  rep2          NUMBER,
  rep3          NUMBER,
  rep4          NUMBER,
  rep5          NUMBER,
  rep6          NUMBER,
  rep7          NUMBER,
  rep8          NUMBER,
  rep9          NUMBER,
  rep10         NUMBER,
  rep11         NUMBER,
  rep12         NUMBER,
  rep13         NUMBER,
  rep14         NUMBER,
  rep15         NUMBER,
  rep16         NUMBER,
  rep17         NUMBER,
  rep18         NUMBER,
  rep19         NUMBER,
  rep20         NUMBER,
  rep21         NUMBER,
  rep22         NUMBER,
  rep23         NUMBER,
  rep24         NUMBER,
  rep25         NUMBER,
  rep26         NUMBER,
  rep27         NUMBER,
  rep28         NUMBER,
  rep29         NUMBER,
  rep30         NUMBER,
  rep31         NUMBER,
  rep32         NUMBER,
  rep33         NUMBER,
  rep34         NUMBER,
  rep35         NUMBER,
  rep36         NUMBER,
  rep37         NUMBER,
  rep38         NUMBER,
  rep39         NUMBER,
  rep40         NUMBER,
  rep41         NUMBER,
  rep42         NUMBER,
  rep43         NUMBER,
  rep44         NUMBER,
  rep45         NUMBER,
  rep46         NUMBER,
  rep47         NUMBER,
  rep48         NUMBER,
  rep49         NUMBER,
  rep50         NUMBER,
  rep51         NUMBER,
  rep52         NUMBER,
  rep53         NUMBER,
  rep54         NUMBER,
  rep55         NUMBER,
  rep56         NUMBER,
  rep57         NUMBER,
  rep58         NUMBER,
  rep59         NUMBER,
  rep60         NUMBER,
  rep61         NUMBER,
  rep62         NUMBER,
  rep63         NUMBER,
  rep64         NUMBER,
  rep65         NUMBER,
  rep66         NUMBER,
  rep67         NUMBER,
  rep68         NUMBER,
  rep69         NUMBER,
  rep70         NUMBER,
  rep71         NUMBER,
  rep72         NUMBER,
  rep73         NUMBER,
  rep74         NUMBER,
  rep75         NUMBER,
  rep76         NUMBER,
  rep77         NUMBER,
  rep78         NUMBER,
  rep79         NUMBER,
  rep80         NUMBER,
  rep81         NUMBER,
  rep82         NUMBER,
  rep83         NUMBER,
  rep84         NUMBER,
  rep85         NUMBER,
  rep86         NUMBER,
  rep87         NUMBER,
  rep88         NUMBER,
  rep89         NUMBER,
  rep90         NUMBER,
  rep91         NUMBER,
  rep92         NUMBER,

```

```

  rep93         NUMBER,
  rep94         NUMBER,
  rep95         NUMBER,
  rep96         NUMBER,
  rep97         NUMBER,
  rep98         NUMBER,
  rep99         NUMBER,
  rep100        NUMBER,
  thk1          NUMBER,
  thk2          NUMBER,
  thk3          NUMBER,
  thk4          NUMBER,
  thk5          NUMBER,
  thk6          NUMBER,
  thk7          NUMBER,
  thk8          NUMBER,
  thk9          NUMBER,
  thk10         NUMBER,
  thk11         NUMBER,
  thk12         NUMBER,
  thk13         NUMBER,
  thk14         NUMBER,
  thk15         NUMBER,
  thk16         NUMBER,
  thk17         NUMBER,
  thk18         NUMBER,
  thk19         NUMBER,
  thk20         NUMBER,
  thk21         NUMBER,
  thk22         NUMBER,
  thk23         NUMBER,
  thk24         NUMBER,
  thk25         NUMBER,
  key1          NUMBER,
  key2          NUMBER,
  key3          NUMBER,
  key4          NUMBER,
  key5          NUMBER,
  key6          NUMBER,
  key7          NUMBER,
  key8          NUMBER,
  key9          NUMBER,
  key10         NUMBER
);

```

```

rem
rem Aggregate results for stock level transactions.
rem These results are from the measurement interval only.
rem

```

```

CREATE TABLE tpcc_sto_res
(
  run_name      VARCHAR2(20),
  rep1          NUMBER,
  rep2          NUMBER,
  rep3          NUMBER,
  rep4          NUMBER,
  rep5          NUMBER,
  rep6          NUMBER,
  rep7          NUMBER,
  rep8          NUMBER,
  rep9          NUMBER,
  rep10         NUMBER,
  rep11         NUMBER,
  rep12         NUMBER,
  rep13         NUMBER,
  rep14         NUMBER,
  rep15         NUMBER,
  rep16         NUMBER,
  rep17         NUMBER,
  rep18         NUMBER,
  rep19         NUMBER,
  rep20         NUMBER,
  rep21         NUMBER,
  rep22         NUMBER,
  rep23         NUMBER,
  rep24         NUMBER,
  rep25         NUMBER,
  rep26         NUMBER,
  rep27         NUMBER,
  rep28         NUMBER,
  rep29         NUMBER,
  rep30         NUMBER,
  rep31         NUMBER,
  rep32         NUMBER,
  rep33         NUMBER,
  rep34         NUMBER,
  rep35         NUMBER,
  rep36         NUMBER,
  rep37         NUMBER,
  rep38         NUMBER,
  rep39         NUMBER,
  rep40         NUMBER,
  rep41         NUMBER,
  rep42         NUMBER,
  rep43         NUMBER,
  rep44         NUMBER,
  rep45         NUMBER,
  rep46         NUMBER,
  rep47         NUMBER,
  rep48         NUMBER,
  rep49         NUMBER,

```

```

rep50      NUMBER,
rep51      NUMBER,
rep52      NUMBER,
rep53      NUMBER,
rep54      NUMBER,
rep55      NUMBER,
rep56      NUMBER,
rep57      NUMBER,
rep58      NUMBER,
rep59      NUMBER,
rep60      NUMBER,
rep61      NUMBER,
rep62      NUMBER,
rep63      NUMBER,
rep64      NUMBER,
rep65      NUMBER,
rep66      NUMBER,
rep67      NUMBER,
rep68      NUMBER,
rep69      NUMBER,
rep70      NUMBER,
rep71      NUMBER,
rep72      NUMBER,
rep73      NUMBER,
rep74      NUMBER,
rep75      NUMBER,
rep76      NUMBER,
rep77      NUMBER,
rep78      NUMBER,
rep79      NUMBER,
rep80      NUMBER,
rep81      NUMBER,
rep82      NUMBER,
rep83      NUMBER,
rep84      NUMBER,
rep85      NUMBER,
rep86      NUMBER,
rep87      NUMBER,
rep88      NUMBER,
rep89      NUMBER,
rep90      NUMBER,
rep91      NUMBER,
rep92      NUMBER,
rep93      NUMBER,
rep94      NUMBER,
rep95      NUMBER,
rep96      NUMBER,
rep97      NUMBER,
rep98      NUMBER,
rep99      NUMBER,
rep100     NUMBER,
thk1       NUMBER,
thk2       NUMBER,
thk3       NUMBER,
thk4       NUMBER,
thk5       NUMBER,
thk6       NUMBER,
thk7       NUMBER,
thk8       NUMBER,
thk9       NUMBER,
thk10      NUMBER,
thk11      NUMBER,
thk12      NUMBER,
thk13      NUMBER,
thk14      NUMBER,
thk15      NUMBER,
thk16      NUMBER,
thk17      NUMBER,
thk18      NUMBER,
thk19      NUMBER,
thk20      NUMBER,
thk21      NUMBER,
thk22      NUMBER,
thk23      NUMBER,
thk24      NUMBER,
thk25      NUMBER,
key1       NUMBER,
key2       NUMBER,
key3       NUMBER,
key4       NUMBER,
key5       NUMBER,
key6       NUMBER,
key7       NUMBER,
key8       NUMBER,
key9       NUMBER,
key10      NUMBER
);

rem
rem Results for stock level transactions.
rem These results are from the measurement interval only.
rem
CREATE TABLE bench_sto_res
(
  run_name      VARCHAR2(20),
  audit_str     VARCHAR2(10),
  proc_no      NUMBER,
  rep1         NUMBER,
  rep2         NUMBER,
  rep3         NUMBER,
  rep4         NUMBER,

```

```

rep5       NUMBER,
rep6       NUMBER,
rep7       NUMBER,
rep8       NUMBER,
rep9       NUMBER,
rep10      NUMBER,
rep11      NUMBER,
rep12      NUMBER,
rep13      NUMBER,
rep14      NUMBER,
rep15      NUMBER,
rep16      NUMBER,
rep17      NUMBER,
rep18      NUMBER,
rep19      NUMBER,
rep20      NUMBER,
rep21      NUMBER,
rep22      NUMBER,
rep23      NUMBER,
rep24      NUMBER,
rep25      NUMBER,
rep26      NUMBER,
rep27      NUMBER,
rep28      NUMBER,
rep29      NUMBER,
rep30      NUMBER,
rep31      NUMBER,
rep32      NUMBER,
rep33      NUMBER,
rep34      NUMBER,
rep35      NUMBER,
rep36      NUMBER,
rep37      NUMBER,
rep38      NUMBER,
rep39      NUMBER,
rep40      NUMBER,
rep41      NUMBER,
rep42      NUMBER,
rep43      NUMBER,
rep44      NUMBER,
rep45      NUMBER,
rep46      NUMBER,
rep47      NUMBER,
rep48      NUMBER,
rep49      NUMBER,
rep50      NUMBER,
rep51      NUMBER,
rep52      NUMBER,
rep53      NUMBER,
rep54      NUMBER,
rep55      NUMBER,
rep56      NUMBER,
rep57      NUMBER,
rep58      NUMBER,
rep59      NUMBER,
rep60      NUMBER,
rep61      NUMBER,
rep62      NUMBER,
rep63      NUMBER,
rep64      NUMBER,
rep65      NUMBER,
rep66      NUMBER,
rep67      NUMBER,
rep68      NUMBER,
rep69      NUMBER,
rep70      NUMBER,
rep71      NUMBER,
rep72      NUMBER,
rep73      NUMBER,
rep74      NUMBER,
rep75      NUMBER,
rep76      NUMBER,
rep77      NUMBER,
rep78      NUMBER,
rep79      NUMBER,
rep80      NUMBER,
rep81      NUMBER,
rep82      NUMBER,
rep83      NUMBER,
rep84      NUMBER,
rep85      NUMBER,
rep86      NUMBER,
rep87      NUMBER,
rep88      NUMBER,
rep89      NUMBER,
rep90      NUMBER,
rep91      NUMBER,
rep92      NUMBER,
rep93      NUMBER,
rep94      NUMBER,
rep95      NUMBER,
rep96      NUMBER,
rep97      NUMBER,
rep98      NUMBER,
rep99      NUMBER,
rep100     NUMBER,
thk1       NUMBER,
thk2       NUMBER,
thk3       NUMBER,
thk4       NUMBER,
thk5       NUMBER,

```



```

thk6          NUMBER,
thk7          NUMBER,
thk8          NUMBER,
thk9          NUMBER,
thk10         NUMBER,
thk11         NUMBER,
thk12         NUMBER,
thk13         NUMBER,
thk14         NUMBER,
thk15         NUMBER,
thk16         NUMBER,
thk17         NUMBER,
thk18         NUMBER,
thk19         NUMBER,
thk20         NUMBER,

thk21         NUMBER,
thk22         NUMBER,
thk23         NUMBER,
thk24         NUMBER,
thk25         NUMBER,
key1          NUMBER,
key2          NUMBER,
key3          NUMBER,
key4          NUMBER,
key5          NUMBER,
key6          NUMBER,
key7          NUMBER,
key8          NUMBER,
key9          NUMBER,
key10         NUMBER
);
commit;
set echo off;
rem spool off;
rem exit;

*****
cre_tab.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          cre_tab.sql
rem DESCRIPTION
rem          Create temporary tables for consistency tests.
=====
rem
rem Usage:  sqlplus tpcc/tpcc @cre_tab
rem

connect tpcc/tpcc;
set echo on;

drop table temp_o1;
drop table temp_no;
drop table temp_o2;
drop table temp_ol;
drop table tpcc_audit_tab;

create table temp_o1 (
  o_w_id integer,
  o_d_id integer,
  o_o_id integer);

create table temp_no (
  no_w_id integer,
  no_d_id integer,
  no_o_id integer);

create table temp_o2 (
  o_w_id integer,
  o_d_id integer,
  o_count integer);

create table temp_ol (
  ol_w_id integer,
  ol_d_id integer,
  ol_count integer);

create table tpcc_audit_tab (starttime date);

delete from tpcc_audit_tab;

set echo off;

*****

```

```

create_cache_views.sql
*****

rem This script creates four views that when queried will return
rem the total number of buffers in the buffer cache and the total
rem number of cloned buffers from each of the database's
rem tablespaces.
rem
rem This assumes that each table and index is in its own
rem tablespace.
rem If this is not the case, another query can be used which uses
rem the
rem database's object tables to decipher the different objects.
rem However,
rem this query is slower.
rem
rem This script assumes 7.3.x. If you are using V7.2.x or below,
rem please
rem replace svrmgr1 with sqldb lmode=y.
rem
rem Modification History:
rem
rem wblattist      16-Jun-1996      Create two additional views to
rem keep                                                  track of the number of clones in
rem each                                                  tablespace.
rem
rem wblattist      24-May-1995      Add the state check for the cbf
rem view                                                  to ensure that cloned blocks are
rem not                                                  counted.
rem

connect internal/internal;
set echo on;
drop view cbf;
create view cbf as
  select distinct(dbarfil) file#, count(1) blocks
  from x$bh
  where dbarfil > 0 and state <> 3
  group by dbarfil;
drop view ccbt;
create view ccbt as
  select ts$.name name,sum(cbf.blocks) buffers
  from cbf, file$, ts$
  where cbf.file#=file$.file# and file$.ts#=ts$.ts#
  group by file$.ts#, ts$.name;
drop view ccbfcln;
create view ccbfcln as
  select distinct(dbarfil) file#, count(1) blocks
  from x$bh
  where dbarfil > 0
  group by dbarfil;
drop view ccbtcln;
create view ccbtcln as
  select ts$.name name,sum(ccbfcln.blocks) buffers
  from ccbfcln, file$, ts$
  where ccbfcln.file#=file$.file# and file$.ts#=ts$.ts#
  group by file$.ts#, ts$.name;

set echo off;

*****
createcust.sql
*****

spool log/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 345600000
  hash is (c_w_id * 30000 + c_id * 10 + c_d_id - 30011)
  size      850
  initrans 3
  pctfree 0
  storage ( initial 1440002k next 1440000k buffer_pool recycle
  freelists 10 freelist groups 4 maxextents 250 pctincrease 0)
  tablespace cust_0;

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;

*****
createdist.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 115200
size 3072
hash is (d_w_id - 1) * 10 + d_id
initrans 4
pctfree 0
storage ( initial 57602k next 57600k 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;

*****
createhist.sql
*****

spool log/step9createhist.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)
)
initrans 4
pctfree 5
partition by range (h_w_id)
(
partition hist1 values less than (1441) tablespace hist_1,
partition hist2 values less than (2881) tablespace hist_2,
partition hist3 values less than (4321) tablespace hist_3,
partition hist4 values less than (5761) tablespace hist_4,

```

```

partition hist5 values less than (7201) tablespace hist_5,
partition hist6 values less than (8641) tablespace hist_6,
partition hist7 values less than (10081) tablespace hist_7,
partition hist8 values less than (MAXVALUE) tablespace hist_8
)
;
spool off;
set echo off;
exit sql.sqlcode;

*****
createicust1.sql
*****

spool log/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
pctfree 1
parallel 8
storage ( freelists 22 freelist groups 43 )
tablespace indexes;
spool off;
set echo off;
exit sql.sqlcode;

*****
createicust2.sql
*****

spool log/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
parallel 8
storage ( freelists 22 freelist groups 43 )
tablespace icust2_0;
spool off;
set echo off;
exit sql.sqlcode;

*****
createidist.sql
*****

spool log/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 4
pctfree 1
storage ( freelists 22 freelist groups 43 )
tablespace indexes
;
spool off;
set echo off;
exit sql.sqlcode;

*****
createiitem.sql
*****

spool log/step31createiitem.log;
set echo on;
drop index iitem;

set timing on;

create unique index iitem on item (i_id)
initrans 4
pctfree 1
storage ( freelists 22 freelist groups 43 )
tablespace item_0;
spool off;
set echo off;
exit sql.sqlcode;

*****
createiordr1.sql
*****

spool log/step35createiordr1.log;

```

```

set echo on;
drop index iordr1;

set timing on;

create unique index iordr1 on ord (o_w_id, o_d_id, o_id)
  intrans 3
  parallel 4
  pctfree 1
  storage ( freelists 22 freelist groups 43 )
  local
  (
    partition iordr11 tablespace iordr1_1,
    partition iordr12 tablespace iordr1_2,
    partition iordr13 tablespace iordr1_3,
    partition iordr14 tablespace iordr1_4,
    partition iordr15 tablespace iordr1_5,
    partition iordr16 tablespace iordr1_6,
    partition iordr17 tablespace iordr1_7,
    partition iordr18 tablespace iordr1_8
  )
;
spool off;
set echo off;
exit sql.sqlcode;

*****
createiordr2.sql
*****

spool log/step36createiordr2.log;
set echo on;
drop index iordr2;

set timing on;

create unique index iordr2 on ord (o_w_id, o_d_id, o_c_id, o_id)
  intrans 4
  parallel 4
  pctfree 25
  storage ( freelists 22 freelist groups 43 )
  local
  (
    partition iordr21 tablespace iordr2_1,
    partition iordr22 tablespace iordr2_2,
    partition iordr23 tablespace iordr2_3,
    partition iordr24 tablespace iordr2_4,
    partition iordr25 tablespace iordr2_5,
    partition iordr26 tablespace iordr2_6,
    partition iordr27 tablespace iordr2_7,
    partition iordr28 tablespace iordr2_8
  )
;
spool off;
set echo off;
exit sql.sqlcode;

*****
createistok.sql
*****

spool log/createistok.log;
set echo on;
drop index istok;

set timing on;

create unique index istok on stok (s_i_id, s_w_id)
  intrans 3
  parallel 16
  pctfree 1
  storage ( freelists 22 freelist groups 43 )
  tablespace indexes;
spool off;
set echo off;
exit sql.sqlcode;

*****
createiware.sql
*****

spool log/step29createiware.log;
set echo on;
drop index iware;

set timing on;

create unique index iware on ware (w_id)
  intrans 3
  parallel 4
  pctfree 1
  storage ( freelists 22 freelist groups 43 )
  tablespace indexes
;
spool off;
set echo off;
exit sql.sqlcode;

```

```

*****
createnord.sql
*****

spool log/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
  intrans 4
  pctfree 5
  storage ( freelists 22 freelist groups 43 )
  partition by range (no_w_id)
(
  partition nord1 values less than (1441) tablespace nord_1,
  partition nord2 values less than (2881) tablespace nord_2,
  partition nord3 values less than (4321) tablespace nord_3,
  partition nord4 values less than (5761) tablespace nord_4,
  partition nord5 values less than (7201) tablespace nord_5,
  partition nord6 values less than (8641) tablespace nord_6,
  partition nord7 values less than (10081) tablespace nord_7,
  partition nord8 values less than (MAXVALUE) tablespace nord_8
)
;
spool off;
set echo off;
exit sql.sqlcode;

*****
createordl.sql
*****

spool log/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
  intrans 4
  pctfree 5
  storage ( freelists 22 freelist groups 43 )
  partition by range (ol_w_id)
(
  partition ordl1 values less than (1441) tablespace ordl_1,
  partition ordl2 values less than (2881) tablespace ordl_2,
  partition ordl3 values less than (4321) tablespace ordl_3,
  partition ordl4 values less than (5761) tablespace ordl_4,
  partition ordl5 values less than (7201) tablespace ordl_5,
  partition ordl6 values less than (8641) tablespace ordl_6,
  partition ordl7 values less than (10081) tablespace ordl_7,
  partition ordl8 values less than (MAXVALUE) tablespace ordl_8
)
;
spool off;
set echo off;
exit sql.sqlcode;

*****
createordr.sql
*****

spool log/step9createordr.log;
set echo on;
drop table ord;

set timing on;

create table ord(
  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
    )
    initrans    4
    pctfree    5
    partition by range (o_w_id)
    (
    partition ordr1 values less than (1441) tablespace ordr_1,
    partition ordr2 values less than (2881) tablespace ordr_2,
    partition ordr3 values less than (4321) tablespace ordr_3,
    partition ordr4 values less than (5761) tablespace ordr_4,
    partition ordr5 values less than (7201) tablespace ordr_5,
    partition ordr6 values less than (8641) tablespace ordr_6,
    partition ordr7 values less than (10081) tablespace ordr_7,
    partition ordr8 values less than (MAXVALUE) tablespace ordr_8
    )
    ;
    spool off;
    set echo off;
    exit sql.sqlcode;

*****
createstok.sql
*****

spool log/step16createstok.log;
set echo 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    1152000000
    hash is    (abs(s_i_id - 1) * 1440 + mod((s_w_id - 1), 1440) +
    trunc ((s_w_id - 1) / 1440) * 144000000)
    size    380
    initrans    3
    pctfree    0
    storage ( initial 1920002k next 1920000k buffer_pool keep
    freelists 10 freelist groups 4 maxextents 250 pctincrease 0)
    tablespace    stok_0;

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;

*****
createts.sh
*****

#!/bin/sh
sqlplus tpcc/tpcc @<<!
    create tablespace cust_0 datafile '/home/oracle/dev/raw/cust-1'
    size 14100M reuse extent management dictionary;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-2'
    size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-3'
    size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-4'
    size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-5'
    size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-6'
    size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-7'
    size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-8'
    size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-9'
    size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-
    10' size 14100M reuse;

```

```

    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-
    11' size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-
    12' size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-
    13' size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-
    14' size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-
    15' size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-
    16' size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-
    17' size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-
    18' size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-
    19' size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-
    20' size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-
    21' size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-
    22' size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-
    23' size 14100M reuse;
    alter tablespace cust_0 add datafile '/home/oracle/dev/raw/cust-
    24' size 15600M reuse;

create tablespace stok_0 datafile '/home/oracle/dev/raw/stock-1'
    size 9400M reuse extent management dictionary;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-2'
    size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-3'
    size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-4'
    size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-5'
    size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-6'
    size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-7'
    size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-8'
    size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-9'
    size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    10' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    11' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    12' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    13' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    14' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    15' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    16' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    17' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    18' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    19' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    20' size 9400M reuse;

    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    21' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    22' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    23' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    24' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    25' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    26' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    27' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    28' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    29' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    30' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    31' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    32' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    33' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    34' size 9400M reuse;
    alter tablespace stok_0 add datafile '/home/oracle/dev/raw/stock-
    35' size 9400M reuse;

```



```

single table
hashkeys 11520
size 3072
hash is w_id - 1
initrans 3
pctfree 0
storage (initial 5762k next 5760k pctincrease 0 freelist groups
4)
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;

connect internal/internal;

```

```

*****
droprollsegs.sql
*****

```

```

set echo on

drop rollback segment t1;
drop rollback segment t2;
drop rollback segment t3;
drop rollback segment t4;
drop rollback segment t5;
drop rollback segment t6;
drop rollback segment t7;
drop rollback segment t8;
drop rollback segment t9;
drop rollback segment t10;
drop rollback segment t11;
drop rollback segment t12;
drop rollback segment t13;
drop rollback segment t14;
drop rollback segment t15;
drop rollback segment t16;
drop rollback segment t17;
drop rollback segment t18;
drop rollback segment t19;
drop rollback segment t20;
drop rollback segment t21;
drop rollback segment t22;
drop rollback segment t23;
drop rollback segment t24;
drop rollback segment t25;

```

```

drop rollback segment t26;
drop rollback segment t27;
drop rollback segment t28;
drop rollback segment t29;
drop rollback segment t30;

```

```

spool off
set echo off
exit sql.sqlcode

```

```

*****
init_1.ora
*****

```

```

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)
ifile=$HOME/p_run.ora

```

```

*****
init_2.ora
*****

```

```

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)
ifile=$HOME/p_run.ora

```

```

*****
init_3.ora
*****

```

```

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)
ifile=$HOME/p_run.ora

```

```

*****
init_4.ora
*****

```

```

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)
ifile=$HOME/p_run.ora

```

```

*****
init_5.ora
*****

```

```

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)
ifile=$HOME/p_run.ora

```

```

*****
init_6.ora
*****

```

```

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)
ifile=$HOME/p_run.ora

```

```

*****
init_7.ora
*****

```

```

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)
ifile=$HOME/p_run.ora

```

```

*****
init_8.ora
*****

```

```

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)
ifile=$HOME/p_run.ora

```

```

*****
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;
idxlarr 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');
idxlarr := idxarr;
END new_init;
END initnew;
/
show errors

```

```

*****
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;
/
show errors;

```

```

CREATE OR REPLACE PACKAGE BODY initpay AS
PROCEDURE pay_init IS
BEGIN
NULL;
END pay_init;
END initpay;
/
show errors;

```

```

*****
loadcust.sh
*****

```

```

#!/bin/sh
./tpccload.exe -M 11520 -c -b 1 -e 1440 >log/loadcust.log0 2>&1
&
./tpccload.exe -M 11520 -c -b 1441 -e 2880 >log/loadcust.log1 2>&1
&
./tpccload.exe -M 11520 -c -b 2881 -e 4320 >log/loadcust.log2 2>&1
&
./tpccload.exe -M 11520 -c -b 4321 -e 5760 >log/loadcust.log3 2>&1
&
./tpccload.exe -M 11520 -c -b 5761 -e 7200 >log/loadcust.log4 2>&1
&
./tpccload.exe -M 11520 -c -b 7201 -e 8640 >log/loadcust.log5 2>&1
&
./tpccload.exe -M 11520 -c -b 8641 -e 10080 >log/loadcust.log6
2>&1 &
./tpccload.exe -M 11520 -c -b 10081 -e 11520 >log/loadcust.log7
2>&1 &
wait

```

```

*****
loadhist.sh
*****

```

```

#!/bin/sh
./tpccload.exe -M 11520 -h -b 1 -e 1440 >log/loadhist.log0 2>&1
&
./tpccload.exe -M 11520 -h -b 1441 -e 2880 >log/loadhist.log1 2>&1
&
./tpccload.exe -M 11520 -h -b 2881 -e 4320 >log/loadhist.log2 2>&1
&
./tpccload.exe -M 11520 -h -b 4321 -e 5760 >log/loadhist.log3 2>&1
&
./tpccload.exe -M 11520 -h -b 5761 -e 7200 >log/loadhist.log4 2>&1
&
./tpccload.exe -M 11520 -h -b 7201 -e 8640 >log/loadhist.log5 2>&1
&
./tpccload.exe -M 11520 -h -b 8641 -e 10080 >log/loadhist.log6
2>&1 &
./tpccload.exe -M 11520 -h -b 10081 -e 11520 >log/loadhist.log7
2>&1 &
wait

```

```

*****
loadnord.sh
*****

```

```

#!/bin/sh
./tpccload.exe -M 11520 -n -b 1 -e 1440 >log/loadnord.log0 2>&1
&
./tpccload.exe -M 11520 -n -b 1441 -e 2880 >log/loadnord.log1 2>&1
&
./tpccload.exe -M 11520 -n -b 2881 -e 4320 >log/loadnord.log2 2>&1
&
./tpccload.exe -M 11520 -n -b 4321 -e 5760 >log/loadnord.log3 2>&1
&
./tpccload.exe -M 11520 -n -b 5761 -e 7200 >log/loadnord.log4 2>&1
&
./tpccload.exe -M 11520 -n -b 7201 -e 8640 >log/loadnord.log5 2>&1
&
./tpccload.exe -M 11520 -n -b 8641 -e 10080 >log/loadnord.log6
2>&1 &
./tpccload.exe -M 11520 -n -b 10081 -e 11520 >log/loadnord.log7
2>&1 &
wait

```

```

*****
loadordrordl.sh
*****

```

```

#!/bin/sh

```

```

./tpccload.exe -M 11520 -o templ -b 1 -e 1440
>log/loadordrordl.log0 2>&1 &
./tpccload.exe -M 11520 -o templ -b 1441 -e 2880
>log/loadordrordl.log1 2>&1 &
./tpccload.exe -M 11520 -o templ -b 2881 -e 4320
>log/loadordrordl.log2 2>&1 &
./tpccload.exe -M 11520 -o templ -b 4321 -e 5760
>log/loadordrordl.log3 2>&1 &
./tpccload.exe -M 11520 -o templ -b 5761 -e 7200
>log/loadordrordl.log4 2>&1 &
./tpccload.exe -M 11520 -o templ -b 7201 -e 8640
>log/loadordrordl.log5 2>&1 &
./tpccload.exe -M 11520 -o templ -b 8641 -e 10080
>log/loadordrordl.log6 2>&1 &
./tpccload.exe -M 11520 -o templ -b 10081 -e 11520
>log/loadordrordl.log7 2>&1 &
wait

*****
loadstok-1.sh
*****

./tpccload.exe -M 11520 -S -j 1 -k 25000 -b 1 -e 1440 >
log/loadstok1_1.log 2>&1 &
./tpccload.exe -M 11520 -S -j 25001 -k 50000 -b 1 -e 1440 >
log/loadstok1_2.log 2>&1 &
./tpccload.exe -M 11520 -S -j 50001 -k 75000 -b 1 -e 1440 >
log/loadstok1_3.log 2>&1 &
./tpccload.exe -M 11520 -S -j 75001 -k 100000 -b 1 -e 1440 >
log/loadstok1_4.log 2>&1 &
wait

*****
loadstok-2.sh
*****

./tpccload.exe -M 11520 -S -j 1 -k 25000 -b 1441 -e 2880 >
log/loadstok2_1.log 2>&1 &
./tpccload.exe -M 11520 -S -j 25001 -k 50000 -b 1441 -e 2880 >
log/loadstok2_2.log 2>&1 &
./tpccload.exe -M 11520 -S -j 50001 -k 75000 -b 1441 -e 2880 >
log/loadstok2_3.log 2>&1 &
./tpccload.exe -M 11520 -S -j 75001 -k 100000 -b 1441 -e 2880 >
log/loadstok2_4.log 2>&1 &
wait

*****
loadstok-3.sh
*****

./tpccload.exe -M 11520 -S -j 1 -k 25000 -b 2881 -e 4320 >
log/loadstok3_1.log 2>&1 &
./tpccload.exe -M 11520 -S -j 25001 -k 50000 -b 2881 -e 4320 >
log/loadstok3_2.log 2>&1 &
./tpccload.exe -M 11520 -S -j 50001 -k 75000 -b 2881 -e 4320 >
log/loadstok3_3.log 2>&1 &
./tpccload.exe -M 11520 -S -j 75001 -k 100000 -b 2881 -e 4320 >
log/loadstok3_4.log 2>&1 &
wait

*****
loadstok-4.sh
*****

./tpccload.exe -M 11520 -S -j 1 -k 25000 -b 4321 -e 5760 >
log/loadstok4_1.log 2>&1 &
./tpccload.exe -M 11520 -S -j 25001 -k 50000 -b 4321 -e 5760 >
log/loadstok4_2.log 2>&1 &
./tpccload.exe -M 11520 -S -j 50001 -k 75000 -b 4321 -e 5760 >
log/loadstok4_3.log 2>&1 &
./tpccload.exe -M 11520 -S -j 75001 -k 100000 -b 4321 -e 5760 >
log/loadstok4_4.log 2>&1 &
wait

*****
loadstok-5.sh
*****

./tpccload.exe -M 11520 -S -j 1 -k 25000 -b 5761 -e 7200 >
log/loadstok5_1.log 2>&1 &
./tpccload.exe -M 11520 -S -j 25001 -k 50000 -b 5761 -e 7200 >
log/loadstok5_2.log 2>&1 &
./tpccload.exe -M 11520 -S -j 50001 -k 75000 -b 5761 -e 7200 >
log/loadstok5_3.log 2>&1 &
./tpccload.exe -M 11520 -S -j 75001 -k 100000 -b 5761 -e 7200 >
log/loadstok5_4.log 2>&1 &
wait

*****
loadstok-6.sh
*****

./tpccload.exe -M 11520 -S -j 1 -k 25000 -b 7201 -e 8640 >
log/loadstok6_1.log 2>&1 &

```

```

./tpccload.exe -M 11520 -S -j 25001 -k 50000 -b 7201 -e 8640 >
log/loadstok6_2.log 2>&1 &
./tpccload.exe -M 11520 -S -j 50001 -k 75000 -b 7201 -e 8640 >
log/loadstok6_3.log 2>&1 &
./tpccload.exe -M 11520 -S -j 75001 -k 100000 -b 7201 -e 8640 >
log/loadstok6_4.log 2>&1 &
wait

*****
loadstok-7.sh
*****

./tpccload.exe -M 11520 -S -j 1 -k 25000 -b 8641 -e 10080 >
log/loadstok7_1.log 2>&1 &
./tpccload.exe -M 11520 -S -j 25001 -k 50000 -b 8641 -e 10080 >
log/loadstok7_2.log 2>&1 &
./tpccload.exe -M 11520 -S -j 50001 -k 75000 -b 8641 -e 10080 >
log/loadstok7_3.log 2>&1 &
./tpccload.exe -M 11520 -S -j 75001 -k 100000 -b 8641 -e 10080 >
log/loadstok7_4.log 2>&1 &
wait

*****
loadstok-8.sh
*****

./tpccload.exe -M 11520 -S -j 1 -k 25000 -b 10081 -e 11520 >
log/loadstok8_1.log 2>&1 &
./tpccload.exe -M 11520 -S -j 25001 -k 50000 -b 10081 -e 11520 >
log/loadstok8_2.log 2>&1 &
./tpccload.exe -M 11520 -S -j 50001 -k 75000 -b 10081 -e 11520 >
log/loadstok8_3.log 2>&1 &
./tpccload.exe -M 11520 -S -j 75001 -k 100000 -b 10081 -e 11520 >
log/loadstok8_4.log 2>&1 &
wait

*****
orst_cre.sql
*****

rem
rem
rem
=====
rem          Copyright (c) 1996 Oracle Corp, Redwood Shores, CA
|
rem          OPEN SYSTEMS PERFORMANCE GROUP
|
rem          All Rights Reserved
|
rem
=====
rem FILENAME
rem      orst_cre.sql
rem DESCRIPTION
rem      Drop and Create Tables for Oracle Statistics
rem
=====
/
rem
rem Usage: sqlplus internal/internal @orst_cre.sql
rem
rem          SET ECHO ON;
rem          SET TERMOUT OFF;

rem          DROP TABLE save_sysstat;
rem          DROP TABLE save_latch;
rem          DROP TABLE save_rollstat;
rem          DROP TABLE save_filestat;
rem          DROP TABLE save_rowcache;
rem          DROP TABLE save_parameter;
rem          DROP TABLE save_wait;
rem          DROP TABLE save_fwait;
rem          DROP TABLE save_event;
rem          DROP TABLE save_lockact;
rem          DROP TABLE save_fping;
rem          DROP TABLE save_fping2;
rem          DROP TABLE save_ping;
rem          DROP TABLE save_ping2;
rem          DROP TABLE save_blkping;
rem          DROP TABLE save_blkping2;
rem          DROP TABLE save_kclwait;
rem          DROP TABLE save_sqlarea;
rem          DROP TABLE save_time;
rem          DROP TABLE save_dfile;
rem          DROP TABLE save_rsrc;
rem          DROP TABLE save_circuit;
rem          DROP TABLE save_dispatcher;
rem          DROP TABLE save_queue;
rem          DROP TABLE save_server;

rem          DROP TABLE tmp_sysstat;
rem          DROP TABLE tmp_latch;
rem          DROP TABLE tmp_filestat;
rem          DROP TABLE tmp_rollstat;
rem          DROP TABLE tmp_rowcache;
rem          DROP TABLE tmp_wait;

```



```

DROP TABLE tmp_fwait;
DROP TABLE tmp_event;
DROP TABLE tmp_fping;
DROP TABLE tmp_fping2;
DROP TABLE tmp_kclwait;
DROP TABLE blockclass;
DROP TABLE tmp_lockact;
DROP TABLE zero_lockact;
DROP TABLE tmp_sqlarea;
DROP TABLE tmp_time;
DROP TABLE tmp_circuit;
DROP TABLE tmp_dispatcher;
DROP TABLE tmp_queue;
DROP TABLE tmp_server;
DROP TABLE tmp_rsrc;

rem
rem save_sysstat corresponds to v$sysstat and v$statname
rem
CREATE TABLE save_sysstat
(
    hid          NUMBER,
    run          NUMBER,
    name         VARCHAR2(64),
    value        NUMBER,
    statistic#   NUMBER
);

rem
rem save_latch corresponds to v$latch and v$latchname
rem
CREATE TABLE save_latch
(
    hid          NUMBER,
    run          NUMBER,
    name         VARCHAR2(64),
    latch#       NUMBER,
    gets         NUMBER,
    misses       NUMBER,
    sleeps       NUMBER,
    immediate_gets NUMBER,
    immediate_misses NUMBER
);

rem
rem save_rollstat corresponds to v$rollstat and v$rollname
rem
CREATE TABLE save_rollstat
(
    hid          NUMBER,
    run          NUMBER,
    name         VARCHAR2(30),
    USN          NUMBER,
    EXTENTS      NUMBER,
    RSSIZE       NUMBER,
    WRITES       NUMBER,
    XACTS        NUMBER,
    GETS         NUMBER,
    WAITS        NUMBER,
    OPTSIZE      NUMBER,
    HWM SIZE     NUMBER,
    SHRINKS      NUMBER,
    WRAPS        NUMBER,
    EXTENDS      NUMBER,
    AVESHINK     NUMBER,
    AVEACTIVE    NUMBER
);

rem
rem save_filestat corresponds to v$filestat and v$dbfile;
rem
CREATE TABLE save_filestat
(
    hid          NUMBER,
    run          NUMBER,
    FILE#        NUMBER,
    PHYRDS       NUMBER,
    PHYWRTS     NUMBER,
    PHYBLKRD    NUMBER,
    PHYBLKWRT   NUMBER,
    READTIM     NUMBER,
    WRITETIM    NUMBER,
    NAME         VARCHAR2(257)
);

rem
rem save_rowcache corresponds to v$rowcache
rem
CREATE TABLE save_rowcache
(
    hid          NUMBER,
    run          NUMBER,
    cache#       NUMBER,
    type         VARCHAR2(11),
    subordinate# NUMBER,
    parameter    VARCHAR2(32),
    count        NUMBER,
    usage        NUMBER,
    fixed        NUMBER,

```

```

    gets         NUMBER,
    getmisses   NUMBER,
    scans        NUMBER,
    scanmisses  NUMBER,
    scancompletes NUMBER,
    modifications NUMBER,
    flushes     NUMBER
);

rem
rem Create table to hold values in v$parameter
rem
CREATE TABLE save_parameter
(
    hid          NUMBER,
    run          NUMBER,
    NAME         VARCHAR2(64),
    VALUE        VARCHAR2(512)
);

rem
rem save_wait corresponds to v$wait_stat
rem
CREATE TABLE save_wait
(
    hid          NUMBER,
    run          NUMBER,
    class        VARCHAR2(18),
    count        NUMBER,
    time         NUMBER
);

rem
rem save_fwai corresponds to X$KCBFWAIT
rem
CREATE TABLE save_fwai
(
    hid          NUMBER,
    run          NUMBER,
    addr         VARCHAR2(20),
    indx         NUMBER,
    count        NUMBER,
    time         NUMBER
);

rem
rem save_event corresponds to v$system_event
rem
CREATE TABLE save_event
(
    hid          NUMBER,
    run          NUMBER,
    event        VARCHAR2(64),
    total_waits  NUMBER,
    time_waited NUMBER,
    average_wait NUMBER
);

rem
rem save_lockact corresponds to v$lock_activity
rem
CREATE TABLE save_lockact
(
    hid          NUMBER,
    run          NUMBER,
    from_val     VARCHAR2(4),
    to_val       VARCHAR2(4),
    action_val   VARCHAR2(51),
    counter      NUMBER
);

rem
rem save_fping corresponds to file_ping
rem
CREATE TABLE save_fping
(
    hid          NUMBER,
    run          NUMBER,
    file_id      NUMBER,
    file_name    VARCHAR2(257),
    ts_name      VARCHAR2(30),
    x_to_n       NUMBER
);

rem
rem save_fping2 corresponds to file_ping with extended ping stats
rem
CREATE TABLE save_fping2
(
    hid          NUMBER,
    run          NUMBER,
    file_id      NUMBER,
    file_name    VARCHAR2(257),
    ts_name      VARCHAR2(30),
    x2n          NUMBER,
    x2s          NUMBER,
    x2ssx        NUMBER,
    s2n          NUMBER,
    c1c          NUMBER,
    crt          NUMBER,

```

```

        hping          NUMBER,
        sping          NUMBER
    );

rem
rem save_ping corresponds to v$ping
rem
    CREATE TABLE save_ping
    (
        hid            NUMBER,
        run            NUMBER,
        tablespace_name VARCHAR2(30),
        file_name      VARCHAR2(257),
        kind           VARCHAR2(12),
        status         VARCHAR2(4),
        xnc            NUMBER
    );

rem
rem save_ping2 corresponds to v$ping with extended ping stats
rem
    CREATE TABLE save_ping2
    (
        hid            NUMBER,
        run            NUMBER,
        tablespace_name VARCHAR2(30),
        file#          NUMBER,
        kind           VARCHAR2(12),
        status         VARCHAR2(4),
        hping          NUMBER,
        sping          NUMBER
    );

rem
rem save_blkping corresponds to v$ping
rem
    CREATE TABLE save_blkping
    (
        hid            NUMBER,
        run            NUMBER,
        tablespace_name VARCHAR2(30),
        file_name      VARCHAR2(257),
        kind           VARCHAR2(12),
        block#         NUMBER,
        status         VARCHAR2(4),
        xnc            NUMBER
    );

rem
rem save_blkping2 corresponds to v$ping with extended ping stats
rem
    CREATE TABLE save_blkping2
    (
        hid            NUMBER,
        run            NUMBER,
        tablespace_name VARCHAR2(30),
        file#          NUMBER,
        kind           VARCHAR2(12),
        block#         NUMBER,
        status         VARCHAR2(4),
        hping          NUMBER,
        sping          NUMBER,
        lock_element_addr RAW(4)
    );

rem
rem save_kclwait corresponds to v$sklwait
rem
    CREATE TABLE save_kclwait
    (
        hid            NUMBER,
        run            NUMBER,
        indx           NUMBER,
        pings          NUMBER,
        hpings         NUMBER,
        spings         NUMBER,
        wpings         NUMBER
    );

rem
rem save_sqlarea corresponds to v$sqlarea
rem
    CREATE TABLE save_sqlarea
    (
        hid            NUMBER,
        run            NUMBER,
        sql_text       VARCHAR2(1000),
        executions     NUMBER,
        buffer_gets    NUMBER,
        disk_reads     NUMBER,
        serializable_aborts NUMBER
    );

rem
rem save_time records duration of each run
rem
    CREATE TABLE save_time
    (
        hid            NUMBER,
        run            NUMBER,
        rtime          NUMBER

```

```

    );

rem
rem save_dfile maps oracle datafile to physical disks and nodes
rem
    CREATE TABLE save_dfile
    (
        file#          NUMBER,
        group#         NUMBER,
        gname          VARCHAR2(20),
        node#          NUMBER,
        nname          VARCHAR2(20),
        disk#          NUMBER,
        dname          VARCHAR2(20)
    );

rem
rem save_rsrc corresponds to v$rsrc_consumer_group
rem
    CREATE TABLE save_rsrc
    (
        hid            NUMBER,
        run            NUMBER,
        NAME           VARCHAR2(32),
        ACTIVE_SESSIONS NUMBER,
        EXECUTION_WAITERS NUMBER,
        REQUESTS       NUMBER,
        CPU_WAIT_TIME  NUMBER,
        CPU_WAITS      NUMBER,
        CONSUMED_CPU_TIME NUMBER,
        YIELDS         NUMBER,
        SESSIONS_QUEUED NUMBER
    );

    CREATE TABLE save_circuit
    (
        hid            NUMBER,
        run            NUMBER,
        circuit        raw(4),
        msg0           NUMBER,
        msg1           NUMBER,
        msgs           NUMBER,
        bytes          NUMBER,
        breaks         NUMBER
    );

    CREATE TABLE save_dispatcher
    (
        hid            NUMBER,
        run            NUMBER,
        paddr          raw(4),
        msgs           NUMBER,
        bytes          NUMBER,
        breaks         NUMBER,
        idle           NUMBER,
        busy           NUMBER
    );

    CREATE TABLE save_server
    (
        hid            NUMBER,
        run            NUMBER,
        name           VARCHAR2(20),
        msgs           NUMBER,
        bytes          NUMBER,
        breaks         NUMBER,
        idle           NUMBER,
        busy           NUMBER,
        requests       NUMBER
    );

    CREATE TABLE save_queue
    (
        hid            NUMBER,
        run            NUMBER,
        paddr          raw(4),
        wait           NUMBER,
        totalq         NUMBER
    );

rem
rem tmp_sysstat corresponds to v$sysstat
rem
    CREATE TABLE tmp_sysstat
    (
        hid            NUMBER,
        state          VARCHAR2(10),
        statistic#     NUMBER,
        value           NUMBER
    );

rem
rem tmp_latch corresponds to v$latch
rem
    CREATE TABLE tmp_latch
    (
        hid            NUMBER,
        state          VARCHAR2(10),
        latch#         NUMBER,
        gets           NUMBER,

```

```

misses      NUMBER,
sleeps      NUMBER,
immediate_gets NUMBER,
immediate_misses NUMBER
);

rem
rem tmp_filestat corresponds to v$filestat
rem
CREATE TABLE tmp_filestat
(
  hid NUMBER,
  state VARCHAR2(10),
  FILE# NUMBER,
  PHYRDS NUMBER,
  PHYWRTS NUMBER,
  PHYBLKRD NUMBER,
  PHYBLKWRT NUMBER,
  READTIM NUMBER,
  WRITETIM NUMBER
);

rem
rem tmp_rollstat corresponds to v$rollstat
rem
CREATE TABLE tmp_rollstat
(
  hid NUMBER,
  state VARCHAR2(10),
  USN NUMBER,
  EXTENTS NUMBER,
  RSSIZE NUMBER,
  WRITES NUMBER,
  XACTS NUMBER,
  GETS NUMBER,
  WAITS NUMBER,
  OPTSIZE NUMBER,
  HWMSIZE NUMBER,
  SHRINKS NUMBER,
  WRAPS NUMBER,
  EXTENDS NUMBER,
  AVESHINK NUMBER,
  AVEACTIVE NUMBER
);

rem
rem tmp_rowcache corresponds to v$rowcache
rem
CREATE TABLE tmp_rowcache
(
  hid NUMBER,
  state VARCHAR2(10),
  cache# NUMBER,
  type VARCHAR2(11),
  subordinate# NUMBER,
  parameter VARCHAR2(32),
  count NUMBER,
  usage NUMBER,
  fixed NUMBER,
  gets NUMBER,
  getmisses NUMBER,
  scans NUMBER,
  scanmisses NUMBER,
  scancompletes NUMBER,
  modifications NUMBER,
  flushes NUMBER
);

rem
rem tmp_wait corresponds to v$wait_stat
rem
CREATE TABLE tmp_wait
(
  hid NUMBER,
  state VARCHAR2(10),
  class VARCHAR2(18),
  count NUMBER,
  time NUMBER
);

rem
rem tmp_fwait corresponds to X$KCBFWAIT
rem
CREATE TABLE tmp_fwait
(
  hid NUMBER,
  state VARCHAR2(10),
  addr VARCHAR2(20),
  indx NUMBER,
  count NUMBER,
  time NUMBER
);

rem
rem tmp_event corresponds to v$system_event
rem
CREATE TABLE tmp_event
(
  hid NUMBER,
  state VARCHAR2(10),
  event VARCHAR2(64),

```

```

total_waits NUMBER,
time_waited NUMBER,
average_wait NUMBER
);

rem
rem tmp_fping corresponds to file_ping
rem
CREATE TABLE tmp_fping
(
  hid NUMBER,
  state VARCHAR2(10),
  file_id NUMBER,
  file_name VARCHAR2(257),
  ts_name VARCHAR2(30),
  x_to_n NUMBER
);

rem
rem tmp_fping2 corresponds to file_ping with extended ping stats
rem
CREATE TABLE tmp_fping2
(
  hid NUMBER,
  state VARCHAR2(10),
  file_id NUMBER,
  file_name VARCHAR2(257),
  ts_name VARCHAR2(30),
  x2n NUMBER,
  x2s NUMBER,
  x2ssx NUMBER,
  s2n NUMBER,
  cic NUMBER,
  crt NUMBER,
  hping NUMBER,
  sping NUMBER
);

rem
rem tmp_kclwait corresponds to v$kclwait
rem
CREATE TABLE tmp_kclwait
(
  hid NUMBER,
  state VARCHAR2(10),
  indx NUMBER,
  pings NUMBER,
  hpings NUMBER,
  spings NUMBER,
  wpings NUMBER
);

rem
rem blockclass contains mapping from KCBC index to text
rem
CREATE TABLE blockclass
(
  indx NUMBER,
  name VARCHAR2(24)
);

INSERT INTO blockclass VALUES (0, 'NONE');
INSERT INTO blockclass VALUES (1, 'DATA');
INSERT INTO blockclass VALUES (2, 'SORT');
INSERT INTO blockclass VALUES (3, 'SAVE UNDO');
INSERT INTO blockclass VALUES (4, 'SEG HDR');
INSERT INTO blockclass VALUES (5, 'SAVE UNDO SEG HDR');
INSERT INTO blockclass VALUES (6, 'FREE LIST|EXTENT MAP');
INSERT INTO blockclass VALUES (7, 'UNDO HDR');
INSERT INTO blockclass VALUES (8, 'UNDO');
INSERT INTO blockclass VALUES (9, 'UNDO HDR');
INSERT INTO blockclass VALUES (10, 'UNDO');

rem Are there more...???

rem
rem tmp_lockact corresponds to v$lock_activity
rem
CREATE TABLE tmp_lockact
(
  hid NUMBER,
  state VARCHAR2(10),
  from_val VARCHAR2(4),
  to_val VARCHAR2(4),
  action_val VARCHAR2(51),
  counter NUMBER
);

rem
rem zero_lockact corresponds to v$lock_activity with no activity
rem
CREATE TABLE zero_lockact
(
  from_val VARCHAR2(4),
  to_val VARCHAR2(4),
  action_val VARCHAR2(51),
  counter NUMBER
);

INSERT INTO zero_lockact (from_val, to_val, action_val, counter)
VALUES

```

```

('NULL', 'S', 'Lock buffers for read',
0);

INSERT INTO zero_lockact (from_val, to_val, action_val, counter)
VALUES
('NULL', 'X', 'Lock buffers for write',
0);

INSERT INTO zero_lockact (from_val, to_val, action_val, counter)
VALUES
('S', 'NULL', 'Make buffers CR (no write)',
0);

INSERT INTO zero_lockact (from_val, to_val, action_val, counter)
VALUES
('S', 'X', 'Upgrade read lock to write',
0);

INSERT INTO zero_lockact (from_val, to_val, action_val, counter)
VALUES
('X', 'NULL', 'Make buffers CR (write dirty buffers)',
0);

INSERT INTO zero_lockact (from_val, to_val, action_val, counter)
VALUES
('X', 'S', 'Downgrade write lock to read (write dirty
buffers)', 0);

INSERT INTO zero_lockact (from_val, to_val, action_val, counter)
VALUES
('X', 'SSX', 'Write transaction table/undo blocks',
0);

INSERT INTO zero_lockact (from_val, to_val, action_val, counter)
VALUES
('SSX', 'NULL', 'Transaction table/undo blocks (write dirty
buffers)', 0);

INSERT INTO zero_lockact (from_val, to_val, action_val, counter)
VALUES
('SSX', 'S', 'Make transaction table/undo block available
share', 0);

INSERT INTO zero_lockact (from_val, to_val, action_val, counter)
VALUES
('SSX', 'X', 'Rearm transaction table write mechanism',
0);

rem
rem tmp_sqlarea corresponds to v$sqlarea
rem
CREATE TABLE tmp_sqlarea
(
hid NUMBER,
state VARCHAR2(10),
sql_text VARCHAR2(1000),
executions NUMBER,
buffer_gets NUMBER,
disk_reads NUMBER,
serializable_aborts NUMBER
);

rem
rem tmp_time records begin and end time
rem
CREATE TABLE tmp_time
(
hid NUMBER,
state VARCHAR2(10),
timestamp DATE
);

CREATE TABLE tmp_circuit
(
hid NUMBER,
state VARCHAR2(10),
circuit raw(4),
msg0 NUMBER,
msg1 NUMBER,
msgs NUMBER,
bytes NUMBER,
breaks NUMBER
);

CREATE TABLE tmp_dispatcher
(
hid NUMBER,
state VARCHAR2(10),
paddr raw(4),
msgs NUMBER,
bytes NUMBER,
breaks NUMBER,
idle NUMBER,
busy NUMBER
);

CREATE TABLE tmp_server
(
hid NUMBER,
state VARCHAR2(10),
name VARCHAR2(20),

```

```

msgs NUMBER,
bytes NUMBER,
breaks NUMBER,
idle NUMBER,
busy NUMBER,
requests NUMBER
);

CREATE TABLE tmp_queue
(
hid NUMBER,
state VARCHAR2(10),
paddr raw(4),
wait NUMBER,
totalq NUMBER
);

rem
rem tmp_rsrc corresponds to v$rsrc_consumer_group
rem
CREATE TABLE tmp_rsrc
(
hid NUMBER,
state VARCHAR2(10),
NAME VARCHAR2(32),
ACTIVE_SESSIONS NUMBER,
EXECUTION_WAITERS NUMBER,
REQUESTS NUMBER,
CPU_WAIT_TIME NUMBER,
CPU_WAITS NUMBER,
CONSUMED_CPU_TIME NUMBER,
YIELDS NUMBER,
SESSIONS_QUEUED NUMBER
);

COMMIT;
SET ECHO OFF;

*****
p_recover.ora
*****

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)

compatible = 9.2.0.0.0
db_name = tpcc
control_files = /home/oracle/dev/raw/control-001
sort_area_size = 10485760
recovery_parallelism = 4
db_files = 250
use_indirect_data_buffers = TRUE
db_block_buffers = 3235840
dml_locks = 2000
log_buffer = 1048576
shared_pool_size = 160M
cursor_space_for_time = TRUE
db_block_size = 4096
_allocate_creation_order = TRUE
cluster_database = FALSE
processes = 64

log_parallelism = 2

parallel_max_servers = 10

*****
p_run.ora
*****

#
# 9i RAC 4K
#
#
_db_block_lru_latches = 64
_db_writer_max_writes = 1024
_db_writer_chunk_writes = 200
db_writer_processes = 4
db_cache_advice = off
compatible = 9.2.0.0.0
db_name = tpcc
control_files = /home/oracle/dev/raw/control-001
sort_area_size = 10485760
parallel_max_servers = 0
parallel_min_servers = 0
recovery_parallelism = 0
log_buffer = 10485760
db_block_size = 4096
_allocate_creation_order =TRUE

```

```

db_files=250
cursor_space_for_time          = TRUE
shared_pool_size = 160M
java_pool_size = 15728640
use_indirect_data_buffers = TRUE #pse36
db_block_buffers = 3235840
_log_simultaneous_copies = 16
dml_locks                      = 60
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         = 80
pre_page_sga                   = TRUE
processes                      = 100
sessions                       = 150
open Cursors                   = 180
replication_dependency_tracking = FALSE
transactions_per_rollback_segment = 1
timed_statistics               = FALSE
transaction_auditing           = FALSE
db_block_checking              = FALSE
cluster_database               = TRUE
cluster_database_instances     = 8
_kc1_name_table_latches = 64
_lm_lms=1
statistics_level = basic
log_parallelism = 2
_lm_dd_interval                = 30
gc_files_to_locks              = "1-8=1000EACH:\
9=1:\
10-12=100000:\
13-15=100000:\
16-18=100000:\
19-21=100000:\
22-24=100000:\
25-27=100000:\
28-30=100000:\
31-33=100000:\
34-39=350000:\
40-45=350000:\
46-51=350000:\
52-57=350000:\
58-63=350000:\
64-69=350000:\
70-75=350000:\
76-81=350000:\
82-195=1EACH:\
197-199=1EACH"
_lm_file_affinity="9,10-12,34-39,82,90,99-
101,123,131,139,172,180=1:\
13-15,40-45,83,91,102-104,124,132,140,173,181,188=2:\
16-18,46-51,84,92,105-107,125,133,141,174,182,189=3:\
19-21,52-57,85,93,108-110,126,134,142,175,183,190=4:\
22-24,58-63,86,94,111-113,127,135,143,176,184,191=5:\
25-27,64-69,87,95,114-116,128,136,144,177,185,192=6:\
28-30,70-75,88,96,117-119,129,137,145,178,186,193=7:\
31-33,76-81,89,97,120-122,130,138,146,179,187,184=8"
_gcs_resources=400000
_pcm_shadow_locks=400000
_interconnect_checksum = FALSE
db_block_checksum = FALSE

```

```

*****
plssql_mon.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    plssql_mon.sql
rem DESCRIPTION
rem    SQL script to create a stored package for PL/SQL stored
rem    procedures to dump messages.
rem
=====
rem Usage:  sqlplus tpcc/tpcc @plssql_mon
rem

connect tpcc/tpcc;
set echo on;
CREATE OR REPLACE PACKAGE plssql_mon_pack
IS
  PROCEDURE print
  (
    info          VARCHAR2
  );

```

```

END;
/
show errors;

CREATE OR REPLACE PACKAGE BODY plssql_mon_pack
IS
  PROCEDURE print
  (
    info          VARCHAR2
  )
  IS
    s              NUMBER;
  BEGIN
    dbms_pipe.pack_message (info);
    s := dbms_pipe.send_message ('plssql_mon');
    IF (s <> 0) THEN
      raise_application_error (-20000, 'Error:' || to_char(s) ||
        ' sending on pipe');
    END IF;
  END;
END;
/
show errors;

set echo off;

*****
pst_c.sql
*****

rem
rem
rem
=====
rem          Copyright (c) 1992 Oracle Corp, Belmont, CA
|
rem          OPEN SYSTEMS PERFORMANCE GROUP
|
rem          All Rights Reserved
|
rem
=====
rem FILENAME
rem    pst_c.sql
rem DESCRIPTION
rem    Create Table for OS Specific Process Stats
rem
=====
/
rem
rem Tables for Unix-specific process statistics
rem
rem Usage:  sqlplus internal/internal @pst_c
rem

connect tpcc/tpcc;
set echo on;
DROP TABLE proc_resource;
DROP TABLE os_stat;

rem
rem Resource usage for a process.
rem

CREATE TABLE proc_resource
(
  config          VARCHAR2(10),
  run             NUMBER,
  proc            NUMBER,
  child           NUMBER,
  user_cpu_ms     NUMBER,
  system_cpu_ms   NUMBER,
  maxrss         NUMBER,
  pagein         NUMBER,
  reclaim        NUMBER,
  zerofill       NUMBER,
  pffincr        NUMBER,
  pffdecr        NUMBER,
  swap           NUMBER,
  syscall        NUMBER,
  volcsw         NUMBER,
  involcsw       NUMBER,
  signal         NUMBER,
  lread          NUMBER,
  lwrite         NUMBER,
  bread          NUMBER,
  bwrite         NUMBER,
  phread         NUMBER,
  phwrite        NUMBER
);

rem
rem OS statistics.
rem These results are from the measurement interval only.
rem

CREATE TABLE os_stat
(
  config          VARCHAR2(10),
  run             NUMBER,

```



```

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 43 )
  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;

```

```

*****
step20loadware.sh
*****

```

```

#!sh
$TPCCLOAD -M 11520 -w > step20loadware.log 2>&1

if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi

```

```

*****
step21loadaddist.sh
*****

```

```

#!sh
$TPCCLOAD -M 11520 -d > step21loadaddist.log 2>&1

if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi

```

```

*****
step22loaditem.sh
*****

```

```

#!sh
P
$TPCCLOAD -M 11520 -i > step22loaditem.log 2>&1

if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi

```

```

*****
step2createdb.sql
*****

```

```
spool log/step2createdb.log
```

```
set echo on
```

```

startup pfile=p_create.ora nomount
create database tpcc
  controlfile reuse
  maxdatafiles 731
  maxinstances 8
  datafile '/home/oracle/dev/raw/system-1' size 96M reuse,
           '/home/oracle/dev/raw/system-2' size 96M reuse,
           '/home/oracle/dev/raw/system-3' size 96M reuse,
           '/home/oracle/dev/raw/system-4' size 96M reuse,
           '/home/oracle/dev/raw/system-5' size 96M reuse,
           '/home/oracle/dev/raw/system-6' size 96M reuse,
           '/home/oracle/dev/raw/system-7' size 96M reuse,
           '/home/oracle/dev/raw/system-8' size 96M reuse
  logfile group 1 ('/home/oracle/dev/raw/log-1-1') size 5717M
  reuse,

```

```

           group 2 ('/home/oracle/dev/raw/log-1-2') size 5717M reuse
  undo tablespace undo_1 datafile '/home/oracle/dev/raw/roll-1'
  size 500M reuse;

```

```

alter database add logfile thread 2
  group 3 ('/home/oracle/dev/raw/log-2-1') size 5717M
  reuse,
  group 4 ('/home/oracle/dev/raw/log-2-2') size 5717M
  reuse;
alter database enable public thread 2;

```

```

alter database add logfile thread 3
  group 5 ('/home/oracle/dev/raw/log-3-1') size 5717M
  reuse,
  group 6 ('/home/oracle/dev/raw/log-3-2') size 5717M
  reuse;
alter database enable public thread 3;

```

```

alter database add logfile thread 4
  group 7 ('/home/oracle/dev/raw/log-4-1') size 5717M
  reuse,
  group 8 ('/home/oracle/dev/raw/log-4-2') size 5717M
  reuse;
alter database enable public thread 4;

```

```

alter database add logfile thread 5
  group 9 ('/home/oracle/dev/raw/log-5-1') size 5717M
  reuse,
  group 10 ('/home/oracle/dev/raw/log-5-2') size 5717M
  reuse;
alter database enable public thread 5;

```

```

alter database add logfile thread 6
  group 11 ('/home/oracle/dev/raw/log-6-1') size 5717M
  reuse,
  group 12 ('/home/oracle/dev/raw/log-6-2') size 5717M
  reuse;
alter database enable public thread 6;

```

```

alter database add logfile thread 7
  group 13 ('/home/oracle/dev/raw/log-7-1') size 5717M
  reuse,
  group 14 ('/home/oracle/dev/raw/log-7-2') size 5717M
  reuse;
alter database enable public thread 7;

```

```

alter database add logfile thread 8
  group 15 ('/home/oracle/dev/raw/log-8-1') size 5717M
  reuse,
  group 16 ('/home/oracle/dev/raw/log-8-2') size 5717M
  reuse;
alter database enable public thread 8;

```

```

spool off
set echo off
exit sql.sqlcode

```

```

*****
step40createstats.sql
*****

```

```

spool log/step40createstats.log
@orst_cre
@c_stat
@pst_c
spool off
exit sql.sqlcode;

```

```

*****
step41createstoreprocs.sql
*****

```

```

spool log/step41createstoreprocs.log
@initpay
@initnew
spool off
exit sql.sqlcode;

```

```

*****
step43createmisc.sql
*****

```

```

spool log/step43createmisc.log
set echo on;
alter user tpcc temporary tablespace system;
grant execute on dbms_lock to public;
grant execute on dbms_pipe to public;
grant select on v_$parameter to public;
@plsqli_mon
@cre_tab
@create_cache_views
@views
@dml
@extent 2048
@freeext 2048
spool off
exit sql.sqlcode;

```

```

*****
step6createddviews.sql
*****

spool log/step6createddviews.log
@$ORACLE_HOME/rdbms/admin/catalog
@$ORACLE_HOME/rdbms/admin/catproc
@$ORACLE_HOME/rdbms/admin/catparr
spool off
exit sql.sqlcode;

```

```

*****
stepcreateuser.sql
*****

```

```

spool log/stepcreateuser.log;

```

```

set echo on;

```

```

create user tpcc identified by tpcc;

```

```

grant dba to tpcc;

```

```

set echo off;

```

```

spool off;

```

```

exit sql.sqlcode;

```

```

*****
views.sql
*****

```

```

connect tpcc/tpcc;
set echo on;

```

```

create or replace view wh_cust
(w_id, w_tax, c_id, c_d_id, c_w_id, c_discount, c_last, c_credit)
as select w.w_id, w.w_tax,
        c.c_id, c.c_d_id, c.c_w_id, c.c_discount, c.c_last,
        c.c_credit
   from cust c, ware w
   where w.w_id = c.c_w_id;

```

```

create or replace view wh_dist
(w_id, d_id, d_tax, d_next_o_id, w_tax )
as select w.w_id, d.d_id, d.d_tax, d.d_next_o_id, w.w_tax
   from dist d, ware w
   where w.w_id = d.d_w_id;

```

```

create or replace view stock_item
(i_id, s_w_id, i_price, i_name, i_data, s_data, s_quantity,
s_order_cnt, s_ytd, s_remote_cnt,
s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10)
as
select i.i_id, s.w_id, i.i_price, i.i_name, i.i_data, s_data,
s_quantity,
s_order_cnt, s_ytd, s_remote_cnt,
s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10
   from stok s, item i
   where i.i_id = s.s_i_id;

```

```

set echo off;

```

Appendix C: Tunable Parameters

SEQUENCE OF EVENTS FOR PERFORMANCE RUN

1. Boot up computers (clients, servers, & RTEs)
2. Run tcp_tune.sh on clients and database servers
3. Run tune_elv.sh on database servers.
4. Startup the database on all nodes.
5. Start apache on the clients.
6. Start tuxedo on the clients.
7. Run realtime.sh on the databases.
8. Start the RTE.

DATABASE SERVERS OS TUNABLES

```
*****  
mkraw.sh  
*****
```

```
raw /home/oracle/dev/raw/log-1-1 /dev/sda1  
  
raw /home/oracle/dev/raw/log-1-2 /dev/sda2  
raw /home/oracle/dev/raw/stock-1 /dev/sdc1  
raw /home/oracle/dev/raw/stock-2 /dev/sdc2  
raw /home/oracle/dev/raw/stock-3 /dev/sde1  
raw /home/oracle/dev/raw/stock-4 /dev/sde2  
raw /home/oracle/dev/raw/stock-5 /dev/sdg1  
raw /home/oracle/dev/raw/stock-6 /dev/sdg2  
raw /home/oracle/dev/raw/ordl-1-1 /dev/sdc3  
raw /home/oracle/dev/raw/ordl-1-2 /dev/sde3  
raw /home/oracle/dev/raw/ordl-1-3 /dev/sdg3  
raw /home/oracle/dev/raw/cust-1 /dev/sdc10  
raw /home/oracle/dev/raw/cust-2 /dev/sde10  
raw /home/oracle/dev/raw/cust-3 /dev/sdg10  
raw /home/oracle/dev/raw/nord-1 /dev/sdc5  
raw /home/oracle/dev/raw/hist-1 /dev/sde5  
raw /home/oracle/dev/raw/ordr-1 /dev/sdg5  
raw /home/oracle/dev/raw/icust2-1 /dev/sdc6  
raw /home/oracle/dev/raw/icust2-2 /dev/sde6  
raw /home/oracle/dev/raw/icust2-3 /dev/sdg6  
raw /home/oracle/dev/raw/ware-1 /dev/sdc7  
raw /home/oracle/dev/raw/dist-1 /dev/sde7  
raw /home/oracle/dev/raw/system-1 /dev/sdg7  
raw /home/oracle/dev/raw/roll-1 /dev/sdc8  
raw /home/oracle/dev/raw/iordr1-1 /dev/sde8  
raw /home/oracle/dev/raw/iordr2-1 /dev/sdg8  
raw /home/oracle/dev/raw/control-001 /dev/sdc9  
raw /home/oracle/dev/raw/temp-1 /dev/sde9  
raw /home/oracle/dev/raw/temp-2 /dev/sdg9  
  
raw /home/oracle/dev/raw/log-2-1 /dev/sdi1  
raw /home/oracle/dev/raw/log-2-2 /dev/sdi2  
raw /home/oracle/dev/raw/stock-7 /dev/sdk1  
raw /home/oracle/dev/raw/stock-8 /dev/sdk2  
raw /home/oracle/dev/raw/stock-9 /dev/sdm1  
raw /home/oracle/dev/raw/stock-10 /dev/sdm2  
raw /home/oracle/dev/raw/stock-11 /dev/sdo1  
raw /home/oracle/dev/raw/stock-12 /dev/sdo2  
raw /home/oracle/dev/raw/ordl-2-1 /dev/sdk3  
raw /home/oracle/dev/raw/ordl-2-2 /dev/sdm3  
raw /home/oracle/dev/raw/ordl-2-3 /dev/sdo3  
raw /home/oracle/dev/raw/cust-4 /dev/sdk10  
raw /home/oracle/dev/raw/cust-5 /dev/sdm10  
raw /home/oracle/dev/raw/cust-6 /dev/sdo10  
raw /home/oracle/dev/raw/nord-2 /dev/sdk5  
raw /home/oracle/dev/raw/hist-2 /dev/sdm5  
raw /home/oracle/dev/raw/ordr-2 /dev/sdo5  
raw /home/oracle/dev/raw/icust2-4 /dev/sdk6  
raw /home/oracle/dev/raw/icust2-5 /dev/sdm6  
raw /home/oracle/dev/raw/icust2-6 /dev/sdo6  
raw /home/oracle/dev/raw/ware-2 /dev/sdk7  
raw /home/oracle/dev/raw/dist-2 /dev/sdm7  
raw /home/oracle/dev/raw/system-2 /dev/sdo7  
raw /home/oracle/dev/raw/roll-2 /dev/sdk8  
raw /home/oracle/dev/raw/iordr1-2 /dev/sdm8  
raw /home/oracle/dev/raw/iordr2-2 /dev/sdo8  
raw /home/oracle/dev/raw/oracmd /dev/sdk9  
raw /home/oracle/dev/raw/temp-3 /dev/sdm9  
raw /home/oracle/dev/raw/temp-4 /dev/sdo9  
  
raw /home/oracle/dev/raw/log-3-1 /dev/sdq1  
raw /home/oracle/dev/raw/log-3-2 /dev/sdq2  
raw /home/oracle/dev/raw/stock-13 /dev/sds1  
raw /home/oracle/dev/raw/stock-14 /dev/sds2  
raw /home/oracle/dev/raw/stock-15 /dev/sdu1  
raw /home/oracle/dev/raw/stock-16 /dev/sdu2  
raw /home/oracle/dev/raw/stock-17 /dev/sdw1
```

```
raw /home/oracle/dev/raw/stock-18 /dev/sdw2  
raw /home/oracle/dev/raw/ordl-3-1 /dev/sds3  
raw /home/oracle/dev/raw/ordl-3-2 /dev/sdu3  
raw /home/oracle/dev/raw/ordl-3-3 /dev/sdw3  
raw /home/oracle/dev/raw/cust-7 /dev/sds10  
raw /home/oracle/dev/raw/cust-8 /dev/sdu10  
raw /home/oracle/dev/raw/cust-9 /dev/sdw10  
raw /home/oracle/dev/raw/nord-3 /dev/sds5  
raw /home/oracle/dev/raw/hist-3 /dev/sdu5  
raw /home/oracle/dev/raw/ordr-3 /dev/sdw5  
raw /home/oracle/dev/raw/icust2-7 /dev/sds6  
raw /home/oracle/dev/raw/icust2-8 /dev/sdu6  
raw /home/oracle/dev/raw/icust2-9 /dev/sdw6  
raw /home/oracle/dev/raw/ware-3 /dev/sds7  
raw /home/oracle/dev/raw/dist-3 /dev/sdu7  
raw /home/oracle/dev/raw/system-3 /dev/sdw7  
raw /home/oracle/dev/raw/roll-3 /dev/sds8  
raw /home/oracle/dev/raw/iordr1-3 /dev/sdu8  
raw /home/oracle/dev/raw/iordr2-3 /dev/sdw8  
raw /home/oracle/dev/raw/indexes-1 /dev/sds9  
raw /home/oracle/dev/raw/temp-5 /dev/sdu9  
raw /home/oracle/dev/raw/temp-6 /dev/sdw9  
  
raw /home/oracle/dev/raw/log-4-1 /dev/sdy1  
raw /home/oracle/dev/raw/log-4-2 /dev/sdy2  
raw /home/oracle/dev/raw/stock-19 /dev/sdaa1  
raw /home/oracle/dev/raw/stock-20 /dev/sdaa2  
raw /home/oracle/dev/raw/stock-21 /dev/sdac1  
raw /home/oracle/dev/raw/stock-22 /dev/sdac2  
raw /home/oracle/dev/raw/stock-23 /dev/sdael  
raw /home/oracle/dev/raw/stock-24 /dev/sdae2  
raw /home/oracle/dev/raw/ordl-4-1 /dev/sdaa3  
raw /home/oracle/dev/raw/ordl-4-2 /dev/sdac3  
raw /home/oracle/dev/raw/ordl-4-3 /dev/sdae3  
raw /home/oracle/dev/raw/cust-10 /dev/sdaa10  
raw /home/oracle/dev/raw/cust-11 /dev/sdac10  
raw /home/oracle/dev/raw/cust-12 /dev/sdae10  
raw /home/oracle/dev/raw/nord-4 /dev/sdaa5  
raw /home/oracle/dev/raw/hist-4 /dev/sdac5  
raw /home/oracle/dev/raw/ordr-4 /dev/sdae5  
raw /home/oracle/dev/raw/icust2-10 /dev/sdaa6  
raw /home/oracle/dev/raw/icust2-11 /dev/sdac6  
raw /home/oracle/dev/raw/icust2-12 /dev/sdae6  
raw /home/oracle/dev/raw/ware-4 /dev/sdaa7  
raw /home/oracle/dev/raw/dist-4 /dev/sdac7  
raw /home/oracle/dev/raw/system-4 /dev/sdae7  
raw /home/oracle/dev/raw/roll-4 /dev/sdaa8  
raw /home/oracle/dev/raw/iordr1-4 /dev/sdac8  
raw /home/oracle/dev/raw/iordr2-4 /dev/sdae8  
raw /home/oracle/dev/raw/indexes-2 /dev/sdaa9  
raw /home/oracle/dev/raw/temp-7 /dev/sdac9  
raw /home/oracle/dev/raw/temp-8 /dev/sdae9  
  
raw /home/oracle/dev/raw/log-5-1 /dev/sdag1  
raw /home/oracle/dev/raw/log-5-2 /dev/sdag2  
raw /home/oracle/dev/raw/stock-25 /dev/sdai1  
raw /home/oracle/dev/raw/stock-26 /dev/sdai2  
raw /home/oracle/dev/raw/stock-27 /dev/sdak1  
raw /home/oracle/dev/raw/stock-28 /dev/sdak2  
raw /home/oracle/dev/raw/stock-29 /dev/sdam1  
raw /home/oracle/dev/raw/stock-30 /dev/sdam2  
raw /home/oracle/dev/raw/ordl-5-1 /dev/sdai3  
raw /home/oracle/dev/raw/ordl-5-2 /dev/sdak3  
raw /home/oracle/dev/raw/ordl-5-3 /dev/sdam3  
raw /home/oracle/dev/raw/cust-13 /dev/sdai10  
raw /home/oracle/dev/raw/cust-14 /dev/sdak10  
raw /home/oracle/dev/raw/cust-15 /dev/sdam10  
raw /home/oracle/dev/raw/nord-5 /dev/sdai5  
raw /home/oracle/dev/raw/hist-5 /dev/sdak5  
raw /home/oracle/dev/raw/ordr-5 /dev/sdam5  
raw /home/oracle/dev/raw/icust2-13 /dev/sdai6  
raw /home/oracle/dev/raw/icust2-14 /dev/sdak6  
raw /home/oracle/dev/raw/icust2-15 /dev/sdam6  
raw /home/oracle/dev/raw/ware-5 /dev/sdai7  
raw /home/oracle/dev/raw/dist-5 /dev/sdak7  
raw /home/oracle/dev/raw/system-5 /dev/sdam7  
raw /home/oracle/dev/raw/roll-5 /dev/sdai8  
raw /home/oracle/dev/raw/iordr1-5 /dev/sdak8  
raw /home/oracle/dev/raw/iordr2-5 /dev/sdam8  
raw /home/oracle/dev/raw/statpack-1 /dev/sdai9  
raw /home/oracle/dev/raw/temp-9 /dev/sdak9  
raw /home/oracle/dev/raw/temp-10 /dev/sdam9  
  
raw /home/oracle/dev/raw/log-6-1 /dev/sdao1  
raw /home/oracle/dev/raw/log-6-2 /dev/sdao2  
raw /home/oracle/dev/raw/stock-31 /dev/sdaq1  
raw /home/oracle/dev/raw/stock-32 /dev/sdaq2  
raw /home/oracle/dev/raw/stock-33 /dev/sdas1  
raw /home/oracle/dev/raw/stock-34 /dev/sdas2  
raw /home/oracle/dev/raw/stock-35 /dev/sdau1  
raw /home/oracle/dev/raw/stock-36 /dev/sdau2  
raw /home/oracle/dev/raw/ordl-6-1 /dev/sdaq3  
raw /home/oracle/dev/raw/ordl-6-2 /dev/sdas3  
raw /home/oracle/dev/raw/ordl-6-3 /dev/sdau3  
raw /home/oracle/dev/raw/cust-16 /dev/sdaq10  
raw /home/oracle/dev/raw/cust-17 /dev/sdas10  
raw /home/oracle/dev/raw/cust-18 /dev/sdau10  
raw /home/oracle/dev/raw/nord-6 /dev/sdaq5  
raw /home/oracle/dev/raw/hist-6 /dev/sdas5  
raw /home/oracle/dev/raw/ordr-6 /dev/sdau5  
raw /home/oracle/dev/raw/icust2-16 /dev/sdaq6
```

```

raw /home/oracle/dev/raw/icust2-17 /dev/sdas6
raw /home/oracle/dev/raw/icust2-18 /dev/sdau6
raw /home/oracle/dev/raw/ware-6 /dev/sdaq7
raw /home/oracle/dev/raw/dist-6 /dev/sdas7
raw /home/oracle/dev/raw/system-6 /dev/sdau7
raw /home/oracle/dev/raw/roll-6 /dev/sdaq8
raw /home/oracle/dev/raw/iordr1-6 /dev/sdas8
raw /home/oracle/dev/raw/iordr2-6 /dev/sdau8
raw /home/oracle/dev/raw/item-1 /dev/sdaq9
raw /home/oracle/dev/raw/temp-11 /dev/sdas9
raw /home/oracle/dev/raw/temp-12 /dev/sdau9

raw /home/oracle/dev/raw/log-7-1 /dev/sdaw1
raw /home/oracle/dev/raw/log-7-2 /dev/sdaw2
raw /home/oracle/dev/raw/stock-37 /dev/sday1
raw /home/oracle/dev/raw/stock-38 /dev/sday2
raw /home/oracle/dev/raw/stock-39 /dev/sdba1
raw /home/oracle/dev/raw/stock-40 /dev/sdba2
raw /home/oracle/dev/raw/stock-41 /dev/sdbc1
raw /home/oracle/dev/raw/stock-42 /dev/sdbc2
raw /home/oracle/dev/raw/ordl-7-1 /dev/sday3
raw /home/oracle/dev/raw/ordl-7-2 /dev/sdba3
raw /home/oracle/dev/raw/ordl-7-3 /dev/sdbc3
raw /home/oracle/dev/raw/cust-19 /dev/sday10
raw /home/oracle/dev/raw/cust-20 /dev/sdba10
raw /home/oracle/dev/raw/cust-21 /dev/sdbc10
raw /home/oracle/dev/raw/nord-7 /dev/sday5
raw /home/oracle/dev/raw/hist-7 /dev/sdba5
raw /home/oracle/dev/raw/ordr-7 /dev/sdbc5
raw /home/oracle/dev/raw/icust2-19 /dev/sday6
raw /home/oracle/dev/raw/icust2-20 /dev/sdba6
raw /home/oracle/dev/raw/icust2-21 /dev/sdbc6
raw /home/oracle/dev/raw/ware-7 /dev/sday7
raw /home/oracle/dev/raw/dist-7 /dev/sdba7
raw /home/oracle/dev/raw/system-7 /dev/sdbc7
raw /home/oracle/dev/raw/roll-7 /dev/sday8
raw /home/oracle/dev/raw/iordr1-7 /dev/sdba8
raw /home/oracle/dev/raw/iordr2-7 /dev/sdbc8
raw /home/oracle/dev/raw/cust-25 /dev/sday9
raw /home/oracle/dev/raw/temp-13 /dev/sdba9
raw /home/oracle/dev/raw/temp-14 /dev/sdbc9

raw /home/oracle/dev/raw/log-8-1 /dev/sdbe1
raw /home/oracle/dev/raw/log-8-2 /dev/sdbe2
raw /home/oracle/dev/raw/stock-43 /dev/sdbg1
raw /home/oracle/dev/raw/stock-44 /dev/sdbg2
raw /home/oracle/dev/raw/stock-45 /dev/sdbi1
raw /home/oracle/dev/raw/stock-46 /dev/sdbi2
raw /home/oracle/dev/raw/stock-47 /dev/sdbk1
raw /home/oracle/dev/raw/stock-48 /dev/sdbk2
raw /home/oracle/dev/raw/ordl-8-1 /dev/sdbg3
raw /home/oracle/dev/raw/ordl-8-2 /dev/sdbi3
raw /home/oracle/dev/raw/ordl-8-3 /dev/sdbk3
raw /home/oracle/dev/raw/cust-22 /dev/sdbg10
raw /home/oracle/dev/raw/cust-23 /dev/sdbi10
raw /home/oracle/dev/raw/cust-24 /dev/sdbk10
raw /home/oracle/dev/raw/nord-8 /dev/sdbg5
raw /home/oracle/dev/raw/hist-8 /dev/sdbi5
raw /home/oracle/dev/raw/ordr-8 /dev/sdbk5
raw /home/oracle/dev/raw/icust2-22 /dev/sdbg6
raw /home/oracle/dev/raw/icust2-23 /dev/sdbi6
raw /home/oracle/dev/raw/icust2-24 /dev/sdbk6
raw /home/oracle/dev/raw/ware-8 /dev/sdbg7
raw /home/oracle/dev/raw/dist-8 /dev/sdbi7
raw /home/oracle/dev/raw/system-8 /dev/sdbk7
raw /home/oracle/dev/raw/roll-8 /dev/sdbg8
raw /home/oracle/dev/raw/iordr1-8 /dev/sdbi8
raw /home/oracle/dev/raw/iordr2-8 /dev/sdbk8
raw /home/oracle/dev/raw/stok-49 /dev/sdbg9
raw /home/oracle/dev/raw/temp-15 /dev/sdbi9
raw /home/oracle/dev/raw/stok-50 /dev/sdbk9

*****

bash_profile
*****
# This file sets options when remaking Apache
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
. ~/.bashrc
fi

export inst=`echo $HOSTNAME| awk '{print substr($0,length($0))}'`

export DB_NAME=tpcc
export ORACLE_SERVICE=tpcc
export ORACLE_SID=tpcc${inst}
export ORACLE_HOME=/home/oracle/OraHome1
export VLM_WINDOW_SIZE=973078528
PATH=$PATH:$HOME/bin:$ORACLE_HOME/bin:./
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib

export PATH
/usr/local/bin/lowermap
unset USERNAME

*****

```

```

rc.local
*****

#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local

/sbin/insmod qla2300

/sbin/insmod softdog nowayout=1 soft_noboot=1

/root/mkraw.sh

echo 2 > /proc/sys/kernel/shm-use-bigpages
echo 98304 > /proc/sys/fs/aio-max-nr

*****
realtime.sh
*****

renice -20 -p `ps -ef | grep oracm | grep -v grep | cut -b 10-15`
renice -20 -p `ps -ef | grep tpcc | grep -v grep | cut -b 10-15`

*****
sysctl.conf
*****
# Disables packet forwarding
net.ipv4.ip_forward = 0
# Enables source route verification
net.ipv4.conf.default.rp_filter = 1
# Disables the magic-sysrq key
kernel.sysrq = 1
kernel.shmmax=3000000000
kernel.sem = 250 32000 100 128
fs.file-max=204800
#kernel.shm-use-bigpages=2

*****
grub.conf
*****
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz

title Red Hat Linux (2.4.9-e.8enterprise)
root (hd0,0)
kernel /vmlinuz-2.4.9-e.8enterprise ro root=/dev/ida/c0d0p2
bigpages=14200MB
initrd /initrd-2.4.9-e.8enterprise.img
title Red Hat Linux Advanced Server-smp (2.4.9-e.3smp)
root (hd0,0)
kernel /vmlinuz-2.4.9-e.3smp ro root=/dev/ida/c0d0p2
initrd /initrd-2.4.9-e.3smp.img
title Red Hat Linux Advanced Server-up (2.4.9-e.3)
root (hd0,0)
kernel /vmlinuz-2.4.9-e.3 ro root=/dev/ida/c0d0p2
initrd /initrd-2.4.9-e.3.img

*****
fstab
*****
LABEL=/ / ext3 defaults
1 1
LABEL=/boot /boot ext3 defaults
1 2
none /dev/pts devpts
gid=5,mode=620 0 0
none /proc proc defaults
0 0
none /dev/shm tmpfs
nr_inodes=32000,nr_block
s=4194304 0 0
/dev/ida/c0d0p4 /home2 ext2 defaults
1 2
/dev/ida/c0d0p3 swap swap defaults
0 0
/dev/cdrom /mnt/cdrom iso9660
noauto,owner,kudzu,ro 0
0
/dev/fd0 /mnt/floppy auto
noauto,owner,kudzu 0 0

*****
tcp_tune.sh
*****

echo 0 > /proc/sys/net/ipv4/tcp_sack
echo 0 > /proc/sys/net/ipv4/tcp_timestamps
echo 0 > /proc/sys/net/ipv4/tcp_window_scaling
/etc/rc.d/init.d/network restart

```



```

*****
tune_elv.sh
*****

/sbin/elvtune -r 512 -w 8192 /dev/sda
/sbin/elvtune -r 512 -w 8192 /dev/sdb
/sbin/elvtune -r 512 -w 8192 /dev/sdc
/sbin/elvtune -r 512 -w 8192 /dev/sdd
/sbin/elvtune -r 512 -w 8192 /dev/sde
/sbin/elvtune -r 512 -w 8192 /dev/sdf
/sbin/elvtune -r 512 -w 8192 /dev/sdg
/sbin/elvtune -r 512 -w 8192 /dev/sdh
/sbin/elvtune -r 512 -w 8192 /dev/sdi
/sbin/elvtune -r 512 -w 8192 /dev/sdj
/sbin/elvtune -r 512 -w 8192 /dev/sdk
/sbin/elvtune -r 512 -w 8192 /dev/sdl
/sbin/elvtune -r 512 -w 8192 /dev/sdm
/sbin/elvtune -r 512 -w 8192 /dev/sdn
/sbin/elvtune -r 512 -w 8192 /dev/sdo
/sbin/elvtune -r 512 -w 8192 /dev/sdp
/sbin/elvtune -r 512 -w 8192 /dev/sdq
/sbin/elvtune -r 512 -w 8192 /dev/sdr
/sbin/elvtune -r 512 -w 8192 /dev/sds
/sbin/elvtune -r 512 -w 8192 /dev/sdt
/sbin/elvtune -r 512 -w 8192 /dev/sdu
/sbin/elvtune -r 512 -w 8192 /dev/sdv
/sbin/elvtune -r 512 -w 8192 /dev/sdw
/sbin/elvtune -r 512 -w 8192 /dev/sdx
/sbin/elvtune -r 512 -w 8192 /dev/sdy
/sbin/elvtune -r 512 -w 8192 /dev/sdz
/sbin/elvtune -r 512 -w 8192 /dev/sdaa
/sbin/elvtune -r 512 -w 8192 /dev/sdab
/sbin/elvtune -r 512 -w 8192 /dev/sdac
/sbin/elvtune -r 512 -w 8192 /dev/sdad
/sbin/elvtune -r 512 -w 8192 /dev/sdae
/sbin/elvtune -r 512 -w 8192 /dev/sdaf
/sbin/elvtune -r 512 -w 8192 /dev/sdag
/sbin/elvtune -r 512 -w 8192 /dev/sdah
/sbin/elvtune -r 512 -w 8192 /dev/sdai
/sbin/elvtune -r 512 -w 8192 /dev/sdaj
/sbin/elvtune -r 512 -w 8192 /dev/sdak
/sbin/elvtune -r 512 -w 8192 /dev/sdal
/sbin/elvtune -r 512 -w 8192 /dev/sdam
/sbin/elvtune -r 512 -w 8192 /dev/sdan
/sbin/elvtune -r 512 -w 8192 /dev/sdao
/sbin/elvtune -r 512 -w 8192 /dev/sdap
/sbin/elvtune -r 512 -w 8192 /dev/sdaq
/sbin/elvtune -r 512 -w 8192 /dev/sdar
/sbin/elvtune -r 512 -w 8192 /dev/sdas
/sbin/elvtune -r 512 -w 8192 /dev/sdat
/sbin/elvtune -r 512 -w 8192 /dev/sdav
/sbin/elvtune -r 512 -w 8192 /dev/sdaw
/sbin/elvtune -r 512 -w 8192 /dev/sdax
/sbin/elvtune -r 512 -w 8192 /dev/sday
/sbin/elvtune -r 512 -w 8192 /dev/sdaz
/sbin/elvtune -r 512 -w 8192 /dev/sdba
/sbin/elvtune -r 512 -w 8192 /dev/sdbb
/sbin/elvtune -r 512 -w 8192 /dev/sdbc
/sbin/elvtune -r 512 -w 8192 /dev/sdbd
/sbin/elvtune -r 512 -w 8192 /dev/sdbe
/sbin/elvtune -r 512 -w 8192 /dev/sdbf
/sbin/elvtune -r 512 -w 8192 /dev/sdbg
/sbin/elvtune -r 512 -w 8192 /dev/sdbh
/sbin/elvtune -r 512 -w 8192 /dev/sdbi
/sbin/elvtune -r 512 -w 8192 /dev/sdbj
/sbin/elvtune -r 512 -w 8192 /dev/sdbk
/sbin/elvtune -r 512 -w 8192 /dev/sdbl

*****
lowermap.c
*****

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <string.h>

#define DEFAULT_MAPPED_BASE 0x10000000

int main(int argc, char *argv[])
{
    pid_t ppid;
    char buf[256];
    unsigned long mapped_base;
    int ret;

    ppid = getppid();
    mapped_base = DEFAULT_MAPPED_BASE;

    if (argc == 2)
    {
        mapped_base = strtoul(argv[1], 0, 0);
    }

    sprintf(buf, "echo %lu >/proc/%u/mapped_base", mapped_base,
    ppid);

```

```

setuid(0);
ret = system(buf);
if (ret == 0)
    printf("Lowering mapped base of pid=%u to 0x%X\n", ppid,
    mapped_base);
else
    printf("unable to lower mapped base. You might need to:\n"
    " chmod 4711 lowermap\n chown root.root lowermap\n");
exit(-ret);
}

*****
/sbin/chkconfig output
*****
keytable 0:off 1:on 2:on 3:off 4:off 5:off 6:off
atd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
kderotate 0:off 1:off 2:off 3:off 4:off 5:off 6:off
syslog 0:off 1:off 2:on 3:on 4:on 5:on 6:off
gpm 0:off 1:off 2:on 3:off 4:off 5:off 6:off
sendmail 0:off 1:off 2:on 3:off 4:off 5:off 6:off
kudzu 0:off 1:off 2:off 3:on 4:on 5:on 6:off
microcode_ctl 0:off 1:off 2:off 3:off 4:off 5:off 6:off
netfs 0:off 1:off 2:off 3:off 4:off 5:off 6:off
network 0:off 1:off 2:on 3:on 4:on 5:on 6:off
random 0:off 1:off 2:on 3:on 4:on 5:on 6:off
rawdevices 0:off 1:off 2:off 3:on 4:on 5:on 6:off
apmd 0:off 1:off 2:on 3:off 4:off 5:off 6:off
ipchains 0:off 1:off 2:off 3:off 4:off 5:off 6:off
iptables 0:off 1:off 2:on 3:off 4:off 5:off 6:off
crond 0:off 1:off 2:on 3:on 4:on 5:on 6:off
anacron 0:off 1:off 2:on 3:off 4:off 5:off 6:off
lpd 0:off 1:off 2:on 3:off 4:off 5:off 6:off
xfs 0:off 1:off 2:on 3:on 4:on 5:on 6:off
ntpd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
portmap 0:off 1:off 2:off 3:on 4:on 5:on 6:off
xinetd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
autofs 0:off 1:off 2:off 3:on 4:on 5:on 6:off
nfs 0:off 1:off 2:off 3:off 4:off 5:off 6:off
nfslock 0:off 1:off 2:off 3:off 4:off 5:off 6:off
nsd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
identd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
radvd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
rwhod 0:off 1:off 2:off 3:off 4:off 5:off 6:off
snmpd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
snmptrapd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
cpqarrayd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
vncserver 0:off 1:off 2:off 3:off 4:off 5:off 6:off
smartd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
rhnmd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
ypbind 0:off 1:off 2:off 3:off 4:off 5:off 6:off
sshd 0:off 1:off 2:on 3:off 4:off 5:off 6:off
rstatd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
rusersd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
rwalld 0:off 1:off 2:off 3:off 4:off 5:off 6:off
netdump-server 0:off 1:off 2:off 3:off 4:off 5:off 6:off
yppasswdd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
ypserv 0:off 1:off 2:off 3:off 4:off 5:off 6:off
ypxfrd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
smb 0:off 1:off 2:off 3:off 4:off 5:off 6:off
bcm5820 0:off 1:off 2:off 3:off 4:off 5:off 6:off
httpd 0:off 1:off 2:off 3:off 4:off 5:off 6:off
squid 0:off 1:off 2:off 3:off 4:off 5:off 6:off
tux 0:off 1:off 2:off 3:off 4:off 5:off 6:off
arpwatch 0:off 1:off 2:off 3:off 4:off 5:off 6:off
cluster 0:off 1:off 2:on 3:off 4:off 5:off 6:off
ipvsadm 0:off 1:off 2:off 3:off 4:off 5:off 6:off
piranha-gui 0:off 1:off 2:off 3:off 4:off 5:off 6:off
pulse 0:off 1:off 2:off 3:off 4:off 5:off 6:off
netdump 0:off 1:off 2:off 3:off 4:off 5:off 6:off
xinetd based services:
  chargen-udp: off
  chargen: off
  daytime-udp: off
  daytime: off
  echo-udp: off
  echo: off
  time-udp: off
  time: on
  sgi_fam: off
  finger: off
  rexec: on
  rlogin: on
  rsh: on
  ntalk: off
  talk: off
  telnet: on
  rsync: off

```

DATABASE TUNABLES

```
*****  
lowering base memory for Oracle  
*****  
cd $ORACLE_HOME/rdbms/lib  
genksms -s 0x1500000 > ksms.s  
make -f ins_rdbms.mk ksms.o  
make -f ins_rdbms.mk ioracle
```

```
*****
```

```
init_1.ora  
*****  
  
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_69,t_1_70,  
t_1_71,t_1_72,t_1_73,t_1_74,t_1_75)  
ifile=$HOME/p_run.ora
```

```
*****
```

```
init_2.ora  
*****  
  
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_69,t_2_70,  
t_2_71,t_2_72,t_2_73,t_2_74,t_2_75)  
ifile=$HOME/p_run.ora
```

```
*****
```

```
init_3.ora  
*****  
  
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_69,t_3_70,  
t_3_71,t_3_72,t_3_73,t_3_74,t_3_75)  
ifile=$HOME/p_run.ora
```

```
*****
```

```
init_4.ora  
*****  
  
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_69,t_4_70,  
t_4_71,t_4_72,t_4_73,t_4_74,t_4_75)  
ifile=$HOME/p_run.ora
```

```
*****
```

```
init_5.ora  
*****  
  
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_69,t_5_70,  
t_5_71,t_5_72,t_5_73,t_5_74,t_5_75)  
ifile=$HOME/p_run.ora
```

```
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_69,t_5_70,  
t_5_71,t_5_72,t_5_73,t_5_74,t_5_75)  
ifile=$HOME/p_run.ora
```

```
*****
```

```
init_6.ora  
*****  
  
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_69,t_6_70,  
t_6_71,t_6_72,t_6_73,t_6_74,t_6_75)  
ifile=$HOME/p_run.ora
```

```
*****
```

```
init_7.ora  
*****  
  
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_69,t_7_70,  
t_7_71,t_7_72,t_7_73,t_7_74,t_7_75)  
ifile=$HOME/p_run.ora
```

```
*****
```

```
init_8.ora  
*****  
  
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_69,t_8_70,  
t_8_71,t_8_72,t_8_73,t_8_74,t_8_75)  
ifile=$HOME/p_run.ora
```

```
*****
```

```
p_run.ora  
*****  
  
#  
# 9i RAC 4K  
#  
#  
  
_db_block_lru_latches = 64  
_db_writer_max_writes = 1024  
_db_writer_chunk_writes = 200  
_db_writer_processes = 4  
db_cache_advice = off  
compatible = 9.2.0.0.0  
db_name = tpcc  
control_files = /home/oracle/dev/raw/control-001  
sort_area_size = 10485760  
parallel_max_servers = 0  
parallel_min_servers = 0  
recovery_parallelism = 0  
log_buffer = 10485760  
db_block_size = 4096  
db_files=250  
cursor_space_for_time = TRUE  
shared_pool_size = 160M  
java_pool_size = 15728640  
use_indirect_data_buffers = TRUE #pse36  
db_block_buffers = 3235840  
_log_simultaneous_copies = 16  
dml_locks = 60  
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      = 80
pre_page_sga                = TRUE
processes                   = 100
sessions                    = 150
open_cursors               = 180
replication_dependency_tracking = FALSE
transactions_per_rollback_segment = 1
timed_statistics           = FALSE
transaction_auditing        = FALSE
db_block_checking          = FALSE
cluster_database           = TRUE
cluster_database_instances = 8
_kc1_name_table_latches    = 64
_lm_lms=1
statistics_level           = basic
log_parallelism            = 2
_lm_dd_interval            = 30
gc_files_to_locks          = "1-8=1000EACH:\
9=1:\
10-12=100000:\
13-15=100000:\
16-18=100000:\
19-21=100000:\
22-24=100000:\
25-27=100000:\
28-30=100000:\
31-33=100000:\
34-39=350000:\
40-45=350000:\
46-51=350000:\
52-57=350000:\
58-63=350000:\
64-69=350000:\
70-75=350000:\
76-81=350000:\
82-195=1EACH:\
197-199=1EACH"
_lm_file_affinity="9,10-12,34-39,82,90,99-
101,123,131,139,172,180=1:\
13-15,40-45,83,91,102-104,124,132,140,173,181,188=2:\
16-18,46-51,84,92,105-107,125,133,141,174,182,189=3:\
19-21,52-57,85,93,108-110,126,134,142,175,183,190=4:\
22-24,58-63,86,94,111-113,127,135,143,176,184,191=5:\
25-27,64-69,87,95,114-116,128,136,144,177,185,192=6:\
28-30,70-75,88,96,117-119,129,137,145,178,186,193=7:\
31-33,76-81,89,97,120-122,130,138,146,179,187,184=8"
_gcs_resources=400000
_pcm_shadow_locks=400000
_interconnect_checksum = FALSE
db_block_checksum        = FALSE

```

CLIENT O/S AND TUXEDO CONFIGURATION

```

*****
config.nice
*****

```

```

#!/bin/sh
#
# Created by configure

"./configure" \
"--with-mpm=worker" \
"--enable-auth=shared" \
"--enable-env=shared" \
"--enable-negotiation=shared" \
"--enable-userdir=shared" \
"--enable-include=shared" \
"--enable-status=shared" \
"--disable-cgid" \
"--enable-setenvif=shared" \
"--enable-dir=shared" \
"--enable-imap=shared" \
"--enable-actions=shared" \
"--enable-alias=shared" \
"--enable-autoindex=shared" \
"--enable-asis=shared" \
"--enable-cgi=shared" \
"--prefix=/usr/local/ap2w" \
"$@"

```

```

*****
grub.conf
*****

```

```

# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to
this file
# NOTICE: You have a /boot partition. This means that
#           all kernel and initrd paths are relative to /boot/, eg.
#           root (hd0,0)
#           kernel /vmlinuz-version ro root=/dev/cciss/c0d0p3
#           initrd /initrd-version.img
#boot=/dev/cciss/c0d0
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz

```

```

title Red Hat Linux (2.4.18-7.80custom)
    root (hd0,0)
    kernel /vmlinuz-2.4.18-7.80custom ro root=LABEL=/ root=LABEL=/
ide=nodma
    initrd /initrd-2.4.18-7.80custom.img
title Red Hat Linux (2.4.18-7.80smp)
    root (hd0,0)
    kernel /vmlinuz-2.4.18-7.80smp ro root=LABEL=/ root=LABEL=/
ide=nodma
    initrd /initrd-2.4.18-7.80smp.img
title Red Hat Linux-up (2.4.18-7.80)
    root (hd0,0)
    kernel /vmlinuz-2.4.18-7.80 ro root=LABEL=/ root=LABEL=/
    initrd /initrd-2.4.18-7.80.img

```

```

*****
httpd.conf
*****

```

```

ServerTokens OS

ServerRoot "/usr/local/ap2"

PidFile run/httpd.pid

Timeout 300

KeepAlive On

MaxKeepAliveRequests 0

KeepAliveTimeout 999

CoreDumpDirectory /usr/local/ap2

##
## Server-Pool Size Regulation (MPM specific)
##

# prefork MPM
# StartServers: number of server processes to start
# MinSpareServers: minimum number of server processes which are
# kept spare
# MaxSpareServers: maximum number of server processes which are
# kept spare
# MaxClients: maximum number of server processes allowed to start
# MaxRequestsPerChild: maximum number of requests a server process
# serves
<IfModule prefork.c>
StartServers      1000
MinSpareServers   5
MaxSpareServers   1000
MaxClients        7300
MaxRequestsPerChild 0
</IfModule>

# worker MPM
# StartServers: initial number of server processes to start
# MaxClients: maximum number of simultaneous client connections
# MinSpareThreads: minimum number of worker threads which are kept
# spare
# MaxSpareThreads: maximum number of worker threads which are kept
# spare
# ThreadsPerChild: constant number of worker threads in each server
# process
# MaxRequestsPerChild: maximum number of requests a server process
# serves
<IfModule worker.c>
ServerLimit      16
ThreadLimit      500
### max processes
StartServers     16
MaxClients       7216
MinSpareThreads  10
MaxSpareThreads  7300
ThreadsPerChild  451
MaxRequestsPerChild 0
</IfModule>

Listen 80

LoadModule tpcc_module /usr/local/ap2/lib/apache/mod_tpcc.so

User apache
Group apache

#
# ServerAdmin: Your address, where problems with the server should
# be
# e-mailed. This address appears on some server-generated pages,
# such
# as error documents. e.g. admin@your-domain.com
#
ServerAdmin you@your.address

ServerName lclil

UseCanonicalName Off

```

```

DocumentRoot "/var/www/html"

<Directory />
    Options FollowSymLinks
        AllowOverride None
</Directory>

TypesConfig /etc/mime.types

#
# DefaultType is the default MIME type the server will use for a
document
# if it cannot otherwise determine one, such as from filename
extensions.
# If your server contains mostly text or HTML documents,
"text/plain" is
# a good value.  If most of your content is binary, such as
applications
# or images, you may want to use "application/octet-stream" instead
to
# keep browsers from trying to display binary files as though they
are
# text.
DefaultType text/plain

#
# The mod_mime_magic module allows the server to use various hints
from the
# contents of the file itself to determine its type.  The
MIMEMagicFile
# directive tells the module where the hint definitions are
located.
#
<IfModule mod_mime_magic.c>
#   MIMEMagicFile /usr/share/magic.mime
#   MIMEMagicFile conf/magic
</IfModule>

#
# HostnameLookups: Log the names of clients or just their IP
addresses
# e.g., www.apache.org (on) or 204.62.129.132 (off).
# The default is off because it'd be overall better for the net if
people
# had to knowingly turn this feature on, since enabling it means
that
# each client request will result in AT LEAST one lookup request to
the
# nameserver.
HostnameLookups Off

#
# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a
<VirtualHost>
# container, error messages relating to that virtual host will be
# logged here.  If you *do* define an error logfile for a
<VirtualHost>
# container, that host's errors will be logged there and not here.
#
ErrorLog logs/error_log

#
# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
LogLevel warn

#
# The following directives define some format nicknames for use
with
# a CustomLog directive (see below).
#
LogFormat "%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-
Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

#CustomLog logs/access_log combined

<Location /tpcc>
    SetHandler tpcc
</Location>

```

```

*****
httpd_modules
*****

[root@cl93 tmp]# /usr/local/ap2/sbin/httpd -l
Compiled in modules:
  core.c
  mod_access.c
  mod_log_config.c
  worker.c
  http_core.c
  mod_mime.c
  mod_so.c

*****
ubb
*****

#
# 9i RAC UBBconfig file for 24 clients configuration
#
# Clients systems have identical 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 cl93
MAXACCESSERS 9800 # 1024 or more
MAXGTT 8000
MAXSERVERS 80
MAXSERVICES 410 #MAXSERVERS * #-of-services-each-server + 10 (for
BBL)
MODEL SHM
LDBAL Y
*MACHINES
DEFAULT:
    TUXDIR="/home/bea/tuxedo8.0"
    APPDIR="/home/bea/tuxedo8.0"
    TUXCONFIG="/home/bea/tuxedo8.0/tuxconfig"
    UID=0
    GID=0
    TYPE="LINUX"
cl93 LMID=cl93

*GROUPS
TPCC
    LMID=cl93 GRPNO=1 OPENINFO=NONE
DELI
    LMID=cl93 GRPNO=2 OPENINFO=NONE

*SERVERS
DEFAULT: CLOPT="-A"
tpccora SRVGRP=TPCC SRVID=1 RQADDR=txnque1 REPLYQ=Y MIN=30 MAX=40
deliora SRVGRP=DELI SRVID=2 RQADDR=txnque2 REPLYQ=N MIN=2 MAX=20

*SERVICES
DEFAULT:
    LOAD=1
    PRIO=1
    BUFTYPE="CARRAY"
    TRANTIME=900
    AUTOTRAN=N
no_transaction
os_transaction
pt_transaction
sl_transaction
dy_transaction

```

RUNTIME CLAST = 87

Appendix D:
Third Party Letters

RESOURCES

- [Order Status](#)
- [My Company](#)
- [My Account](#)
- [Account Team](#)
- [New Accounts](#)
- [Rebates](#)
- [Special Events](#)
- [CDW Outlet](#)
- [CDW Leasing](#)
- [Technical Support](#)
- [E-Newsletters](#)



Manufacturer
NETGEAR
Product Links
<ul style="list-style-type: none"> > Similar Products > Send to Associate

Netgear GS516T 16 port Rack Mountable Switch

Product Information

▶ 16-port 10/100/1000Mbps Gigabit Ethernet unmanaged stackable rackmountable switch

Usually Ships:	Same Day
CDW Part No.:	392502
Mfg. Part No.:	GS516TNA

Price: **\$1,399.58**

▶ **OVERVIEW**

Your office network gets gigabit speed to burn with NETGEAR's GS516T 10/100/1000Mbps Gigabit Switch

Your office network gets gigabit speed to burn with NETGEAR's GS516T 10/100/1000Mbps Gigabit Switch! Its 16 ports send data at scorching speeds – up to 2000Mbps per port in full-duplex mode, and every port also features 10/100/1000 automatic speed and full/half-duplex sensing plus Auto Uplink™, making this unmanaged, rack-mountable switch ideal for combining 10, 100, and 1000Mbps devices. Users can take advantage of the GS516's ability to deliver large amounts of multimedia, image, and video information in no time at all. It's invaluable as a robust and reliable network backbone for your 50- to 250-employee company.

Accessible:

Plenty of bandwidth for all users, with 16 switched 10/100/1000 ports for PCs, servers, or switches.

Smart:

All 16 ports provide automatic speed and duplex sensing, plus Auto Uplink™ to adjust for straight-through or crossover cables and make the right link.

Efficient:

Each port delivers network speeds of up to 2000Mbps per port.

Straightforward:

Easy to set up and easy to use. All ports feature integrated LEDs, so network monitoring couldn't be easier.

Features:

16 10/100/1000 ports

RESOURCES

- [Order Status](#)
- [My Company](#)
- [My Account](#)
- [Account Team](#)
- [New Accounts](#)
- [Rebates](#)
- [Special Events](#)
- [CDW Outlet](#)
- [CDW Leasing](#)
- [Technical Support](#)
- [E-Newsletters](#)

ONLINEHELP
CDW eSupport

 [Printable version](#)



Manufacturer
NETGEAR
Product Links
<ul style="list-style-type: none"> > Accessories > Similar Products > Send to Associate

NetGear 8-port Copper Gigabit Switch, GS508TNA

Product Information

▶ 8-port 1000BASE-T Gigabit Copper unmanaged Layer 2 Switch

Usually Ships:	<u>Same Day</u>
CDW Part No.:	287841
Mfg. Part No.:	GS508TNA

Price: **\$719.16**

 **ADD TO CART**

▶ **OVERVIEW**

Delivers up to 11.8 million packets per second

Copper Gigabit ports in this Ethernet switch give you high-speed connectivity without the cost and hassle of fiber cables. Providing backbone for power workgroups, data centers, and server farms with convenient Plug and Play installation NETGEAR's high-performance GS508T Copper Gigabit Switch delivers the power of Copper Gigabit Ethernet to optimize your small to medium-sized business network.

Key Features:

- Eight, 10/100/1000 auto-negotiating ports
- Auto-detects optimum network speed
- Delivers up to 11.8 million packets per second
- Supports 8,000 network users or devices
- Conveniently rack-mountable

▶ **SPECIFICATIONS**

Dimensions

Weight: 5.3 lbs.
Dimensions: 13" (W) x 8.2" (D) x 1.7" (H)

LEDs

LEDs: Link Activity, Power, Link Status

Network ports

Port Type: 1000BASE-T

August 6, 2002

Hewlett-Packard
Mike Nikolaiev
MS150402
20555 SH 249
Houston, TX 77070

Dear Mike:

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

Part number	Description	Unit Price	Quantity	Price
RHF0090US	Red Hat Linux Advanced Server 2.1 (Basic)	\$800	8	\$6,400
RHF0090US	2 additional years maintenance & support	\$800	16	\$12,800
RHF099US	Red Hat Linux Personal	\$60	1	\$60
MCT0172US	Red Hat Linux Personal 8 x 3 Bundle (8 systems, 3 years support & maintenance)	\$4800	2	\$9,600

Products orderable through www.redhat.com or Red Hat Sales 1-888-REDHAT-1

Quote is valid for the next 90 days.

If we can be of any further assistance, please contact Nick Carr at ncarr@redhat.com

*Support and maintenance for software includes 30 day configuration and installation support proactive update support via Red Hat Network and product upgrades.

August 13, 2002

Raghunath K. Othayoth
ISS - Solutions and Strategy
Hewlett Packard Company
281-518-2748 tel
281-514-8375 fax

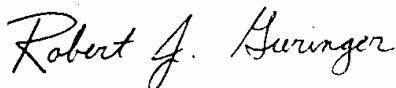
Per your request I am enclosing the pricing information regarding TUXEDO 6.5 that you requested. This pricing applies to Tuxedo 6.4, 6.5, 7.1, and 8.0. Please note that Tuxedo 8.0 is our most recent version of Tuxedo. Core functionality services pricing is appropriate for your activities. As per the 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 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,



Rob Gieringer,
Worldwide Pricing Manager

Appendix G:

Product Description	Price	Quantity	Extended Price
Oracle9i Database Enterprise Edition Release 2, per processor for 3 years	\$20,000	32	\$640,000
Real Application Clusters, per processor for 3 years	\$10,000	32	\$320,000
Partitioning, per processor for 3 years	\$5,000	32	\$160,000
Oracle Database Server Support Package for 3 years	2,000 per yr.	8	48,000
Oracle Mandatory E-Business Discount	-\$292,000	1	-\$292,000

Oracle pricing contact: MaryBeth Pierantoni
mary.beth.pierantoni@oracle.com
650-506-2118