



Hewlett-Packard Company

TPC Benchmark™ C
Full Disclosure Report
for
HP Integrity rx2600
Using
Oracle Database 10g Standard Edition One and
Red Hat Enterprise Linux AS 3

Second Edition
October 6, 2004

Second Edition October 6, 2004

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 2004 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., 2004

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

ORACLE 10i, Pro*C, PL/SQL, SQL*Net, SQL*Plus are registered trademarks of Oracle Corporation.

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

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

Table of Contents

TABLE OF CONTENTS..... 3

PREFACE..... 5

TPC BENCHMARK C OVERVIEW 5

ABSTRACT..... 6

OVERVIEW 6

TPC BENCHMARK C METRICS..... 6

STANDARD AND EXECUTIVE SUMMARY STATEMENTS 6

AUDITOR 6

GENERAL ITEMS 7

APPLICATION CODE AND DEFINITION STATEMENTS 7

TEST SPONSOR..... 7

PARAMETER SETTINGS..... 7

CONFIGURATION ITEMS..... 7

CLAUSE 1 RELATED ITEMS 9

TABLE DEFINITIONS..... 9

PHYSICAL ORGANIZATION OF DATABASE 9

Priced Configuration:..... 9

INSERT AND DELETE OPERATIONS..... 9

PARTITIONING..... 9

REPLICATION, DUPLICATION OR ADDITIONS 9

CLAUSE 2 RELATED ITEMS 10

RANDOM NUMBER GENERATION..... 10

INPUT/OUTPUT SCREEN LAYOUT 10

PRICED TERMINAL FEATURE VERIFICATION 10

PRESENTATION MANAGER OR INTELLIGENT TERMINAL..... 10

TRANSACTION STATISTICS..... 11

QUEUING MECHANISM..... 11

CLAUSE 3 RELATED ITEMS 12

TRANSACTION SYSTEM PROPERTIES (ACID)..... 12

ATOMICITY 12

Completed Transactions..... 12

Aborted Transactions..... 12

CONSISTENCY 12

ISOLATION..... 12

DURABILITY 12

Durable Media Failure..... 12

Loss of Data..... 13

Loss of Log..... 13

Instantaneous Interruption, Loss of Memory 13

CLAUSE 4 RELATED ITEMS 15

INITIAL CARDINALITY OF TABLES..... 15

DATABASE LAYOUT	15
TYPE OF DATABASE	15
DATABASE MAPPING.....	16
60 DAY SPACE	16
CLAUSE 5 RELATED ITEMS	17
THROUGHPUT	17
RESPONSE TIMES.....	17
KEYING AND THINK TIMES.....	17
RESPONSE TIME FREQUENCY DISTRIBUTION CURVES AND OTHER GRAPHS	17
STEADY STATE DETERMINATION.....	22
WORK PERFORMED DURING STEADY STATE.....	22
MEASUREMENT PERIOD DURATION	23
REGULATION OF TRANSACTION MIX	23
TRANSACTION STATISTICS.....	23
CHECKPOINT COUNT AND LOCATION.....	24
CHECKPOINT DURATION	ERROR! BOOKMARK NOT DEFINED.
CLAUSE 6 RELATED ITEMS	25
RTE DESCRIPTIONS.....	25
EMULATED COMPONENTS.....	25
FUNCTIONAL DIAGRAMS.....	25
NETWORKS.....	25
OPERATOR INTERVENTION	25
CLAUSE 7 RELATED ITEMS	26
SYSTEM PRICING.....	26
AVAILABILITY, THROUGHPUT, AND PRICE PERFORMANCE.....	26
COUNTRY SPECIFIC PRICING.....	26
USAGE PRICING.....	26
CLAUSE 9 RELATED ITEMS	27
AUDITOR'S REPORT	27
AVAILABILITY OF THE FULL DISCLOSURE REPORT	28
APPENDIX A: SOURCE CODE.....	29
APPENDIX B: DATABASE DESIGN	78
APPENDIX C: TUNABLE PARAMETERS.....	100
APPENDIX D: THIRD PARTY LETTERS.....	102
APPENDIX E: DATABASE PRICING.....	ERROR! BOOKMARK NOT DEFINED.

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 5.2, released December, 2003.

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 integrity rx2600**. The operating system used for the benchmark was Red Hat Enterprise Linux AS 3 . The DBMS used was Oracle Database 10g Standard Edition.

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:

51,506.4 tpmC

\$1.81 per tpmC

Available as of October 6, 2004.

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 Integrity rx2600

C/S with 2 ProLiant ML110

TPC-C Version 5.3

Report Date

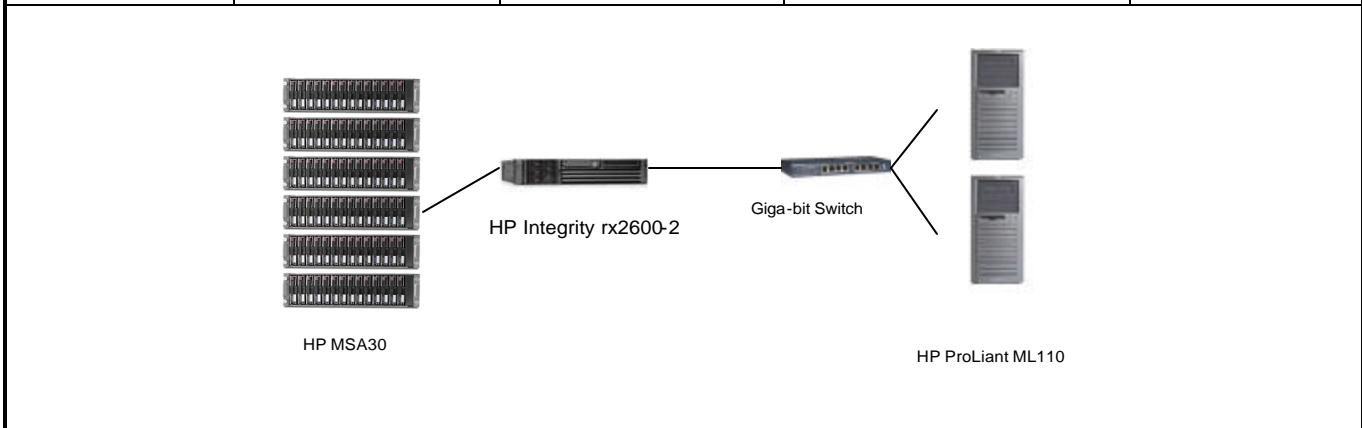
September 29, 2004

Revised Date


October 6, 2004

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date
\$93,217 USD	51,506 tpmC	\$1.81 USD/ tpmC	September 29, 2004

Processors	Database Manager	Operating System	Other Software	Number of Users
2 x 1.3GHz Intel Itanium2 3M Processors – Server 2 x Xeon 3.0GHz – Client	Oracle Database 10g Standard Edition One	Red Hat Enterprise Linux AS 3	Microsoft COM+	40800



	Server		Each Client	
System Components	Quantity	Description	Quantity	Description
Processor	2	1.3 GHz Itanium2 3M w/ 3MB Cache	2	3.0GHz Xeon w/ 256K cache
Memory	12	2GB	1	1024MB
Disk Controllers	4	HP SMART Array Controller 5302/128	1	Integrated SMART 5i Controller
	1	Integrated SCSI Controller		
Disk Drives	84	18GB 15K SCSI Drives	1	40 GB ATA Drive
	2	146GB 10K SCSI Drives		
Total Storage		1804 GB		40 GB

 invent	HP Integrity rx2600 1.30GHz/3MB 2P			TPC-C Version 5.3		
				Report Date:	29-Sep-04	
Description	Part Number	Third Party	Unit Price	Qty	Extended Price	3 yr. Maint. Price
		Brand	Pricing			
Server Hardware						
HP Integrity rx2600, Intel Itanium2 processor 1.30GHz with 3MB L3 cache	A6870B		1	5,195	1	5,195
Single power supply, 1-Year Limited Warranty, Core I/O (2NICs, Mgt LAN, U320 SCSI, RS-232 sp, VGA, USB)						
36GB Ultra 320 SCSI 15,000 rpm Hard Drive	A9896A		1	760	1	760
DVD-ROM Drive	A9919A		1	150	1	150
Intel Itanium2 processor 1.30GHz with 3MB L3 cache	A9872A		1	3,200	1	3,200
8GB PC2100 DDR SDRAM memory quad (4x2GB DIMMs)	AB228A		1	9,460	3	28,380
146GB Pluggable Ultra320 SCSI 10,000 rpm Hard Drive (DB Log)	A9898A		1	1,910	2	3,820
Smart Array 5302/128MB Cache Controller	A9825A		1	1,399	4	5,596
Smart Array Internal Cables for rx2600	A9827A		1	65	1	65
USB Keyboard kit PC-104/105 (inc. USB Kydb and USB mouse)	A7861#ABA		1	25	1	25
HP S5500 color monitor (carbon/silver 15-inch CRT)	P9006A#ABA		1	129	1	129
TR5 10/20-Gigabyte Tape Drive - Carbon	294243-B22		1	299	1	299
Pro UPS 500 (500VA/300 Watts; 110-127 VAC, 60 Hz)	136386-001		1	146	1	146
Rack Model 9142 (42U - Opal) - Flat Pallet	120663-B21		1	1,321	1	1,321
Rack Sidewall Kit	120670-B21		1	207	1	207
3YR 24X7/4HR for HP Integrity rx2600	HA110A3-6BT		1	2,488	1	2,488
Server Hardware Subtotal					49,293	2,488
External Storage						
StorageWorks Enclosure Model MSA 30 SB - Rack-mountable	302969-B21		1	2,978	6	17,868
18.2GB 15Krpm U320 UNI HDD	286775-B22		1	269	84	22,596
18.2GB 15Krpm U320 UNI HDD (10% spares)	286775-B22		1	269	9	2,421
FM-4E724-36 3YR 24X7/4HR EMPTY DISK ENCL	171242-002			157	6	942
Server External Storage Subtotal					40,464	3,363
Server Software						
Red Hat Enterprise Linux AS 3.0	T2744AA-324	HP	1	1,908	1	1,908
3 Yrs 24 x 7 Support Contract for Red Hat Enterprise Linux AS 3.0	HA110A3-6L4	HP	1	4,839	1	4,839
Oracle Database 10g Standard Edition One for 3 years, Per Processor, Unlimited Users	run-time	Oracle	2	2,498	2	4,995
Oracle Database Server Support Package for 3 years	run-time	Oracle	2	2,000	3	6,000
Server Software Subtotal					6,903	10,839
Client Hardware						
HP ProLiant ML110 - ATA Model, 3.0GHz/800MHz, 1GB, 40GB, 24x7 4-Hr, 3Yrs	359661-xx1		1	1,315	2	2,630
PCI Gigabit NIC (embedded) 10/100/1000 WOL (Wake on LAN)						896
Client Hardware Subtotal					2,630	896
Client Software						
Microsoft Windows 2000 Server	C11-00821	Microsoft	4	738	2	1,476
Client Software Subtotal					1,476	Included
User Connectivity						
NetGear GS105 5-port Gigabit Ethernet Switch 10/100/1000 Mbps	GS105	NetGear	5	70	3	210
Connectivity Subtotal					210	
Large Purchase and Net 30 discount	30.0%		1			(\$746)
Large Purchase and Net 30 discount	16.0%		1			(\$681)
Large Purchase and Net 30 discount	23.0%		1			(\$439)
Total					\$79,068	\$14,149
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:						\$93,217 USD
tpmC Rating:						51.506
\$ / tpmC:						\$1.81 USD
Oracle pricing contact: Mary Beth Pierantoni, mary.beth.pierantoni@oracle.com, 916-315-5081						
Note 1 : Discount based on HP Direct guidance with large purchase discount.						
The benchmark results and test methodology audited by Lorna Livingtree of Performance Metrics, Inc.						

Numerical Quantities Summary

MQTH, Computed Maximum Qualified Throughput

51506.4 tpmC

Response Times (in seconds)	Average	90%	Maximum
New-Order	0.168	0.202	
Payment	0.154	0.186	1.812
Order-Status	0.164	0.201	1.383
Delivery (interactive portion)	0.102	0.102	0.978
Delivery (deferred portion)	0.138	0.173	0.117
Stock-Level	0.061	0.108	1.005
Menu	0.102	0.102	0.625

Transaction Mix, in percent of total transaction

New-Order	44.914%
Payment	43.021%
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	18.005/0.00	18.008/12.205	18.021/1250.266
Payment	3.010/0.00	3.018/12.014	3.031/1251.333
Order-Status	2.010/0.00	2.018/10.015	2.029/100.037
Delivery (interactive)	2.010/0.00	2.018/5.025	2.029/50.078
Stock-Level	2.010/0.00	2.018/5.015	2.030/50.082

Test Duration

Ramp-up time	80 minutes
Measurement interval	120 minutes
Transactions (all types) completed during measurement interval	13,761,271
Ramp down time	129 minutes

Checkpointing

Number of checkpoints	5
Checkpoint interval	23 minutes

General Items

Application Code and Definition Statements

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

Appendix A contains all source code implemented in this benchmark.

Test Sponsor

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

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

Parameter Settings

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

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

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

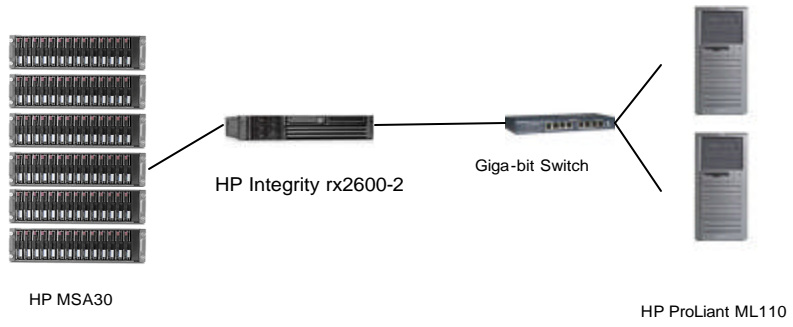
Appendix C contains the tunable parameters for the database, the operating system, and the transaction monitor.

Configuration Items

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

The configuration diagram for both the tested and priced system are the same and included on the following page

Figure 1. Benchmarked and Priced Configuration



Clause 1 Related Items

Table Definitions

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

Physical Organization of Database

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

84 disks used in the benchmark had a capacity of 18.2GB 15K rpm, and 2 disks used in the benchmark had a capacity of 146.8 GB 10K rpm.

Controller	Storage	Unformatted Capacity	Contents
1. hp SMART Array Controller 5304	2 Storageworks Enclosure Model 4314R (2 x 14 x 18.2 GB 15K rpm disk drives)	254.8GB	Tables, Indexes
2. hp SMART Array Controller 5304	2 Storageworks Enclosure Model 4314R (2 x 14 x 18.2 GB 15K rpm disk drives)	254.8GB	Tables, Indexes
3. hp SMART Array Controller 5304	2 Storageworks Enclosure Model 4314R (2 x 14 x 18.2 GB 15K rpm disk drives)	254.8GB	Tables, Indexes
4. hp SMART Array Controller 5304	Integrity rx2600 internal drive bay (2 x 146.8 GB 10K rpm disk drives)	293.6GB	Redo Logs

Priced Configuration:

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

Insert and Delete Operations

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

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

Partitioning

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

None.

Replication, Duplication or Additions

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

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

Clause 2 Related Items

Random Number Generation

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

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

Input/Output Screen Layout

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

All screen layouts followed the specifications exactly.

Priced Terminal Feature Verification

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

The terminal attributes were verified by the auditor manually exercising each specification on a representative ProLiant ML110.

Presentation Manager or Intelligent Terminal

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

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

Transaction Statistics

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

Table 2.1 Transaction Statistics

Statistic		Value
New Order	Home warehouse order lines	99.00%
	Remote warehouse order lines	1.00%
	Rolled back transactions	1.00%
	Average items per order	10.00
Payment	Home warehouse	85.01%
	Remote warehouse	14.99%
	Accessed by last name	60.00%
Order Status	Accessed by last name	69.99%
Delivery	Skipped transactions	None
Transaction Mix	New Order	44.914%
	Payment	43.021%
	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.

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

Clause 3 Related Items

Transaction System Properties (ACID)

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

All ACID property tests were successful. The executions are described below.

Atomicity

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

Completed Transactions

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

Aborted Transactions

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

Consistency

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

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

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

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

Isolation

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

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

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 4080 warehouses. The standard driving mechanism was used to generate the transaction load of 40800 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 40800 users
4. The test was allowed to run for a minimum of 10 minutes.
5. The backed up partition was overwritten with garbage information.
6. Oracle10g 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. The database was opened and Oracle 10g performed instance recovery.
9. Consistency conditions were executed and verified.
10. Step 2 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.

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 4080 users.
3. The test was allowed to run for a minimum of 10 minutes.
4. A log disk containing log information 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

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 4080 warehouses under a full load of 40800 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 40800 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. Oracle10g 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.

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	4320
District	43200
Customer	129600000
History	129600000
Order	129600000
New Order	38880000
Order Line	1295979368
Stock	432000000
Item	100000
Unused Warehouses	240

Database Layout

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

The benchmarked configuration used three Smart Array 5304 Controllers attached to two StorageWorks Enclosure MSA30s with 14 18.2 disk drives each for the database, one three Smart Array 5304 Controllers attached to internal storage array with 2 146.6 disk drives for redo log. 2 Array accelerator cache for data volumes were set to 100% write, and disabled for redolog volume.

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

Type of Database

A statement must be provided that describes:

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

Oracle Database 10g Edition is a relational DBMS.

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

Database Mapping

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

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

60 Day Space

Details of the 60 day space computations along with proof that the database is configured to sustain 8 hours of growth for the dynamic tables (Order, Order-Line, and History) must be disclosed.

SEGMENT	BLOCKS	BLOCK_SIZE	REQUIRED	STATIC	DYNAMIC	OVERSIZE
CUSTCLUSTER	57784320	2048	47708325	47708325	0	10075995
DISTCLUSTER	51200	2048	47712	47712	0	3488
HIST	5775360	2048	5285009	0	4441184	490351
ICUST1	4096000	2048	1750676	1750676	0	2345324
ICUST2	4300800	2048	4085626	4085626	0	215174
IDIST	15360	2048	11878	11878	0	3482
IITEM	10240	2048	5914	5914	0	4326
IORDR2	3471360	2048	2908280	2908280	0	563080
ISTOK	8192000	2048	5159939	5159939	0	3032061
ITEMCLUSTER	10240	2048	8868	8868	0	1372
IWARE	10240	2048	5914	5914	0	4326
NORDCLUSTER_QUEUE	757760	2048	554131	554131	0	203629
ORDRCLUSTER_QUEUE	9826560	16384	8690948	0	7303318	1135612
STOKCLUSTER	65940480	2048	64881365	64881365	0	1059115
SYSAUX	61440	2048	61440	61440	0	0
SYSTEM	204800	2048	204800	204800	0	0
WARECLUSTER	10240	2048	5914	5914	0	4326
	Disk Capacity	Quantity	Total			
	18.20	84	1528.80			
	146.00	2	292.00			
	36.40	1	36.40			
			1857.20			
		Space Required	Space Configured			
	Data	1410.3	1528.8			
	Log	208.6	292.0			

Clause 5 Related Items

Throughput

Measured tpmC must be reported

Measured tpmC 51506.4 tpmC

Price per tpmC \$1.81 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	0.168	0.202	
Payment	0.154	0.186	1.812
Order-Status	0.164	0.201	1.383
Interactive Delivery	0.102	0.102	0.978
Deferred Delivery	0.138	0.173	0.117
Stock-Level	0.061	0.108	1.005
Menu	0.102	0.102	0.625

Keying and Think Times

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

Table 5.2: Keying Times/Think Times

Type	Minimum	Average	Maximum
New-Order	18.005/0.00	18.008/12.205	18.021/1250.266
Payment	3.010/0.00	3.018/12.014	3.031/1251.333
Order-Status	2.010/0.00	2.018/10.015	2.029/100.037
Interactive Delivery	2.010/0.00	2.018/5.025	2.029/50.078
Stock-Level	2.010/0.00	2.018/5.015	2.030/50.082

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

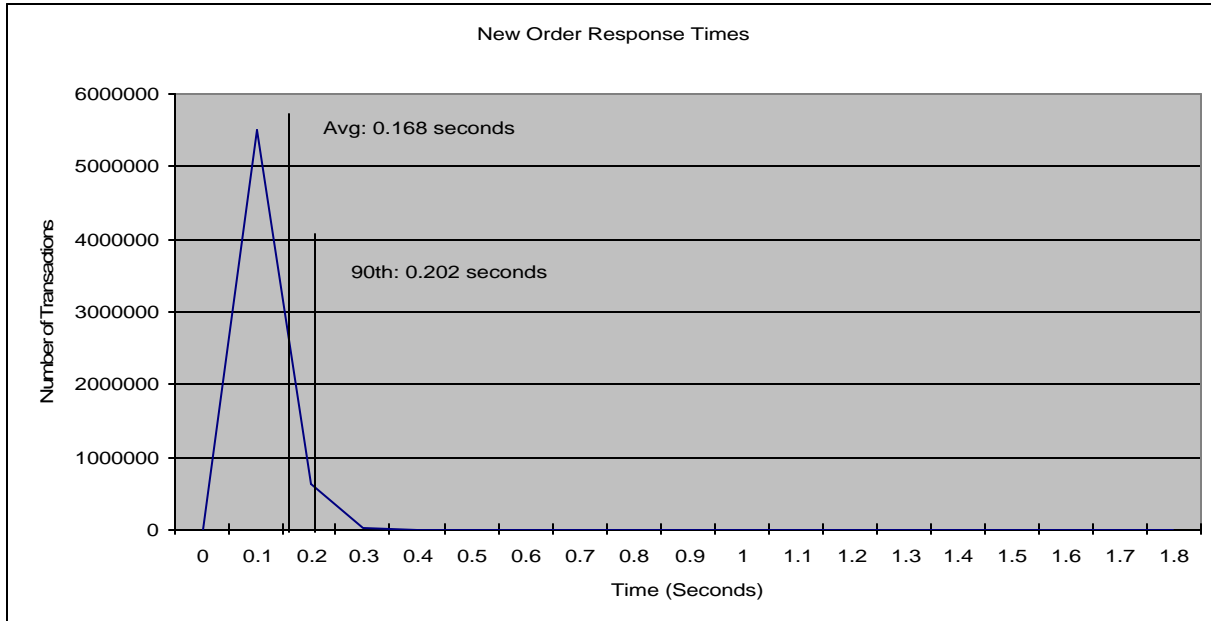


Figure 5.2: Response Times Frequency Distribution for Payment Transactions

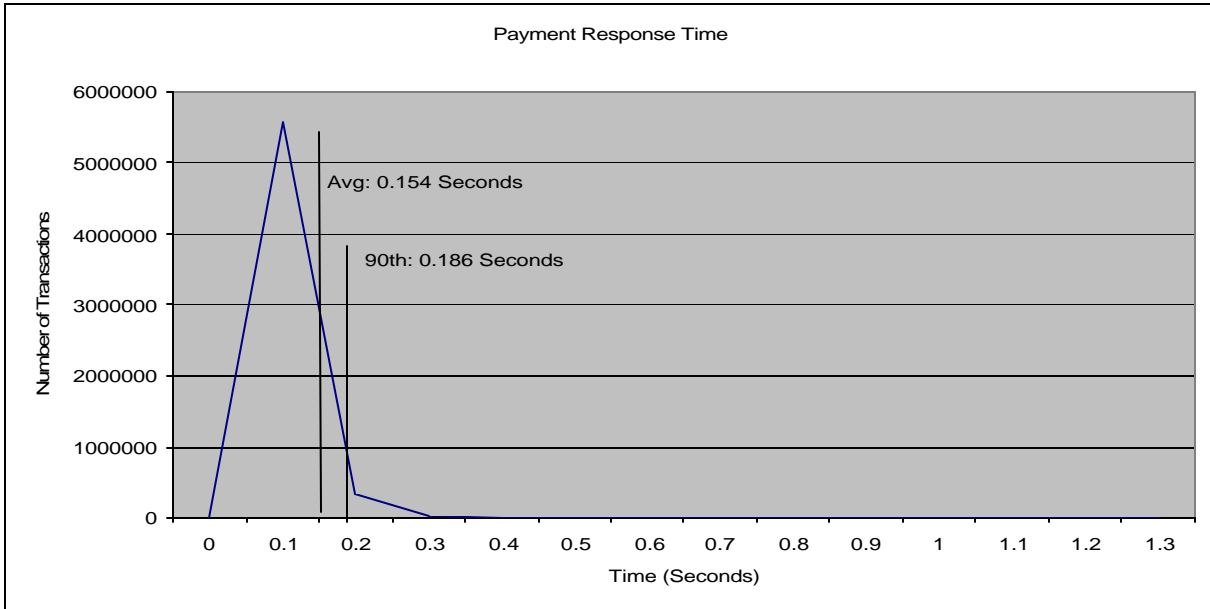


Figure 5.3: Response Times Frequency Distribution for Order Status Transactions

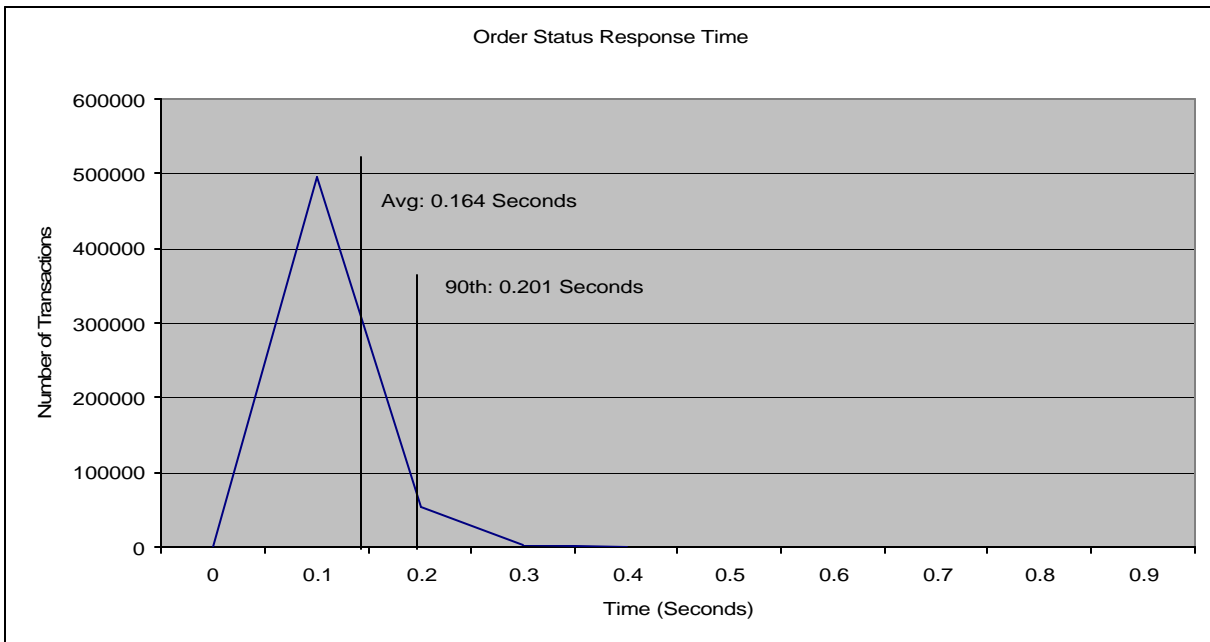


Figure 5.4: Response Times Frequency Distribution for Delivery Transactions

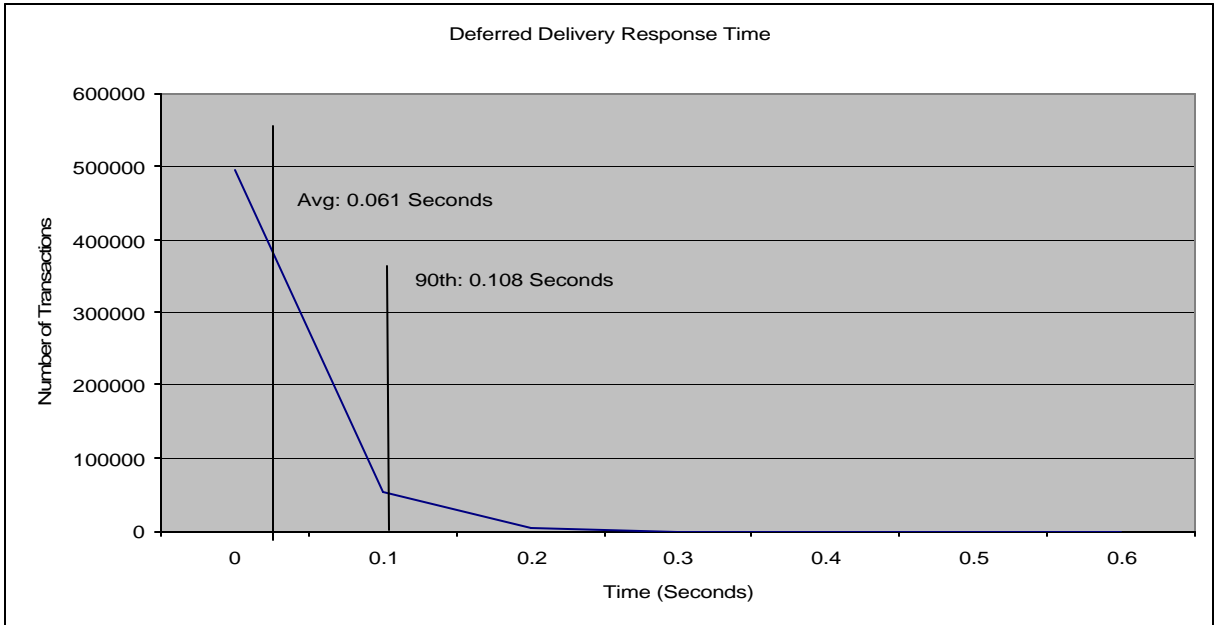


Figure 5.5: Response Times Frequency Distribution for Stock Level Transactions

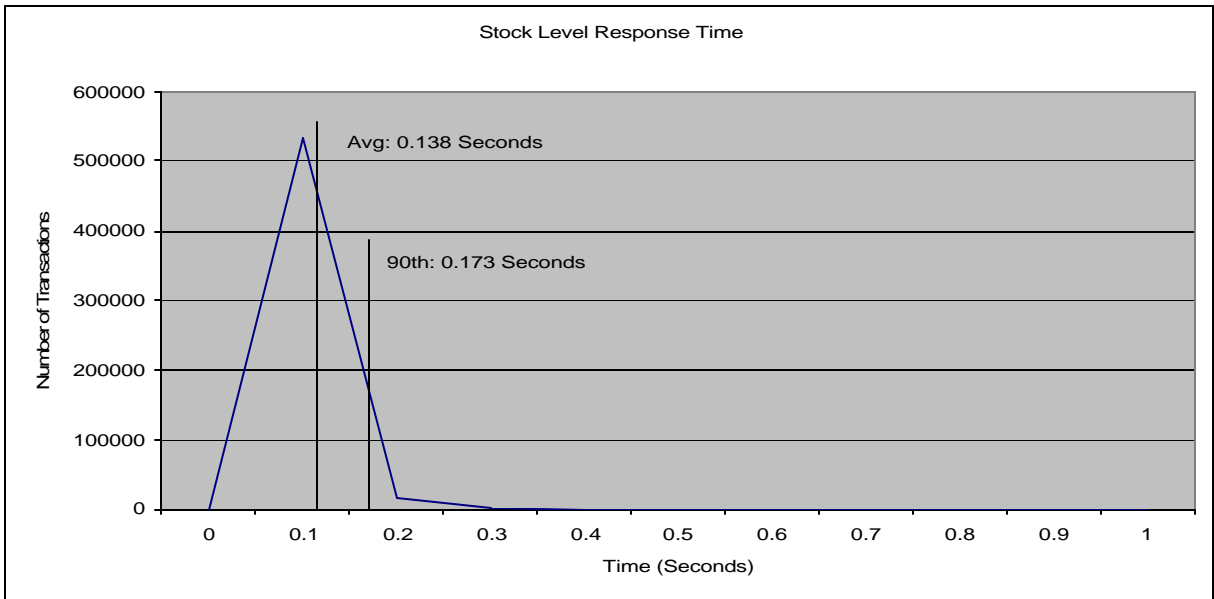


Figure 5.6: Response Time versus Throughput

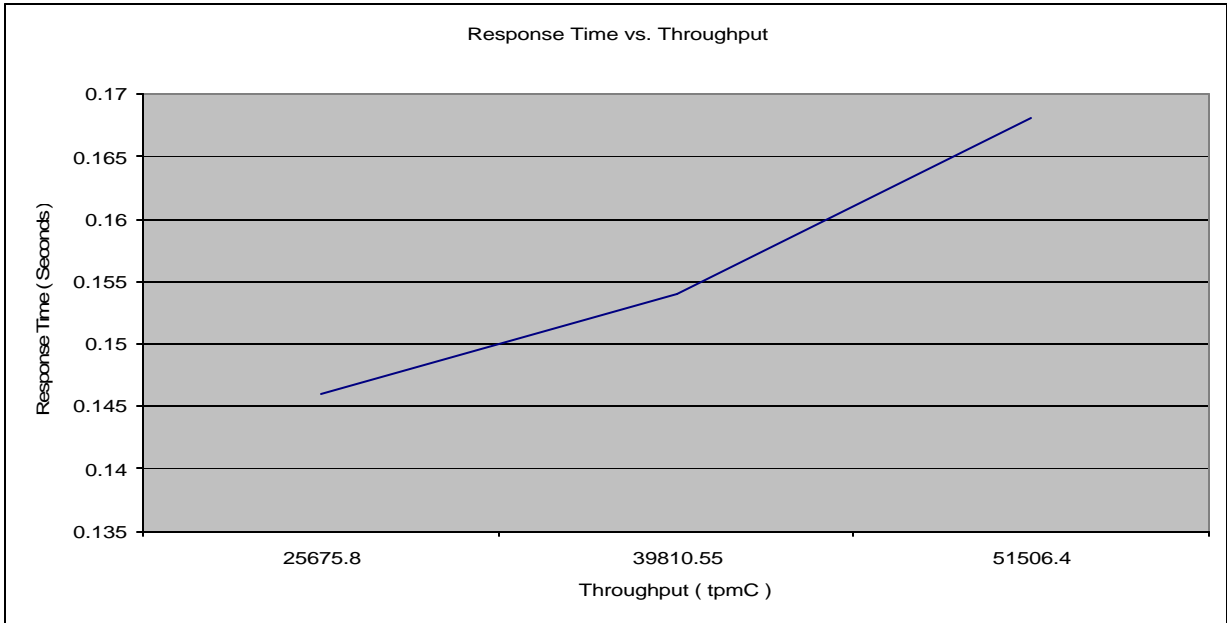


Figure 5.7: Think Times distribution for New Order Transactions

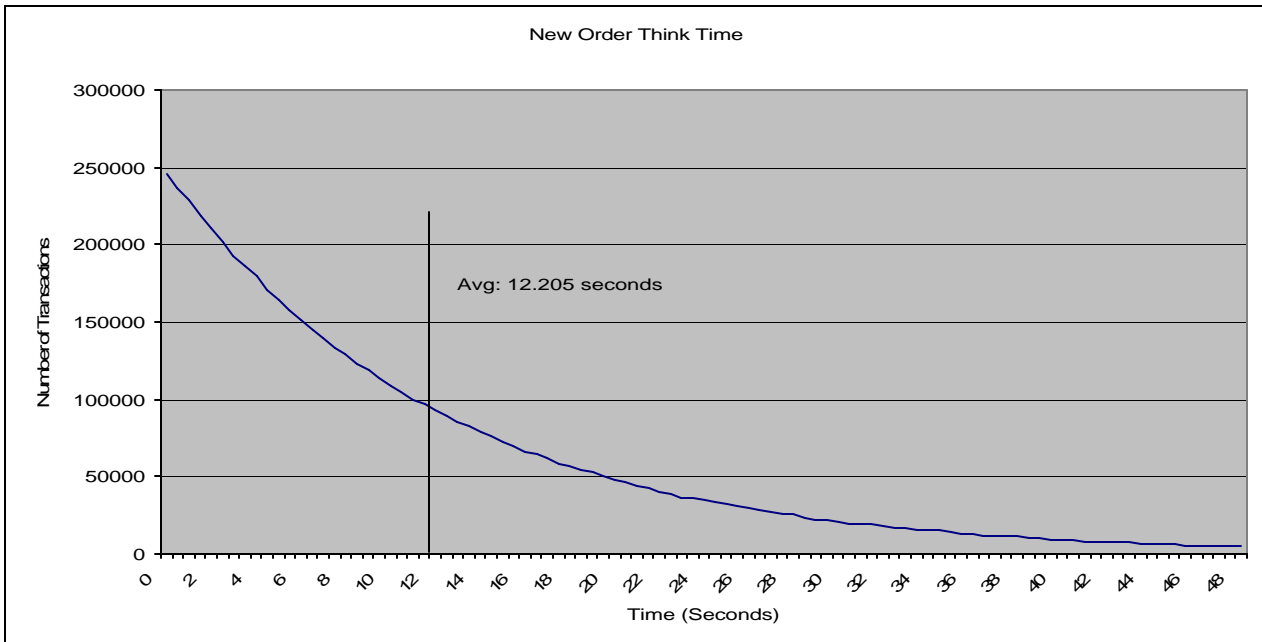
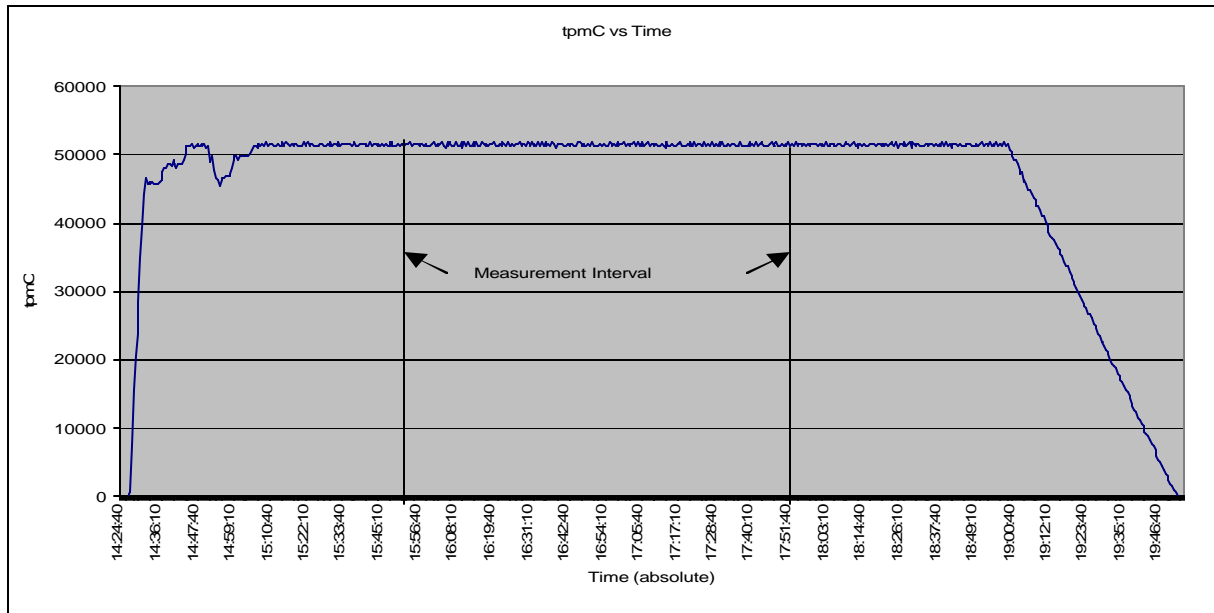


Figure 5.8: Throughput versus Time



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 7200 seconds.

Regulation of Transaction Mix

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

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

Transaction Statistics

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

Table 5.3: Transaction Statistics

Statistic		Value
New Order	Home warehouse order lines	99.00%
	Remote warehouse order lines	1.00%
	Rolled back transactions	1.00%
	Average items per order	10.00
Payment	Home warehouse	85.01%
	Remote warehouse	49.99%
	Accessed by last name	60.00%
Order Status	Accessed by last name	59.99%

Statistic		Value
Delivery	Skipped transactions	0
Transaction Mix	New Order	44.914%
	Payment	43.021%
	Order status	4.020%
	Delivery	4.025%
	Stock level	4.020%

Checkpoint

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

Oracle database was set for incremental checkpoint using log_checkpoint_timeout parameter (maximum time interval between checkpoints in seconds) which was set to 1200 seconds. One checkpoint occurred during the warm-up period and 5 checkpoints occurred during the measurement period.

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 2 ProLiant DL580 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, the server system and two client systems were connected to 8 port 1000BaseT Ethernet switch.

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

Operator Intervention

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

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

Clause 7 Related Items

System Pricing

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

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

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

Availability, Throughput, and Price Performance

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

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

- **Maximum Qualified Throughput** 51506.4 tpmC
- **Price per tpmC** \$1.81 per tpmC
- **Available** October 6, 2004

All components are available now.

Country Specific Pricing

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

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

Usage Pricing

For any usage pricing, the sponsor must disclose:

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

The component pricing based on usage is shown below:

- Oracle Database 10g Standard Edition One
- Red Hat Enterprise Linux AS 3.0
- Microsoft Windows 2000 Server

Clause 9 Related Items

Auditor's Report

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

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

Performance Metrics, Inc. 137 Yankton St. #101 Folsom, CA 95630
phone: 916-985-1131 fax: 916-985-1185 email: lorna@perfmetrics.com



PERFORMANCE METRICS INC.
TPC Certified Auditors

September 28, 2004

Raghunath Othayoth , Bryon Georgson, and
Paul Cao
Hewlett-Packard Company
20555 SH 249
Houston, TX 77070

I have verified by remote the TPC Benchmark™ C for the following configuration:

Platform: HP Integrity rx2600
Database Manager: Oracle 10g Standard Edition
Operating System: Red Hat Linux AS 3
Transaction Monitor: Microsoft COM+

System Under Test:				
CPU's	Memory	Disks (total)	90% Response	TpmC
2 Itanium2 @ 1.3 Ghz	Main: 24 GB	84 @ 18 GB 2 @ 146 GB	0.20	51,506

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

- The transactions were correctly implemented.
- The database files were properly sized.
- The database was properly scaled with 4320 warehouses, 4080 of which were active during the measured interval.
- The ACID properties were successfully demonstrated.
- Input data was generated according to the specified percentages.
- Eight hours of mirrored log space was present on the tested system.
- Eight hours of growth space for the dynamic tables was present on the tested system.
- The data for the 60 days space calculation was verified.
- The steady state portion of the test was 120 minutes.
- There was one complete checkpoint in steady state before the measured interval.

PERFORMANCE METRICS INC.
TPC Certified Auditors

- There were 5 checkpoints completed inside the measured interval.
- The system pricing was checked for major components and maintenance.
- Third party quotes were verified for compliance.

Auditor Notes: None

Sincerely,

A handwritten signature in cursive script that reads "Lorna Livingtree".

Lorna Livingtree
Auditor

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

TPC Benchmark C Full Disclosure Reports are available at www.tpc.org

Appendix A: Source Code

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

#ifdef HISTORY

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

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

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

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

Fixed a bug where the critical section was not
released when a buffer read or write operation
timed out.
*/

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

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

#include "buf.h"

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

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

    *bufptr = buf;
    return(BUF_SUCCESS);
}

static void
calcstoredsize(BUFPTR buf, size_t *storedsize, size_t
*storedsizehigh)
{
    if( buf->storedstart < buf->freestart ) {
```

```
        *storedsizehigh = *storedsize = buf->freestart - buf-
>storedstart;
    }
    else if( buf->full || buf->storedstart > buf->freestart ) {
        *storedsizehigh = ((uchar *) buf->buf + buf->size) - buf-
>storedstart;
        *storedsize = *storedsizehigh + (buf->freestart - (uchar
*)buf->buf);
    }
    else {
        *storedsizehigh = *storedsize = 0;
    }
    return;
}

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

    if( btr <= 0 )
        return BUF_SUCCESS;

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

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

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

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

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

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

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

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

```

    LeaveCriticalSection( &buf->control );

    *br = btr;
}
return BUF_SUCCESS;
}

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

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

    if( btw <= 0 )
        return BUF_SUCCESS;

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

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

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

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

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

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

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

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

    LeaveCriticalSection( &buf->control );

    *bw = btw;

    return BUF_SUCCESS;
}

```

```

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

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

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

#ifndef _buf_h_
#define _buf_h_

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

#define BUF_INFINITE INFINITE

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

#define BUF_MINSIZE 4

typedef unsigned int uint;
typedef unsigned char uchar;

struct _buf
{
    uchar *freestart;
    uchar *storedstart;
    size_t size;
    uchar *maxplus1;
    BOOL full;
    int blockedreadercount;
    int blockedwritercount;
    CRITICAL_SECTION control;
    HANDLE dataready;
    HANDLE spacefreed;
    char buf[BUF_MINSIZE]; /* MUST BE AT END for malloc to
succeed */
};
typedef struct _buf BUF, *BUFPTR;

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

#endif

-----
code.txt

```

```

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

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

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

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

    if (ul_reason_for_call == DLL_PROCESS_ATTACH) {
        DisableThreadLibraryCalls((HMODULE)hModule);

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

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

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

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

        TotalLoop=DBConnections*2;

        DBExecution_lock=(HANDLE*)malloc(sizeof(HANDLE)*DBConnections);
        for (i=0; i<DBConnections; i++)
            if ((DBExecution_lock[i]=CreateMutex(NULL, FALSE,
            NULL))==NULL) {
                userlog("Cannot create mutex : DBExecution_lock[%d]\n",
            i);
                return FALSE;
            }

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

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

        ready=1;
    }
    else if (ul_reason_for_call == DLL_PROCESS_DETACH) {
        if ((TlsFree(TlsPtr)) == NULL) {
            userlog("Error during TlsFree\n");
            return FALSE;
        }

        for (int i=0; i<DBConnections; i++) {
            (DBExecution *) (DBExecution_pool[i].pointer)->TPCexit();
            free(DBExecution_pool[i].pointer);
        }
        free (DBExecution_pool);
        CloseHandle(waitIdle);

        for (i=0; i<DBConnections; i++)
            CloseHandle(DBExecution_lock[i]);
    }

    return TRUE;
}

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

    DelFiles=(FILE **)malloc(sizeof(FILE *)*DelThreads);

```

```

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

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

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

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

#ifdef LOOPBACK

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

    DBExecution *dbexec;
    struct newstruct input;

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

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

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

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

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

    output->status = dbexec->status;

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

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

```

```

    return SUCCESS;
}

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

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

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

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

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

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

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

    return SUCCESS;
}

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

    input.delin.w_id = output->w_id;
    input.delin.plsqlflag = 1;
    input.delin.o_carrier_id = output->o_carrier_id;
    output->dequeue_time = (double) GetTickCount();

#ifdef CONNECTION_MUTEX
    ptr=findIdleDBExecution(mutexptr);
#else
    ptr=findIdleDBExecution();
#endif

```

```

    dbexec=(DBExecution *) (ptr->pointer);
    // ptr->delivery_count++;

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

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

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

    write_delivery_log(output, id);

    return SUCCESS;
}

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

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

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

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

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

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

    return SUCCESS;
}

```

```

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

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

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

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

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

    output->low_stock = input.stoout.low_stock;

    return SUCCESS;
}

void write_delivery_log(T_delivery_data *pdata, int threadId)
{
    fprintf(DelFiles[threadId],
        "Q %1d %1d %6.5f %6.5f %1d %1d %1d %1d %1d %1d\n",
        %1d %1d %1d %1d %1d\n",
        pdata->w_id, pdata->ld_id, pdata->enqueue_time,
        pdata->complete_time - pdata->enqueue_time,
        pdata->dequeue_time - pdata->enqueue_time, pdata->
>txn_status,
        pdata->o_id[0], pdata->o_id[1], pdata->o_id[2], pdata->
>o_id[3],
        pdata->o_id[4], pdata->o_id[5], pdata->o_id[6], pdata->
>o_id[7],
        pdata->o_id[8], pdata->o_id[9]);
}

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

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

#ifdef CONNECTION_MUTEX
    if (mutexptr==NULL)
        userlog("Thread %d has mutexptr=NULL\n",
GetCurrentThreadId());
        ReleaseMutex((*mutexptr));
#else
    if (!SetEvent(waitIdle)) {
        userlog("Error on SetEvent, in function: free
DBExecution\n");
        return FALSE;
    }
}

return TRUE;
}

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

#ifdef DEBUG
    findDBExecutionCall++;
}

```

```

#endif

while (1) {
    for (int count=0; count<TotalLoop; count++) {
        if (DBExecution_pool[current].current_status == IDLE) {
            switch(WaitForSingleObject(DBExecution_lock[current],
0)) {
                case WAIT_ABANDONED:
#ifdef DEBUG
                    userlog("connection mutex returns
WAIT_ABANDONED\n");
#endif
                case WAIT_OBJECT_0:
#ifdef DEBUG_DETAIL
                    userlog("Thread %d get connection: %d\n",
GetCurrentThreadId(), current);
#endif
                    if (DBExecution_pool[current].current_status ==
IDLE) {
                        DBExecution_pool[current].current_status = IN_USE;
#ifdef CONNECTION_MUTEX
                        ReleaseMutex(DBExecution_lock[current]);
#else
                        mutexptr=&(DBExecution_lock[current]);
#endif
                        TlsSetValue(TlsPtr, (void *)
DBExecution_pool[current].pointer);
                        return &(DBExecution_pool[current]);
                    }
                    else {
                        ReleaseMutex(DBExecution_lock[current]);
#ifdef DEBUG
                        userlog("get connection mutex, but current_status
is not IDLE\n");
#endif
                    }
                    break;

                case WAIT_TIMEOUT:
                    break;

                default:
                    userlog("Error on WaitForSingleObject,
DBExecution\n");
                    return NULL;
            }
        }
        current++;
        if (current==DBConnections) current=0;
    }

#ifdef DEBUG
    findDBExecutionWait++;
    if (findDBExecutionWait !=0 && findDBExecutionWait % 10000
== 0)
        userlog("wait: %d, total call: %d\n",
findDBExecutionWait, findDBExecutionCall);
#endif

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

    return NULL;
}

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

int initializeDBExecutionPool()
{
    DBExecution *ptr;

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

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

```

```

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

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

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

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

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

return TRUE;
}

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

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

    threadId = GetCurrentThreadId();
    time (&t);
    currtime = localtime(&t);
    if ((file=fopen(LogFile, "a"))==(FILE *) NULL) {
        fprintf(stderr, "Can't open file : %s\n", LogFile);
        exit(-1);
    }
    va_start(va, str);
    fprintf(file, "[Time %d:%d:%d Thread: %d] ", currtime-
>tm_hour, currtime->tm_min, currtime->tm_sec, threadId);
    vfprintf(file, str, va);
    fprintf(file, "\n");
    va_end(va);
    fclose(file);

    ReleaseMutex(logMutex);
    CloseHandle(logMutex);
}

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

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

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

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

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

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

```

```

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

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

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

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

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

DBExecution::~DBExecution()
{
}

#define SQLTXNEW2 "BEGIN inittppc.init_no(:idxlarr); END;"
#define SQLTXDEL "BEGIN inittppc.init_del ; END;"
#define SQLTXDEL1 "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 SQLTXDEL3 "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 SQLTXDEL4 "UPDATE ordl \
SET ol_delivery_d = :cr_date \
WHERE ol_w_id = :w_id AND ol_d_id = :d_id AND ol_o_id = :o_id
\
RETURNING sum(ol_amount) into :ol_amount "

#define SQLTXDEL6 "UPDATE cust SET c_balance = c_balance +
:amt, \
c_delivery_cnt = c_delivery_cnt + 1 WHERE c_w_id = :w_id AND
\
c_d_id = :d_id AND c_id = :c_id"

#define SQLCUR0 "SELECT rowid FROM cust \
WHERE c_d_id = :d_id AND c_w_id = :w_id AND
c_last = :c_last \
ORDER BY c_last, c_d_id, c_w_id, c_first"

#define SQLCUR1 "SELECT /*+ USE_NL(cust) INDEX_DESC(ordr iordr2)
*/ \
c_id, c_balance, c_first, c_middle, c_last, \
o_id, o_entry_d, o_carrier_id, o_ol_cnt \
FROM cust, ordr \
WHERE cust.rowid = :cust_rowid \
AND o_d_id = c_d_id AND o_w_id = c_w_id AND
o_c_id = c_id \
ORDER BY o_c_id DESC, o_d_id DESC, o_w_id DESC,
o_id DESC"

#define SQLCUR2 "SELECT /*+ USE_NL(cust) INDEX_DESC (ordr
iordr2) */ \
c_balance, c_first, c_middle, c_last, \
o_id, o_entry_d, o_carrier_id, o_ol_cnt \
FROM cust, ordr \
WHERE c_id = :c_id AND c_d_id = :d_id AND
\
AND o_d_id = c_d_id AND o_w_id = c_w_id AND
o_c_id = c_id \
ORDER BY o_c_id DESC, o_d_id DESC, o_w_id DESC
, o_id DESC"

#define SQLCUR3 "SELECT /*+ INDEX(ordl) */ \

```

```

ol_delivery_d \      ol_i_id, ol_supply_w_id, ol_quantity, ol_amount,
FROM ordl \
WHERE ol_o_id = :o_id AND ol_d_id = :d_id AND
ol_w_id = :w_id"
#define SQLCUR4 "SELECT count(c_last) FROM cust \
WHERE c_d_id = :d_id AND c_w_id = :w_id AND
c_last = :c_last "
#ifdef PLSQLSTO
#define SQLTXTSTO "BEGIN stocklevel.getstocklevel (:w_id, :d_id,
:threshold, \
:low_stock); END;"
#else
#define SQLTXTSTO "SELECT /* nocache (stok) */ count (DISTINCT
s_i_id) \
FROM ordl, stok, dist \
WHERE d_id = :d_id AND d_w_id = :w_id AND \
d_id = ol_d_id AND d_w_id = ol_w_id AND \
ol_i_id = s_i_id AND ol_w_id = s_w_id AND \
s_quantity < :threshold AND \
ol_o_id BETWEEN (d_next_o_id - 20) AND
(d_next_o_id - 1) \
order by ol_o_id desc"
#endif

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

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

sprintf(realfile,"%s",fnam);
fd = fopen(realfile,"r");
if (!fd){
fprintf(stderr," fopen on %s failed %d\n",fnam,fd);
exit(-1);
}
while (fgets((char *)linebuf+nulpt, SQL_BUF_SIZE,fd))
nulpt = strlen((char *)linebuf);

return(nulpt);
}

int DBExecution::ocierror(char *fname, int lineno, OCIError
*errhp, sword status)
{
text errbuf[512];
sb4 errcode;
sb4 lstat;
ub4 recno=2;

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

```

```

userlog("ocierror: Module %s Line %d\n", fname, lineno);
userlog("ocierror: Error - OCI_CONTINUE\n");
return (IRRECERR);
default:
userlog("ocierror: Module %s Line %d\n", fname, lineno);
userlog("ocierror: Status - %s\n", status);
return (IRRECERR);
}
return (RECOVERR);
}

/*****
*****
* TPCinit TPCexit
*
*****
*****
DBExecution::TPCinit (int id, char *uid, char *pwd)
{
#ifdef LOOPBACK
text stmbuf[100];

#define SQLTXT "alter session set isolation_level =
serializable"
#define SQLTXTTRC "alter session set sql_trace = true"
#define SQLTXTTIM "alter session set timed_statistics = true"
#define SQLTXTTOPS "alter session set current_schema = tpcc"

proc_no = id;
/*
char *temp;

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

OCIInitialize(OCI_DEFAULT|OCI_OBJECT,(dvoid *)0,0,0); /*
// OCIERROR(errhp, OCIInitialize(OCI_THREADED|OCI_OBJECT,(dvoid
*)0,0,0));
OCIERROR(errhp, OCIEnvInit(&tpcenv, OCI_DEFAULT, 0, (dvoid
**0));
OCIERROR(errhp, OCIHandleAlloc((dvoid *)tpcenv, (dvoid
**)&tpcsrv, OCI_HTYPE_SERVER, 0, (dvoid **)0));
OCIERROR(errhp, OCIHandleAlloc((dvoid *)tpcenv, (dvoid
**)&errhp, OCI_HTYPE_ERROR, 0, (dvoid **)0));
OCIERROR(errhp, OCIHandleAlloc((dvoid *)tpcenv, (dvoid
**)&tpcsvc, OCI_HTYPE_SVCCTX, 0, (dvoid **)0));

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

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

if (i==100) {
userlog("Can't attach to Server after 100 tries\n");
return -1;
}

OCIERROR(errhp, OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX,
(dvoid *)tpcsrv, (ub4)0,OCI_ATTR_SERVER, errhp));
OCIERROR(errhp, OCIHandleAlloc((dvoid *)tpcenv, (dvoid
**)&tpcusr, OCI_HTYPE_SESSION, 0, (dvoid **)0));
#ifdef OPS_LOGIN
OCIERROR(errhp, OCIEnvInit(tpcsvc, errhp, tpcusr,
OCI_CRED_EXT, OCI_DEFAULT));
#else
OCIERROR(errhp, OCIAttrSet((dvoid *)tpcusr,
OCI_HTYPE_SESSION, (dvoid *)uid,
(ub4)strlen(uid),OCI_ATTR_USERNAME, errhp));
OCIERROR(errhp, OCIAttrSet((dvoid *)tpcusr,
OCI_HTYPE_SESSION, (dvoid *)pwd, (ub4)strlen(pwd),
OCI_ATTR_PASSWORD, errhp));
OCIERROR(errhp, OCIEnvInit(tpcsvc, errhp, tpcusr,
OCI_CRED_RDBMS, OCI_DEFAULT));
#endif

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

/* run all transaction in serializable mode */

OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0,
(dvoid**)0);
sprintf ((char *) stmbuf, SQLTXT);
OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);
OCIERROR(errhp,OCIStmtExecute(tpcsvc, curi,
errhp,1,0,0,OCI_DEFAULT));

OCIHandleFree(curi, OCI_HTYPE_STMT);

#ifdef OPS_LOGIN
OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0,
(dvoid**)0);
memset(stmbuf,0,100);
sprintf ((char *) stmbuf, SQLTXTTOPS);
OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);

```



```

OCIERROR(errhp, OCIStmtExecute(tpcsvc, curi,
errhp,1,0,0,OCI_DEFAULT));
OCIHandleFree((dvoid *)curi, OCI_HTYPE_STMT);
#endif

if (tracelevel == 3) {
OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0,
(dvoid**)0);
memset(stmbuf,0,100);
sprintf((char *) stmbuf, SQLTXXTIM);
OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);
OCIERROR(errhp, OCIStmtExecute(tpcsvc, curi,
errhp,1,0,0,OCI_DEFAULT));
OCIHandleFree((dvoid *)curi, OCI_HTYPE_STMT);
}

logon = 1;
OCIERROR(errhp,OCIDateSysDate(errhp,&cr_date));

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

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

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

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

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

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

#endif

return (0);
}

void DBExecution::TPCexit()
{
#ifdef LOOPBACK
if (new_init) {
tkvcndone();
new_init = 0;
}
if (pay_init) {
tkvcpdone();
pay_init = 0;
}
if (ord_init) {
tkvcdone();
ord_init = 0;
}
if (del_init_oci) {
tkvcdone(0);
del_init_oci = 0;
}
if (del_init_plsql) {
tkvcdone(1);
del_init_plsql = 0;
}
if (sto_init) {
tkvcsdone();
sto_init = 0;
}

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

```

```

#endif
}

/*****
*****
* tkvcninit tkvcndone tkvcpinit tkvcpdone tkvcdinit
tkvcdone tkvcoint tkvcodone *
* tkvcsinit tkvcsdone
*****
*****/

int DBExecution::tkvcninit ()
{
text stmbuf[32*1024];

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

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

#ifdef ISO
sqlfile(".\\blocks\\tkvcnpnew_iso.sql",stmbuf);
#else
#ifdef ISO7
sqlfile(".\\blocks\\tkvcnpnew_iso7.sql",stmbuf);
#else
sqlfile(".\\blocks\\tkvcnpnew.sql",stmbuf);
#endif
#endif

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

/* bind variables */

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

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

```

```

        nol_supply_w_id,SIZ(int),SQLT_INT, nctx-
>nol_supply_w_id_len,
        NITEMS, &nctx->nol_s_count);

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

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

        OCIBNDPLA(nctx->curn1, nctx-
>i_price_bp, errhp, ":i_price", i_price, SIZ(int),
        SQLT_INT, nctx->i_price_len, NITEMS, &nctx-
>nol_item_count);
#endif /* USE_IEEE_NUMBER */
        OCIBNDPLA(nctx->curn1, nctx->i_name_bp, errhp, ":i_name", i_name,
        SIZ(i_name[0]),SQLT_STR, nctx->i_name_len, NITEMS,
        &nctx->nol_name_count);

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

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

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

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

        /* open second cursor */
        DISCARD OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid
**)(&nctx->curn2),
        OCI_HTYPE_STMT, 0, (dvoid**)0));
        DISCARD sprintf((char *) stmbuf, SQLTXNEW2);
        DISCARD OCIERROR(errhp,OCIStmtPrepare(nctx->curn2, errhp,
stmbuf,
        strlen((char *)stmbuf), OCI_NTV_SYNTAX,
OCI_DEFAULT));

        /* execute second cursor to init newinit package */
        {
        int idxlarr[NITEMS];
        OCIBind *idxlarr_bp;
        ub2 idxlarr_len[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;
        o_ol_cnt = NITEMS;

        /* Bind array */
        OCIBNDPLA(nctx->curn2, idxlarr_bp, errhp, ":idxlarr", idxlarr,
        SIZ(int), SQLT_INT, idxlarr_len,
        NITEMS, &idxlarr_count);

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

        if (execstatus != OCI_SUCCESS) {
            OCITransRollback(tpcsvc, errhp, OCI_DEFAULT);

```

```

        errcode = OCIERROR(errhp,execstatus);
        return -1;
    }
}

return (0);
}

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

int DBExecution::tkvcinit (int plsqliflag)
{
    text stmbuf[SQL_BUF_SIZE];

    if (plsqliflag)
    {
        pldctx = (pldelctx *) malloc (sizeof(pldelctx));
        DISCARD memset(pldctx, (char)0, (ub4)sizeof(pldelctx));
        /* Initialize */
        DISCARD OCIHandleAlloc(tpcenv, (dvoid**) &pldctx->curp1,
OCI_HTYPE_STMT, 0,
        (dvoid**)0);
        DISCARD sprintf ((char *) stmbuf, SQLTXTDEL);
        DISCARD OCIStmtPrepare(pldctx->curp1, errhp, stmbuf,
        (ub4) strlen((char *)stmbuf),
        OCI_NTV_SYNTAX, OCI_DEFAULT);
        DISCARD OCIERROR(errhp,
        OCIStmtExecute(tpcsvc, pldctx-
>curp1, errhp, 1, 0, NULLP(OCI_Snapshot),
        NULLP(OCI_Snapshot), OCI_DEFAULT));

        DISCARD OCIHandleAlloc(tpcenv, (dvoid**) &pldctx->curp2,
OCI_HTYPE_STMT,
        0, (dvoid**)0);
#ifdef ISO5 || defined(ISO6) || defined(ISO8)
        #if defined(ISO5)
            sqlfile("\\blocks\\tkvcpd1_iso5.sql", stmbuf);
        #endif
        #if defined(ISO6)
            sqlfile("\\blocks\\tkvcpd1_iso6.sql", stmbuf);
        #endif
        #if defined(ISO8)
            sqlfile("\\blocks\\tkvcpd1_iso8.sql", stmbuf);
        #endif
    #else
        sqlfile("\\blocks\\tkvcpd1.sql", stmbuf);
    #endif
        DISCARD OCIStmtPrepare(pldctx->curp2, errhp, stmbuf,
        (ub4)strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);
        OCIBNDPL(pldctx->curp2, pldctx->w_id_bp, errhp, ":w_id",
        ADR(w_id), SIZ(int), SQLT_INT, &pldctx->w_id_len);
        OCIBNDPL(pldctx->curp2, pldctx->ordcnt_bp,
        errhp, ":ordcnt",
        ADR(pldctx->ordcnt), SIZ(int), SQLT_INT, &pldctx-
>ordcnt_len);
        OCIBNDPL(pldctx->curp2, pldctx->del_date_bp, errhp, ":now",
        ADR(pldctx->del_date), SIZ(OCIdate),
        SQLT_ODT, &pldctx->del_date_len);
        OCIBNDPL(pldctx->curp2, pldctx->carrier_id_bp, errhp,
        ":carrier_id", ADR(o_carrier_id), SIZ(int),
        SQLT_INT, &pldctx->carrier_id_len);

        OCIBNDPLA(pldctx->curp2, pldctx->d_id_bp, errhp, ":d_id",
        pldctx->del_d_id, SIZ(int),SQLT_INT, pldctx-
>del_d_id_len,
        NDISTS, &pldctx->del_d_id_rcnt);
        OCIBNDPLA(pldctx->curp2, pldctx->o_id_bp,
        errhp, ":order_id",
        pldctx->del_o_id, SIZ(int),SQLT_INT, pldctx-
>del_o_id_len, NDISTS,
        &pldctx->del_o_id_rcnt);
#ifdef USE_IEEE_NUMBER
        OCIBNDPLA(pldctx->curp2, pldctx->sums_bp, errhp, ":sums",
        pldctx->sums, SIZ(float),SQLT_BFLOAT, pldctx-
>sums_len, NDISTS,
        &pldctx->sums_rcnt);
#else
        OCIBNDPLA(pldctx->curp2, pldctx->sums_bp, errhp, ":sums",
        pldctx->sums, SIZ(int),SQLT_INT, pldctx-
>sums_len, NDISTS,
        &pldctx->sums_rcnt);
#endif
        OCIBNDPLA(pldctx->curp2, pldctx->o_c_id_bp,
        errhp, ":o_c_id",
        pldctx->o_c_id, SIZ(int),SQLT_INT, pldctx-
>o_c_id_len, NDISTS,
        &pldctx->o_c_id_rcnt);
        OCIBND(pldctx->curp2, pldctx->retry_bp, errhp, ":retry",
        ADR(pldctx->retry), SIZ(int),SQLT_INT);
    }
}

```

```

else
{
    dctx = (delctcx *) malloc (sizeof(delctcx));
    memset(dctx, (char)0, sizeof(delctcx));
    dctx->norow = 0;
    actx = (amtctcx *) malloc (sizeof(amtctcx));
    memset(actx, (char)0, sizeof(amtctcx));

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

    OCIWND(dctx->curd1, dctx->w_id_bp, errhp, ":w_id", dctx-
>w_id, SIZ(int),
        SFLT_INT);
    OCIWNDRA(dctx->curd1, dctx->d_id_bp, errhp, ":d_id", dctx-
>d_id, SIZ(int),
        SFLT_INT, NULL, NULL, NULL);

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

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

    /* bind variables */
    OCIWNDRA(dctx->curd3, dctx-
>carrier_id_bp, errhp, ":carrier_id",
        dctx->carrier_id, SIZ(dctx-
>carrier_id[0]), SFLT_INT,
        dctx->carrier_id_ind, dctx->carrier_id_len, dctx-
>carrier_id_rcode);

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

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

    /* bind variables */
    OCIWND(dctx->curd4, dctx->w_id_bp4, errhp, ":w_id", dctx-
>w_id,
        SIZ(int), SFLT_INT);
    OCIWND(dctx->curd4, dctx->d_id_bp4, errhp, ":d_id", dctx-
>d_id,
        SIZ(int), SFLT_INT);
    OCIWND(dctx->curd4, dctx->o_id_bp, errhp, ":o_id", dctx-
>del_o_id,
        SIZ(int), SFLT_INT);
    OCIWND(dctx->curd4, dctx->cr_date_bp, errhp, ":cr_date",
dctx->del_date,
        SIZ(OCIDate), SFLT_ODT);
    OCIWNDRAD(dctx->curd4, dctx->olamt_bp, errhp, ":ol_amount",
SIZ(int), SFLT_INT, NULL,
    actx, no_data, amt_data);

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

    /* bind variables */

    OCIWND(dctx->curd6, dctx->amt_bp, errhp, ":amt", dctx-
>amt, SIZ(int),
        SFLT_INT);
    OCIWND(dctx->curd6, dctx->w_id_bp6, errhp, ":w_id", dctx-
>w_id, SIZ(int),

```

```

        SFLT_INT);
    OCIWND(dctx->curd6, dctx->d_id_bp6, errhp, ":d_id", dctx-
>d_id, SIZ(int),
        SFLT_INT);
    OCIWND(dctx->curd6, dctx->c_id_bp, errhp, ":c_id", dctx-
>c_id, SIZ(int),
        SFLT_INT);
    }
    return (0);
}

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

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

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

    octx = (ordctcx *) malloc (sizeof(ordctcx));
    DISCARD memset(octx, (char)0, sizeof(ordctcx));
    octx->cs = 1;
    octx->norow = 0;
    octx->somerows = 10;
    for(i=0; i<100; i++) {
        DISCARD OCIERROR(errhp, OCIDescriptorAlloc(tpcenv,
            (dvoid **)&octx->c_rowid_ptr[i],
OCI_DTYPE_ROWID, 0, (dvoid**)0));
    }

    DISCARD OCIERROR(errhp,
OCIHandleAlloc(tpcenv, (dvoid **)&octx-
>curo0, OCI_HTYPE_STMT, 0, (dvoid**)0));
    DISCARD OCIERROR(errhp,
OCIHandleAlloc(tpcenv, (dvoid **)&octx-
>curo0, OCI_HTYPE_STMT, 0, (dvoid**)0));
    DISCARD OCIERROR(errhp,
OCIHandleAlloc(tpcenv, (dvoid **)&octx-
>curo1, OCI_HTYPE_STMT, 0, (dvoid**)0));
    DISCARD OCIERROR(errhp,
OCIHandleAlloc(tpcenv, (dvoid **)&octx-
>curo2, OCI_HTYPE_STMT, 0, (dvoid**)0));
    DISCARD OCIERROR(errhp,
OCIHandleAlloc(tpcenv, (dvoid **)&octx-
>curo3, OCI_HTYPE_STMT, 0, (dvoid**)0));
    DISCARD OCIERROR(errhp,
OCIHandleAlloc(tpcenv, (dvoid **)&octx-
>curo4, OCI_HTYPE_STMT, 0, (dvoid**)0));

    /* c_id = 0, use find customer by lastname. Get an array of
rowid's back*/
    DISCARD sprintf((char *) stmbuf, SQLCUR0);
    DISCARD OCIERROR(errhp,
OCIStmtPrepare(octx->curo0, errhp, stmbuf, (ub4)strlen((char
*)stmbuf),
        OCI_NTV_SYNTAX, OCI_DEFAULT));
    DISCARD OCIERROR(errhp,
OCIAttrSet(octx->curo0, OCI_HTYPE_STMT, &octx->norow, 0,
OCI_ATTR_PREFETCH_ROWS, errhp));
    /* get order/customer info back based on rowid */
    DISCARD sprintf((char *) stmbuf, SQLCUR1);
    DISCARD OCIERROR(errhp,
OCIStmtPrepare(octx->curo1, errhp, stmbuf, (ub4)strlen((char
*)stmbuf),

```

```

OCI_NTV_SYNTAX,OCI_DEFAULT));
DISCARD OCIERROR(errhp,
OCIAttrSet(octx->curo1,OCI_HTYPE_STMT,&octx->norow,0,
OCI_ATTR_PREFETCH_ROWS,errhp));
/* c_id == 0, use lastname to find customer */
DISCARD sprintf((char *) stmbuf, SQLCUR2);
DISCARD OCIERROR(errhp,
OCIStmtPrepare(octx->curo2,errhp,stmbuf,(ub4)strlen((char
*)stmbuf),
OCI_NTV_SYNTAX,OCI_DEFAULT));
DISCARD OCIERROR(errhp,
OCIAttrSet(octx->curo2,OCI_HTYPE_STMT,&octx->norow,0,
OCI_ATTR_PREFETCH_ROWS,errhp));
DISCARD sprintf((char *) stmbuf, SQLCUR3);
DISCARD OCIERROR(errhp,
OCIStmtPrepare(octx->curo3,errhp,stmbuf,(ub4)strlen((char
*)stmbuf),
OCI_NTV_SYNTAX,OCI_DEFAULT));
DISCARD OCIERROR(errhp,
OCIAttrSet(octx->curo3,OCI_HTYPE_STMT,&octx->norow,0,
OCI_ATTR_PREFETCH_ROWS,errhp));
DISCARD sprintf((char *) stmbuf, SQLCUR4);
DISCARD OCIERROR(errhp,
OCIStmtPrepare(octx->curo4,errhp,stmbuf,(ub4)strlen((char
*)stmbuf),
OCI_NTV_SYNTAX,OCI_DEFAULT));
DISCARD OCIERROR(errhp,
OCIAttrSet(octx->curo4,OCI_HTYPE_STMT,&octx->norow,0,
OCI_ATTR_PREFETCH_ROWS,errhp));
for (i = 0; i < NITEMS; i++) {
octx->ol_supply_w_id_len[i] = sizeof(int);
octx->ol_i_id_len[i] = sizeof(int);
octx->ol_quantity_len[i] = sizeof(int);
octx->ol_amount_len[i] = sizeof(int);
octx->ol_delivery_d_len[i] = sizeof(ol_d_base[0]);
}
octx->ol_supply_w_id_csize = NITEMS;
octx->ol_i_id_csize = NITEMS;
octx->ol_quantity_csize = NITEMS;
octx->ol_amount_csize = NITEMS;
octx->ol_delivery_d_csize = NITEMS;
octx->ol_w_id_csize = NITEMS;
octx->ol_o_id_csize = NITEMS;
octx->ol_d_id_csize = NITEMS;
octx->ol_w_id_len = sizeof(int);
octx->ol_d_id_len = sizeof(int);
octx->ol_o_id_len = sizeof(int);
/* bind variables */
/* c_id (customer id) is not known */
OCIBND(octx->curo0,octx->w_id_bp[0],errhp,":w_id",ADR(w_id),
SIZ(int),SQLT_INT);
OCIBND(octx->curo0,octx->d_id_bp[0],errhp,":d_id",ADR(d_id),
SIZ(int),SQLT_INT);
OCIBND(octx->curo0,octx->c_last_bp[0],errhp,":c_last",c_last,
SIZ(c_last), SQLT_STR);
OCIDFNRA(octx->curo0,octx->c_rowid_dp,errhp,1,octx->
c_rowid_ptr,
SIZ(OCIRowid*), SQLT_RDD, NULL, octx->c_rowid_len,
NULL);
OCIBND(octx->curo1,octx->c_rowid_bp,errhp,":cust_rowid",
&octx->c_rowid_cust,
sizeof(octx->c_rowid_ptr[0]),SQLT_RDD);
OCIDF(octx->curo1,octx->
c_id_dp,errhp,1,ADR(c_id),SIZ(int),SQLT_INT);
#ifdef USE_IEEE_NUMBER
OCIDF(octx->curo1,octx->
c_balance_dp[0],errhp,2,ADR(c_balance),
SIZ(double),SQLT_BDOUBLE);
#else
OCIDF(octx->curo1,octx->
c_balance_dp[0],errhp,2,ADR(c_balance),
SIZ(double),SQLT_FLT);
#endif /* USE_IEEE_NUMBER */
OCIDF(octx->curo1,octx->
c_first_dp[0],errhp,3,c_first,SIZ(c_first)-1,
SQLT_CHR);
OCIDF(octx->curo1,octx->c_middle_dp[0],errhp,4,c_middle,
SIZ(c_middle)-1,SQLT_AFC);
OCIDF(octx->curo1,octx->
c_last_dp[0],errhp,5,c_last,SIZ(c_last)-1,
SQLT_CHR);
OCIDF(octx->curo1,octx->
o_id_dp[0],errhp,6,ADR(o_id),SIZ(int),SQLT_INT);
OCIDF(octx->curo1,octx->o_entry_d_dp[0],errhp,7,
&o_entry_d_base,SIZ(OCIDate),SQLT_ODT);
OCIDF(octx->curo1,octx->
o_cr_id_dp[0],errhp,8,ADR(o_carrier_id),
SIZ(int),SQLT_INT);
OCIDF(octx->curo1,octx->
o_ol_cnt_dp[0],errhp,9,ADR(o_ol_cnt),
SIZ(int),SQLT_INT);
/* Bind for third cursor , no-zero customer id */
OCIBND(octx->curo2,octx->w_id_bp[1],errhp,":w_id",ADR(w_id),
SIZ(int),SQLT_INT);
OCIBND(octx->curo2,octx->d_id_bp[1],errhp,":d_id",ADR(d_id),
SIZ(int),SQLT_INT);
OCIBND(octx->curo2,octx->c_id_bp,errhp,":c_id",ADR(c_id),
SIZ(int),SQLT_INT);
#ifdef USE_IEEE_NUMBER

```

```

OCIDF(octx->curo2,octx->
c_balance_dp[1],errhp,1,ADR(c_balance),
SIZ(double),SQLT_BDOUBLE);
#else
OCIDF(octx->curo2,octx->
c_balance_dp[1],errhp,1,ADR(c_balance),
SIZ(double),SQLT_FLT);
#endif /* USE_IEEE_NUMBER */
OCIDF(octx->curo2,octx->
c_first_dp[1],errhp,2,c_first,SIZ(c_first)-1,
SQLT_CHR);
OCIDF(octx->curo2,octx->c_middle_dp[1],errhp,3,c_middle,
SIZ(c_middle)-1,SQLT_AFC);
OCIDF(octx->curo2,octx->
c_last_dp[1],errhp,4,c_last,SIZ(c_last)-1,
SQLT_CHR);
OCIDF(octx->curo2,octx->
o_id_dp[1],errhp,5,ADR(o_id),SIZ(int),SQLT_INT);
OCIDF(octx->curo2,octx->o_entry_d_dp[1],errhp,6,
&o_entry_d_base,
SIZ(OCIDate),SQLT_ODT);
OCIDF(octx->curo2,octx->
o_cr_id_dp[1],errhp,7,ADR(o_carrier_id),
SIZ(int),SQLT_INT);
OCIDF(octx->curo2,octx->
o_ol_cnt_dp[1],errhp,8,ADR(o_ol_cnt),
SIZ(int),SQLT_INT);
/* Bind for last cursor */
OCIBND(octx->curo3,octx->w_id_bp[2],errhp,":w_id",ADR(w_id),
SIZ(int),SQLT_INT);
OCIBND(octx->curo3,octx->d_id_bp[2],errhp,":d_id",ADR(d_id),
SIZ(int),SQLT_INT);
OCIBND(octx->curo3,octx->o_id_bp,errhp,":o_id",ADR(o_id),
SIZ(int),SQLT_INT);
/*
OCIBND(octx->curo3,octx->c_id_bp,errhp,":c_id",ADR(c_id),
SIZ(int),SQLT_INT);
*/
OCIDFNRA(octx->curo3,octx->ol_i_id_dp,errhp,1,
ol_i_id,SIZ(int),SQLT_INT,
NULL,octx->ol_i_id_len,NULL);
OCIDFNRA(octx->curo3,octx->ol_supply_w_id_dp,errhp,2,
ol_supply_w_id,
SIZ(int),SQLT_INT, NULL,
octx->ol_supply_w_id_len, NULL);
#ifdef USE_IEEE_NUMBER
OCIDFNRA(octx->curo3,octx->ol_quantity_dp,errhp,3,
ol_quantity,SIZ(float),
SQLT_BFLOAT, NULL,octx->ol_quantity_len, NULL);
OCIDFNRA(octx->curo3,octx->ol_amount_dp,errhp,4,ol_amount,
SIZ(float),
SQLT_BFLOAT,NULL, octx->ol_amount_len, NULL);
#else
OCIDFNRA(octx->curo3,octx->ol_quantity_dp,errhp,3,
ol_quantity,SIZ(int),
SQLT_INT, NULL,octx->ol_quantity_len, NULL);
OCIDFNRA(octx->curo3,octx->ol_amount_dp,errhp,4,ol_amount,
SIZ(int),
SQLT_INT,NULL, octx->ol_amount_len, NULL);
#endif /* USE_IEEE_NUMBER */
OCIDFNRA(octx->curo3,octx->
ol_d_base_dp,errhp,5,ol_d_base,SIZ(OCIDate),
SQLT_ODT, NULL,octx->ol_delivery_d_len,NULL);
OCIBND(octx->curo4,octx->w_id_bp[3],errhp,":w_id",ADR(w_id),
SIZ(int),SQLT_INT);
OCIBND(octx->curo4,octx->d_id_bp[3],errhp,":d_id",ADR(d_id),
SIZ(int),SQLT_INT);
OCIBND(octx->curo4,octx->c_last_bp[1],errhp,":c_last",c_last,
SIZ(c_last), SQLT_STR);
OCIDF(octx->curo4,octx->c_cnt_dp,errhp,1,ADR(octx->
rcount),SIZ(int),
SQLT_INT);
return (0);
}
void DBExecution::tkvcodone ()
{
if (octx)
free (octx);
}
int DBExecution::tkvcpinit (void)
{
text stmbuf[SQL_BUF_SIZE];
pctx = (payctx *)malloc(sizeof(payctx));
memset(pctx,(char)0,sizeof(payctx));
/* cursor for init */
DISCARD OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid
**>(&pctx->curp1)),
OCI_HTYPE_STMT,0,(dvoid**)0));
DISCARD OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid
**>(&pctx->curp0)),
OCI_HTYPE_STMT,0,(dvoid**)0));
DISCARD OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid
**>(&pctx->curp1)),

```

```

OCI_HTYPE_STMT,0,(dvoid**)0);

/* build the init statement and execute it */
sprintf ((char*)stmbuf, SQLTXT_INIT);
DISCARD OCIERROR(errhp,OCIStmtPrepare(pctx->curpi, errhp,
stmbuf,
        strlen((char *)stmbuf), OCI_NTV_SYNTAX,
OCI_DEFAULT));
DISCARD OCIERROR(errhp, OCIStmtExecute(tpcsvc,pctx->
curpi,errhp,1,0,
        NULLP(CONST
OCI_Snapshot),NULLP(OCI_Snapshot),OCI_DEFAULT));

/* customer id != 0, go by last name */
sqlfile(".\\blocks\\paynz.sql",stmbuf);
DISCARD OCIERROR(errhp,OCIStmtPrepare(pctx->curp0, errhp,
stmbuf,
        strlen((char *)stmbuf), OCI_NTV_SYNTAX,
OCI_DEFAULT));

/* customer id == 0, go by last name */
sqlfile(".\\blocks\\payz.sql",stmbuf); /* sqlfile opens
$/bench/.../blocks/... */
DISCARD OCIERROR(errhp,OCIStmtPrepare(pctx->curp1, errhp,
stmbuf,
        strlen((char *)stmbuf), OCI_NTV_SYNTAX,
OCI_DEFAULT));

pctx->w_id_len = SIZ(w_id);
pctx->d_id_len = SIZ(d_id);
pctx->c_w_id_len = SIZ(c_w_id);
pctx->c_d_id_len = SIZ(c_d_id);
pctx->c_id_len = 0;
pctx->h_amount_len = SIZ(h_amount);
pctx->c_last_len = 0;
pctx->w_street_1_len = 0;
pctx->w_street_2_len = 0;
pctx->w_city_len = 0;
pctx->w_state_len = 0;
pctx->w_zip_len = 0;
pctx->d_street_1_len = 0;
pctx->d_street_2_len = 0;
pctx->d_city_len = 0;
pctx->d_state_len = 0;
pctx->d_zip_len = 0;
pctx->c_first_len = 0;
pctx->c_middle_len = 0;
pctx->c_street_1_len = 0;
pctx->c_street_2_len = 0;
pctx->c_city_len = 0;
pctx->c_state_len = 0;
pctx->c_zip_len = 0;
pctx->c_phone_len = 0;
pctx->c_since_len = 0;
pctx->c_credit_len = 0;
pctx->c_credit_lim_len = 0;
pctx->c_discount_len = 0;
pctx->c_balance_len = sizeof(double);
pctx->c_data_len = 0;
pctx->h_date_len = 0;
pctx->retries_len =SIZ(retries) ;
pctx->cr_date_len = 7;

/* bind variables */

OCIBNDPL(pctx->curp0, pctx->w_id_bp[0],
errhp,":w_id",ADR(w_id),SIZ(int),
        SQLT_INT, NULL);
OCIBNDPL(pctx->curp0, pctx->d_id_bp[0],
errhp,":d_id",ADR(d_id),SIZ(int),
        SQLT_INT, NULL);
OCIBND(pctx->curp0, pctx->c_w_id_bp[0],
errhp,":c_w_id",ADR(c_w_id),SIZ(int),
        SQLT_INT);
OCIBND(pctx->curp0, pctx->c_d_id_bp[0],
errhp,":c_d_id",ADR(c_d_id),SIZ(int),
        SQLT_INT);
OCIBND(pctx->curp0, pctx->c_id_bp[0],
errhp,":c_id",ADR(c_id),SIZ(int),
        SQLT_INT);
#ifdef USE_IEEE_NUMBER
OCIBNDPL(pctx->curp0, pctx->h_amount_bp[0],
errhp,":h_amount",ADR(h_amount),
        SIZ(float),SQLT_BFLOAT, &pctx->h_amount_len);
#else
OCIBNDPL(pctx->curp0, pctx->h_amount_bp[0],
errhp,":h_amount",ADR(h_amount),
        SIZ(int),SQLT_INT, &pctx->h_amount_len);
#endif /* USE_IEEE_NUMBER */
OCIBNDPL(pctx->curp0, pctx->c_last_bp[0],
errhp,":c_last",c_last,SIZ(c_last),
        SQLT_STR, &pctx->c_last_len);
OCIBNDPL(pctx->curp0, pctx->w_street_1_bp[0],
errhp,":w_street_1",w_street_1,
        SIZ(w_street_1),SQLT_STR, &pctx->w_street_1_len);
OCIBNDPL(pctx->curp0, pctx->w_street_2_bp[0],
errhp,":w_street_2",w_street_2,
        SIZ(w_street_2),SQLT_STR, &pctx->w_street_2_len);
OCIBNDPL(pctx->curp0, pctx->w_city_bp[0],
errhp,":w_city",w_city,SIZ(w_city),
        SQLT_STR, &pctx->w_city_len);

```

```

OCIBNDPL(pctx->curp0, pctx->w_state_bp[0],
errhp,":w_state",w_state,
        SIZ(w_state), SQLT_STR, &pctx->w_state_len);
OCIBNDPL(pctx->curp0, pctx->w_zip_bp[0],
errhp,":w_zip",w_zip,SIZ(w_zip),
        SQLT_STR, &pctx->w_zip_len);
OCIBNDPL(pctx->curp0, pctx->d_street_1_bp[0],
errhp,":d_street_1",d_street_1,
        SIZ(d_street_1),SQLT_STR, &pctx->d_street_1_len);
OCIBNDPL(pctx->curp0, pctx->d_street_2_bp[0],
errhp,":d_street_2",d_street_2,
        SIZ(d_street_2),SQLT_STR, &pctx->d_street_2_len);
OCIBNDPL(pctx->curp0, pctx->d_city_bp[0],
errhp,":d_city",d_city,SIZ(d_city),
        SQLT_STR, &pctx->d_city_len);
OCIBNDPL(pctx->curp0, pctx->d_state_bp[0],
errhp,":d_state",d_state,
        SIZ(d_state), SQLT_STR, &pctx->d_state_len);
OCIBNDPL(pctx->curp0, pctx->d_zip_bp[0],
errhp,":d_zip",d_zip,SIZ(d_zip),
        SQLT_STR, &pctx->d_zip_len);
OCIBNDPL(pctx->curp0, pctx->c_first_bp[0],
errhp,":c_first",c_first,
        SIZ(c_first), SQLT_STR, &pctx->c_first_len);
OCIBNDPL(pctx->curp0, pctx->c_middle_bp[0],
errhp,":c_middle",c_middle,2,
        SQLT_AFC, &pctx->c_middle_len);
OCIBNDPL(pctx->curp0, pctx->c_street_1_bp[0],
errhp,":c_street_1",c_street_1,
        SIZ(c_street_1),SQLT_STR, &pctx->c_street_1_len);
OCIBNDPL(pctx->curp0, pctx->c_street_2_bp[0],
errhp,":c_street_2",c_street_2,
        SIZ(c_street_2),SQLT_STR, &pctx->c_street_2_len);
OCIBNDPL(pctx->curp0, pctx->c_city_bp[0],
errhp,":c_city",c_city,SIZ(c_city),
        SQLT_STR, &pctx->c_city_len);
OCIBNDPL(pctx->curp0, pctx->c_state_bp[0],
errhp,":c_state",c_state,
        SIZ(c_state), SQLT_STR, &pctx->c_state_len);
OCIBNDPL(pctx->curp0, pctx->c_zip_bp[0],
errhp,":c_zip",c_zip,SIZ(c_zip),
        SQLT_STR,&pctx->c_zip_len);
OCIBNDPL(pctx->curp0, pctx->c_phone_bp[0],
errhp,":c_phone",c_phone,
        SIZ(c_phone), SQLT_STR, &pctx->c_phone_len);
OCIBNDPL(pctx->curp0, pctx->c_since_bp[0],
errhp,":c_since",&c_since,
        SIZ(OCIDate), SQLT_ODT, &pctx->c_since_len);
OCIBNDPL(pctx->curp0, pctx->c_credit_bp[0],
errhp,":c_credit",c_credit,
        SIZ(c_credit),SQLT_CHR, &pctx->c_credit_len);
OCIBNDPL(pctx->curp0, pctx->c_credit_lim_bp[0],
errhp,":c_credit_lim",
        ADR(c_credit_lim),SIZ(int), SQLT_INT, &pctx->
c_credit_lim_len);
OCIBNDPL(pctx->curp0, pctx->c_discount_bp[0],
errhp,":c_discount",
        ADR(c_discount),SIZ(c_discount), SQLT_FLT, &pctx->
c_discount_len);
#ifdef USE_IEEE_NUMBER
OCIBNDPL(pctx->curp0, pctx->c_balance_bp[0],
errhp,":c_balance",
        ADR(c_balance), SIZ(double),SQLT_BDOUBLE, &pctx->
c_balance_len);
#else
OCIBNDPL(pctx->curp0, pctx->c_balance_bp[0],
errhp,":c_balance",
        ADR(c_balance), SIZ(double),SQLT_FLT, &pctx->
c_balance_len);
#endif /* USE_IEEE_NUMBER */
OCIBNDPL(pctx->curp0, pctx->c_data_bp[0],
errhp,":c_data",c_data,SIZ(c_data),
        SQLT_STR, &pctx->c_data_len);
/*
OCIBNDPL(pctx->curp0, pctx->retries_bp[0],
errhp,":retry",ADR(retries),
        SIZ(int), SQLT_INT, &pctx->retries_len);
OCIBNDPL(pctx->curp0, pctx->cr_date_bp[0],
errhp,":cr_date",ADR(cr_date),
        SIZ(OCIDate),SQLT_ODT, &pctx->cr_date_len);
*/
/* ---- Binds for the second cursor */

OCIBNDPL(pctx->curp1, pctx->w_id_bp[1],
errhp,":w_id",ADR(w_id),SIZ(int),
        SQLT_INT, &pctx->w_id_len);
OCIBNDPL(pctx->curp1, pctx->d_id_bp[1],
errhp,":d_id",ADR(d_id),SIZ(int),
        SQLT_INT, &pctx->d_id_len);
OCIBND(pctx->curp1, pctx->c_w_id_bp[1],
errhp,":c_w_id",ADR(c_w_id),SIZ(int),
        SQLT_INT);
OCIBND(pctx->curp1, pctx->c_d_id_bp[1],
errhp,":c_d_id",ADR(c_d_id),SIZ(int),
        SQLT_INT);
OCIBNDPL(pctx->curp1, pctx->c_id_bp[1],
errhp,":c_id",ADR(c_id),SIZ(int),
        SQLT_INT, &pctx->c_id_len);
#ifdef USE_IEEE_NUMBER
OCIBNDPL(pctx->curp1, pctx->h_amount_bp[1],
errhp,":h_amount",ADR(h_amount),
        SIZ(float),SQLT_BFLOAT, &pctx->h_amount_len);

```

```

#else
OCIBNDPL(pctx->curpl, pctx->h_amount_bp[1],
errhp,":h_amount",ADR(h_amount),
        SIZ(int),SQLT_INT, &pctx->h_amount_len);
#endif /* USE_IEEE_NUMBER */
OCIBND(pctx->curpl, pctx->c_last_bp[1],
errhp,":c_last",c_last,SIZ(c_last),
        SQLT_STR);
OCIBNDPL(pctx->curpl, pctx->w_street_1_bp[1],
errhp,":w_street_1",w_street_1,
        SIZ(w_street_1),SQLT_STR, &pctx->w_street_1_len);
OCIBNDPL(pctx->curpl, pctx->w_street_2_bp[1],
errhp,":w_street_2",w_street_2,
        SIZ(w_street_2),SQLT_STR, &pctx->w_street_2_len);
OCIBNDPL(pctx->curpl, pctx->w_city_bp[1],
errhp,":w_city",w_city,SIZ(w_city),
        SQLT_STR, &pctx->w_city_len);
OCIBNDPL(pctx->curpl, pctx->w_state_bp[1],
errhp,":w_state",w_state,
        SIZ(w_state), SQLT_STR, &pctx->w_state_len);
OCIBNDPL(pctx->curpl, pctx->w_zip_bp[1],
errhp,":w_zip",w_zip,SIZ(w_zip),
        SQLT_STR, &pctx->w_zip_len);
OCIBNDPL(pctx->curpl, pctx->d_street_1_bp[1],
errhp,":d_street_1",d_street_1,
        SIZ(d_street_1),SQLT_STR, &pctx->d_street_1_len);
OCIBNDPL(pctx->curpl, pctx->d_street_2_bp[1],
errhp,":d_street_2",d_street_2,
        SIZ(d_street_2),SQLT_STR, &pctx->d_street_2_len);
OCIBNDPL(pctx->curpl, pctx->d_city_bp[1],
errhp,":d_city",d_city,SIZ(d_city),
        SQLT_STR, &pctx->d_city_len);
OCIBNDPL(pctx->curpl, pctx->d_state_bp[1],
errhp,":d_state",d_state,
        SIZ(d_state), SQLT_STR, &pctx->d_state_len);
OCIBNDPL(pctx->curpl, pctx->d_zip_bp[1],
errhp,":d_zip",d_zip,SIZ(d_zip),
        SQLT_STR, &pctx->d_zip_len);
OCIBNDPL(pctx->curpl, pctx->c_first_bp[1],
errhp,":c_first",c_first,
        SIZ(c_first), SQLT_STR, &pctx->c_first_len);
OCIBNDPL(pctx->curpl, pctx->c_middle_bp[1],
errhp,":c_middle",c_middle,2,
        SQLT_AFC, &pctx->c_middle_len);

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

return (0);
}

```

```

void DBExecution::tkvcpdone ()
{
if(pctx) {
free(pctx);
}
}

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

sctx->norow=0;

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

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

return (0);
}

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

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

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

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

/* get number of order lines, and check if all are local */
o_ol_cnt = NITEMS;
o_all_local = 1;
for (i = 0; i < NITEMS; i++) {
if (nol_i_id[i] == 0) {
o_ol_cnt = i;
break;
}
if (nol_supply_w_id[i] != w_id) {
#ifdef USE_IEEE_NUMBER
nctx->s_remote[i] = 1.0;
#else
nctx->s_remote[i] = 1;
#endif /* USE_IEEE_NUMBER */
o_all_local = 0;
}
}
else
}

```

```

    nctx->s_remote[i] = 0;
}

nctx->w_id_len = sizeof(w_id);
nctx->d_id_len = sizeof(d_id);
nctx->c_id_len = sizeof(c_id);
nctx->o_all_local_len = sizeof(o_all_local);
nctx->o_ol_cnt_len = sizeof(o_ol_cnt);
nctx->w_tax_len = 0;
nctx->d_tax_len = 0;
nctx->o_id_len = sizeof(o_id);
nctx->c_discount_len = 0;
nctx->c_credit_len = 0;
nctx->c_last_len = 0;
nctx->retries_len = sizeof(retries);
nctx->cr_date_len = sizeof(cr_date);
/* this is the row count */
rcount = o_ol_cnt;
nctx->nol_i_count = o_ol_cnt;
nctx->nol_g_count = o_ol_cnt;
nctx->nol_s_count = o_ol_cnt;
nctx->s_remote_count = o_ol_cnt;

nctx->nol_qty_count = 0;
nctx->nol_bg_count = 0;
nctx->nol_item_count = 0;
nctx->nol_name_count = 0;
nctx->nol_am_count = 0;

/* initialization for array operations */
for (i = 0; i < o_ol_cnt; i++) {
    nctx->ol_number[i] = i + 1;
    nctx->nol_i_id_len[i] = sizeof(int);
    nctx->nol_supply_w_id_len[i] = sizeof(int);
    nctx->nol_quantity_len[i] = sizeof(int);
    nctx->nol_amount_len[i] = sizeof(int);
    nctx->ol_o_id_len[i] = sizeof(int);
    nctx->ol_number_len[i] = sizeof(int);
    nctx->ol_dist_info_len[i] = nctx->s_dist_info_len[i];
    nctx->s_remote_len[i] = sizeof(int);
    nctx->s_quant_len[i] = sizeof(int);
    nctx->i_name_len[i]=0;
    nctx->s_bg_len[i] = 0;
}
for (i = o_ol_cnt; i < NITEMS; i++) {

    nctx->nol_i_id_len[i] = 0;
    nctx->nol_supply_w_id_len[i] = 0;
    nctx->nol_quantity_len[i] = 0;
    nctx->nol_amount_len[i] = 0;
    nctx->ol_o_id_len[i] = 0;
    nctx->ol_number_len[i] = 0;
    nctx->ol_dist_info_len[i] = 0;
    nctx->s_remote_len[i] = 0;
    nctx->s_quant_len[i] = 0;
    nctx->i_name_len[i]=0;
    nctx->s_bg_len[i] = 0;
}

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

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

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

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

return (0);
}

int DBExecution::tkvcd (int plsqliflag)
{
    int i;
    int rpc,rcount;
    int invalid;

    if (plsqliflag)
    {

```

```

pldctx->w_id_len = sizeof (int);
pldctx->carrier_id_len = sizeof (int);
for (i = 0; i < NDISTS; i++)
{
    pldctx->del_o_id_len[i] = sizeof(int);
    del_o_id[i] = 0;
}
pldctx->del_date_len = DEL_DATE_LEN;
DISCARD memcpy(&pldctx->del_date,&cr_date,sizeof(OCIDate));

pldctx->retry=0;

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

retry:
    invalid = 0;

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

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

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

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

/* array select from new_order and orders tables */
execstatus=OCISstmtExecute(tpcsvc,dctx-
>curdl,errhp,NDISTS,0,
    NULLP(CONST
OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
if((execstatus != OCI_SUCCESS) && (execstatus !=
OCI_NO_DATA))
{
    DISCARD OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    errcode = OCIERROR(errhp,execstatus);
    if(errcode == NOT_SERIALIZABLE)
    {
        retries++;
        goto retry;
    }
    else if (errcode == RECOVERERR)
    {
        retries++;
        goto retry;
    }
    else if (errcode == SNAPSHOT_TOO_OLD)
    {
        retries++;
        goto retry;
    }
    else
    {
        return -1;
    }
}

/* mark districts with no new order */
DISCARD OCIAttrGet(dctx-
>curdl,OCI_HTYPE_STMT,&rcount,NULLP(ub4),
    OCI_ATTR_ROW_COUNT,errhp);
rpc = rcount;
if (rcount != NDISTS )
{
    int j = 0;
    for (i=0;i < NDISTS; i++)
    {
        if (dctx->del_o_id_ind[j] == 0) /* there is data here
*/
            j++;
        else
            shiftdata(j);
    }
}
}
}

```

```

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

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

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

/* array update of order_line table */
execstatus=OCIStmtExecute(tpcsvc,dctx->curd4,errhp,rcp,0,
    NULLP(CONST
OCI_Snapshot),NULLP(OCI_Snapshot),OCI_DEFAULT);
if(execstatus != OCI_SUCCESS)
{
    DISCARD OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    errcode = OCIERROR(errhp,execstatus);
    if(errcode == NOT_SERIALIZABLE)
    {
        retries++;
        goto retry;
    }
    else if (errcode == RECOVERERR)
    {
        retries++;
        goto retry;
    }
    else if (errcode == SNAPSHOT_TOO_OLD)
    {
        retries++;
        goto retry;
    }
    else
    {
        return -1;
    }
}

DISCARD OCIAttrGet(dctx-
>curd4,OCI_HTYPE_STMT,&rcount,NULLP(ub4),
    OCI_ATTR_ROW_COUNT,errhp);
/* transfer amounts */
for (i=0;i<rpc;i++)
{
    dctx->amt[i]=0;
    if ( actx->ol_amt_rcode[i] == 0)
    {
        dctx->amt[i] = actx->ol_amt[i];
    }
}
#endif OLD
if (rcount > rpc) {
    userlog
        ("Error in TPC-C server %d: %d ordnrs updated, %d
ordl updated\n",
        proc_no, rpc, rcount);
}
#endif

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

if(execstatus != OCI_SUCCESS)
{
    OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
    errcode = OCIERROR(errhp,execstatus);
    if(errcode == NOT_SERIALIZABLE)
    {
        retries++;
        goto retry;
    }
    else if (errcode == RECOVERERR)
    {
        retries++;
        goto retry;
    }
}

```

```

    }
    else if (errcode == SNAPSHOT_TOO_OLD)
    {
        retries++;
        goto retry;
    }
    else
    {
        return -1;
    }
}

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

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

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

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

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

#ifdef IS09
    int secondread = 0;
    char sdate[30];
    ub4 datelen;
    sysdate(sdate);
    printf("Order Status started at: %s\n", sdate);
#endif

    for (i = 0; i < NITEMS; i++) {
        octx->ol_supply_w_id_len[i] = sizeof(int);
        octx->ol_i_id_len[i] = sizeof(int);
        octx->ol_quantity_len[i] = sizeof(int);
        octx->ol_amount_len[i] = sizeof(int);
        octx->ol_delivery_d_len[i] = sizeof(OCIDate);
    }
    octx->ol_supply_w_id_csize = NITEMS;
    octx->ol_i_id_csize = NITEMS;
    octx->ol_quantity_csize = NITEMS;
    octx->ol_amount_csize = NITEMS;
    octx->ol_delivery_d_csize = NITEMS;

    retry:
    if (bylastname)
    {
        cbctx.reexec = FALSE;
        execstatus=OCIStmtExecute(tpcsvc,octx->curo0,errhp,100,0,
            NULLP(CONST
OCI_Snapshot),NULLP(OCI_Snapshot),OCI_DEFAULT);
        /* will get OCI_NO_DATA if <100 found */
        if ((execstatus != OCI_NO_DATA) && (execstatus !=
OCI_SUCCESS))
        {
            errcode=OCIERROR(errhp, execstatus);
            if((errcode == NOT_SERIALIZABLE) || (errcode ==
RECOVERERR))
            {
                DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
                retries++;
                goto retry;
            }
            else {
                return -1;
            }
        }
        if (execstatus == OCI_NO_DATA) /* there are no more rows */
        {
            /* get rowcount, find middle one */
            DISCARD OCIAttrGet(octx-
>curo0,OCI_HTYPE_STMT,&rcount,NULL,
                OCI_ATTR_ROW_COUNT,errhp);

            if (rcount <1)
            {
                /*
                userlog("ORDERSTATUS rcount=%d\n",rcount);
                */
                return (-1);
            }
            octx->cust_idx=(rcount)/2 ;
        }
        else
        {
            /* count the number of rows */
            execstatus=OCIStmtExecute(tpcsvc,octx->curo4,errhp,1,0,
                NULLP(CONST
OCI_Snapshot),NULLP(OCI_Snapshot),OCI_DEFAULT);
            if ((execstatus != OCI_NO_DATA) && (execstatus !=
OCI_SUCCESS))
            {

```



```

        errcode=OCIERROR(errhp, execstatus);
    if ((errcode == NOT_SERIALIZABLE) || (errcode == RECOVER))
    {
        DISCARD OCITransCommit(tpcsvc, errhp, OCI_DEFAULT);
        retries++;
        goto retry;
    } else {
        return -1;
    }
}
cbctx.reexec = TRUE;
cbctx.count = (octx->rcount+1)/2 ;
execstatus=OCISmtExecute(tpcsvc, octx-
>curo0, errhp, cbctx.count,
                                0, NULLP(CONST OCISnapshot),
NULLP(OCISnapshot), OCI_DEFAULT);
        DISCARD OCIAttrGet(octx-
>curo0, OCI_HTYPE_STMT, &rcount, NULL,
                                OCI_ATTR_ROW_COUNT, errhp);

/* will get OCI_NO_DATA if <100 found */
if ((int)cbctx.count != rcount)
{
}

/*
userlog ("did not get all rows ");
*/
return (-1);
}

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

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

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

OCIERROR(errhp, OCIDateToText(errhp, &o_entry_d_base, (text*)FULLDA
TE, SIZ(FULLDATE), (text*)0, 0, &datelen, o_entry_d));

```

```

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

execstatus = OCISmtExecute(tpcsvc, octx-
>curo3, errhp, o_ol_cnt, 0,
                                NULLP(CONST
OCIISnapshot), NULLP(OCISnapshot),
                                OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
if (execstatus != OCI_SUCCESS)
{
    errcode=OCIERROR(errhp, execstatus);
    DISCARD OCITransCommit(tpcsvc, errhp, OCI_DEFAULT);
    if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVER))
    || (errcode == SNAPSHOT_TOO_OLD)
    {
        retries++;
        goto retry;
    }
    else
    {
        return -1;
    }
}
/* clean up and convert the delivery dates */
for (i = 0; i < o_ol_cnt; i++)
{
    ol_del_len[i]=sizeof(ol_delivery_d[i]);
    DISCARD OCIERROR(errhp, OCIDateToText(errhp, &ol_d_base[i],
(const
text*)SHORTDATE, (ub1)strlen(SHORTDATE), (text*)0, 0,
&ol_del_len[i], ol_delivery_d[i]));
}
/*
cvtdmy(ol_d_base[i], ol_delivery_d[i]);
*/
}
return (0);
}

int DBExecution::tkvcv ()
{
    retry:

    pctx->w_id_len = SIZ(w_id);
    pctx->d_id_len = SIZ(d_id);
    pctx->c_w_id_len = 0;
    pctx->c_d_id_len = 0;
    pctx->c_id_len = 0;
    pctx->h_amount_len = SIZ(h_amount);
    pctx->c_last_len = SIZ(c_last);
    pctx->w_street_1_len = 0;
    pctx->w_street_2_len = 0;
    pctx->w_city_len = 0;
    pctx->w_state_len = 0;
    pctx->w_zip_len = 0;
    pctx->d_street_1_len = 0;
    pctx->d_street_2_len = 0;
    pctx->d_city_len = 0;
    pctx->d_state_len = 0;
    pctx->d_zip_len = 0;
    pctx->c_first_len = 0;
    pctx->c_middle_len = 0;
    pctx->c_street_1_len = 0;
    pctx->c_street_2_len = 0;
    pctx->c_city_len = 0;
    pctx->c_state_len = 0;
    pctx->c_zip_len = 0;
    pctx->c_phone_len = 0;
    pctx->c_since_len = 0;
    pctx->c_credit_len = 0;
    pctx->c_credit_lim_len = 0;
    pctx->c_discount_len = 0;
    pctx->c_balance_len = sizeof(double);
    pctx->c_data_len = 0;
    pctx->h_date_len = 0;
    pctx->retries_len = SIZ(retries);
    pctx->cr_date_len = 7;

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

```

```

OCI_DEFAULT|OCI_COMMIT_ON_SUCCESS);
}

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

int DBExecution::tkvcs ()
{
retry:
execstatus= OCISmtExecute(tpcsvc, sctx->curs, errhp, 1, 0, 0, 0,
OCI_COMMIT_ON_SUCCESS |
OCI_DEFAULT);

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

return (0);
}

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

int DBExecution::TPCnew (struct newstruct *str)
{
int i;

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

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

for (i = 0; i < NITEMS; i++) indx[i] = i;
q_sort(nol_i_id, str, 0, ordl_cnt-1);
#endif

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

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

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

```

```

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

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

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

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

int DBExecution::TPCpay (struct paystruct *str)
{
w_id = str->payin.w_id;
d_id = str->payin.d_id;
c_w_id = str->payin.c_w_id;
c_d_id = str->payin.c_d_id;
#ifdef USE_IEEE_NUMBER
h_amount = (float) str->payin.h_amount;
#else
h_amount = str->payin.h_amount;
#endif /* USE_IEEE_NUMBER */
bylastname = str->payin.bylastname;

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

if (bylastname) {
c_id = 0;
strncpy (c_last, str->payin.c_last, 17);
}
else {
c_id = str->payin.c_id;
strcpy (c_last, " ");
}
retries = 0;

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

/*
cvtdmyhms(cr_date, h_date);
*/
hlen=SIZ(h_date);
OCIERROR(errhp, OCIDateToText(errhp, &cr_date,
(text*)FULLDATE, strlen(FULLDATE), (text*)0, 0, &hlen, h_date));

/*
cvtdmy(c_since, c_since_d);
*/
sincelen=SIZ(c_since_d);
OCIERROR(errhp, OCIDateToText(errhp, &c_since,
(text*)SHORTDATE, strlen(SHORTDATE), (text*)0, 0, &sincelen, c_since_
d));

str->payout.terror = NOERR;
strncpy (str->payout.w_street_1, w_street_1, 21);
strncpy (str->payout.w_street_2, w_street_2, 21);
strncpy (str->payout.w_city, w_city, 21);
strncpy (str->payout.w_state, w_state, 3);
strncpy (str->payout.w_zip, w_zip, 10);
strncpy (str->payout.d_street_1, d_street_1, 21);
strncpy (str->payout.d_street_2, d_street_2, 21);
strncpy (str->payout.d_city, d_city, 21);
strncpy (str->payout.d_state, d_state, 3);
strncpy (str->payout.d_zip, d_zip, 10);

```

```

str->payout.c_id = c_id;
strncpy (str->payout.c_first, c_first, 17);
strncpy (str->payout.c_middle, c_middle, 3);
strncpy (str->payout.c_last, c_last, 17);
strncpy (str->payout.c_street_1, c_street_1, 21);
strncpy (str->payout.c_street_2, c_street_2, 21);
strncpy (str->payout.c_city, c_city, 21);
strncpy (str->payout.c_state, c_state, 3);
strncpy (str->payout.c_zip, c_zip, 10);
strncpy (str->payout.c_phone, c_phone, 17);
strncpy (str->payout.c_since, (char*)c_since_d, 11);
strncpy (str->payout.c_credit, c_credit, 3);
str->payout.c_credit_lim = (float)(c_credit_lim)/100;
str->payout.c_discount = c_discount;
str->payout.c_balance = (float)(c_balance)/100;
strncpy (str->payout.c_data, c_data, 201);
strncpy (str->payout.h_date, (char*)h_date, 20);
str->payout.retry = retries;
return(1);
}

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

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

    datelen = sizeof(o_entry_d);
    OCIERROR(errhp,
             &datelen, o_entry_d);

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

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

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

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

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

    for (i = 0; i < 10; i++) {

```

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

```

str->newout.ol_amount[j] = temp;

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

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

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

#endif

#ifdef LOOPBACK

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

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

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

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

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

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

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

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

    return SUCCESS;
}

int mod_tpcc_payment(T_payment_data *output)

```

```

{
    int i;
    char c;

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

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

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

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

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

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

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

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

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

    return SUCCESS;
}

```



```

char c_first[17];
char c_middle[3];
double c_balance;
int o_id;
text o_entry_d[20];
ub4 datelen;
int o_carrier_id;
int o_ol_cnt;
int ol_supply_w_id[15];
int ol_i_id[15];
#ifdef USE_IEEE_NUMBER
float ol_quantity[15];
float ol_amount[15];
#else
int ol_quantity[15];
int ol_amount[15];
#endif /* USE_IEEE_NUMBER */
ub4 ol_del_len[15];
text ol_delivery_d[15][11];
OCIRowid *o_rowid;

/* for payment transaction */

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

/* for new order transaction */

int nol_i_id[15];
int nol_supply_w_id[15];
#ifdef USE_IEEE_NUMBER
float nol_quantity[15];
float nol_amount[15];
float s_quantity[15];
float i_price[15];
#else
int nol_quantity[15];
int nol_amount[15];
int s_quantity[15];
int i_price[15];
#endif /* USE_IEEE_NUMBER */
int nol_quant10[15];
int nol_quant19[15];
int nol_ytdqty[15];
int o_all_local;
float w_tax;
float d_tax;
float total_amount;
char i_name[15][25];
char brand_gen[15];
char brand_generic[15][1];
int tracelevel;

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

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

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

```

```

-----
loopback.cpp
-----
#include "stdafx.h"
#include "DBConnection.h"

-----
modtpcc.cpp
-----
// modtpcc.cpp : Defines the entry point for the DLL
application.
//
#include "stdafx.h"
#include "modtpcc.h"
#include <httpext.h>

#ifdef DEBUG
#define DELIVERY_MUTEX
#define NEW_ALLOCATE_FROM
#endif

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

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

        strcpy(origin, DllPath);
        if (DllPath[0]!='\\' && DllPath[1]!='\\' && DllPath[2]!='?'
            && DllPath[3]!='\\')
            strcpy(DllPath, DllPath+4);
        for (int i=strlen(DllPath); DllPath[i]!='\\' && i; i--)
            DllPath[i]='\0';
        sprintf(InitFile, "%s\\%s", DllPath, InitName);
        sprintf(DllFile, "%s\\%s", DllPath, DllName);
        sprintf(LogFile, "%s\\%s", DllPath, LogName);
        OCIInitialize(OCI_THREADED|OCI_OBJECT,(dvoid *)0,0,0,0);
        //    sprintf(LogFile, "d:\\%s", LogName);

        /* load DBConnection.dll */

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

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

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

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

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

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

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

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

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

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

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

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

#ifdef NEW_ALLOCATE_FROM

```

```

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

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

readInit(string, "KMaxterms", Default_Maxterms);
userlog("number of Max Terms is %s00\n", string);
/* add one more form for special characters */
if ((Maxterms = atoi(string) * 100 + 1) <= 1) {
    userlog("number of Max Terms is %d\n", Maxterms - 1);
    return FALSE;
}
readInit(string, "DeliveryQueues", Default_DeliveryQueues);
userlog("number of Delivery Queues is %s\n", string);
if ((DeliveryQueues = atoi(string)) <= 0) {
    userlog("number of Delivery Queues is %d\n",
DeliveryQueues);
    return FALSE;
}

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

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

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

    DeleteCriticalSection(&(resp_global_pool.form_template_spinlock
k));
DeleteCriticalSection(&(txn_data_pool.form_template_spinlock));

    int i_type, i_pool;
#define GPOOL txn_global_pool[i_type][i_pool]
    for (i_type = 0; i_type < POOL_TYPE_TXN_MAX; i_type++)
        for (i_pool = 0; i_pool < TXN_TYPE_MAX; i_pool++)
DeleteCriticalSection(&(GPOOL.form_template_spinlock));
#undef GPOOL
}

return TRUE;
}

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

DWORD WINAPI HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB)
{
    if (!modtpcc_ready)
return FALSE;

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

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

```

```

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

return process_query(pECB)==TRUE ? HSE_STATUS_SUCCESS :
HSE_STATUS_ERROR;

/*
HSE_SEND_HEADER_EX_INFO info = { 0 };

char szOut[256];
DWORD nOut;

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

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

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

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

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

return HSE_STATUS_SUCCESS;
*/
}

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

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

    DeliveryThreadstop = 1;

    for (int i=0; i<DeliveryThreads; i++) {
        if (!SetEvent(waitDelWork)) {
            userlog("Error on SetEvent(waitDelWork) on
deleteDelQueue\n");
        }

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

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

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

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

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

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

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

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

#ifdef DELIVERY_MUTEX

```

```

        CloseHandle(ptr->queue_lock);
#endif
        free(ptr->pdata);
        free(ptr);
        ptr=next;
    }
    return TRUE;
}

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

    userlog("execute initDelQueue\n");

    for (int i=0; i<DeliveryQueues; i++) {
        if ((ptr = (DelQueue_info *) malloc(sizeof(DelQueue_info)))
            == NULL) {
            userlog("malloc error in initDelQueue\n");
            return FALSE;
        }
        ptr->pdata=(T_delivery_data
        *)malloc(sizeof(T_delivery_data));
#ifdef DELIVERY_MUTEX
        if ((ptr->queue_lock=CreateMutex(NULL, FALSE, NULL))==NULL)
        {
            userlog("Cannot create mutex on queue lock\n");
            return FALSE;
        }
#endif
        if (!i)
            DelQueue_free=curr=ptr;
        else {
            curr->Next = ptr;
            curr = ptr;
        }
    }

    DelQueue_begin = DelQueue_end = curr->Next = NULL;

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

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

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

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

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

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

    return TRUE;
}

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

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

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

```

```

                userlog("Error on
                WaitForSingleObject(waitDelQueueWork) in initDeliveryThread\n");
                endDeliveryThread(thread_number);
                return;
            }
        }
    }

    if (!DeliveryThreadstop) {
        (void)mod_tpcc_delivery(queue_info->pdata,
        thread_number);
        addFreeDelQueue(queue_info);
    }
}

endDeliveryThread(thread_number);

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

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

DelQueue_info *DequeueDel()
{
    DelQueue_info *ptr;

    if (DelQueue_begin == NULL) return NULL;

    EnterCriticalSection(&critical_DelQueue_work);

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

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

    LeaveCriticalSection(&critical_DelQueue_work);

    return ptr;
}

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

    LeaveCriticalSection(&critical_DelQueue_work);
}

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

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

```



```

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

    EnterCriticalSection(&critical_DelQueue_free);

    while (ptr==NULL) {
        if (DelQueue_free==NULL) {
            LeaveCriticalSection(&critical_DelQueue_free);
            if (WaitForSingleObject(waitAvailableDelQueue, INFINITE)
!= WAIT_OBJECT_0) {
                userlog("WaitForSingleObject(waitAvailableDelQueue) in
findFreeDelQueue\n");
            }
            userlog("Delivery queue is full, sleep for 10 seconds\n");
#ifdef DEBUG
            userlog("used del queue: %d\n", useddel);
#endif
            /* sleep for 10 seconds */
            Sleep(10000);
            EnterCriticalSection(&critical_DelQueue_free);
        }
        else {
            ptr = DelQueue_free;
            DelQueue_free = DelQueue_free->Next;
#ifdef DEBUG
            useddel++;
            if (useddel % 300 == 0)
                userlog("allocate a del queue current used: %d\n",
useddel);
#endif
        }
    }

    LeaveCriticalSection(&critical_DelQueue_free);

    return ptr;
}

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

    form = w_id = ld_id = 0;

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

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

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

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

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

        if (CMD_PROCESS(cmd)) { /* cmd = Process */
            if (form == 'N') {
                /* New Order transaction */
                return mod_neworder_query(pECB, w_id, ld_id, ptr);
            }
            else if (form == 'P') {
                /* Payment order transaction */
                return mod_payment_query(pECB, w_id, ld_id, ptr);
            }
            else if (form == 'D') {
                /* Delivery order transaction */
                return mod_delivery_query(pECB, w_id, ld_id, ptr);
            }
            else if (form == 'O') {
                /* Order Status order transaction */
                return mod_orderstatus_query(pECB, w_id, ld_id,
ptr);
            }
            else if (form == 'S') {
                /* Stock Level order transaction */
                return mod_stocklevel_query(pECB, w_id, ld_id, ptr);
            }
            else {
                return send_error_message(pECB, 0, INVALID_FORM, "", w_id,
ld_id, 0);
            }
        }
        else if (CMD_BEGIN(cmd)) return mod_begin_cmd(pECB);
        else if (CMD_NEWORDER(cmd)) return mod_neworder_cmd(pECB,
w_id, ld_id);
        else if (CMD_PAYMENT(cmd)) return mod_payment_cmd(pECB,
w_id, ld_id);

```

```

        else if (CMD_DELIVERY(cmd)) return mod_delivery_cmd(pECB,
w_id, ld_id);
        else if (CMD_ORDERSTATUS(cmd)) return
mod_orderstatus_cmd(pECB, w_id, ld_id);
        else if (CMD_STOCKLEVEL(cmd)) return
mod_stocklevel_cmd(pECB, w_id, ld_id);
        else if (CMD_EXIT(cmd)) return mod_exit_cmd(pECB);
        else if (CMD_MENU(cmd)) return mod_menu_cmd(pECB,
w_id, ld_id);
        else
            return send_error_message(pECB, 0, COMMAND_UNDEFINED, "",
w_id, ld_id, 0);
    }

    return TRUE;
}

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

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

    return FALSE;
}

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

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

void allocate_response_pool()
{
    int i;

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

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

    txn_global_pool[POOL_TYPE_TXN_INPUT][txn_type].form_template_len
gth = length;

    /* For output form.
    */
    length = sprintf(response_form, FormHeader, mod_name);
    length = build_form_index(response_form,
response_form_template,

```

```

        form_index[POOL_TYPE_TXN_OUTPUT][txn_type],
        length);
        length = (length + 128) & ~(int)7);
txn_global_pool[POOL_TYPE_TXN_OUTPUT][txn_type].form_template_length = length;
return;
}

```

```

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

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

```

```

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

    char
    DeliveryInput[sizeof(DeliveryFormInput_Template)+FORM_PAD];
    char
    OrderStatusInput[sizeof(OrderStatusInput_Template)+FORM_PAD];
    char PaymentInput[sizeof(PaymentInput_Template)+FORM_PAD];
    char NewOrderInput[sizeof(NewOrderInput_Template)+FORM_PAD];
    char
    StockLevelInput[sizeof(StockLevelInput_Template)+FORM_PAD];
    char
    DeliveryOutput[sizeof(DeliveryFormOutput_Template)+FORM_PAD];
    char
    OrderStatusOutput[sizeof(OrderStatusOutput_Template)+FORM_PAD];
    char PaymentOutput[sizeof(PaymentOutput_Template)+FORM_PAD];
    char
    NewOrderOutput[sizeof(NewOrderOutput_Template)+FORM_PAD];
    char
    StockLevelOutput[sizeof(StockLevelOutput_Template)+FORM_PAD];
    int i_type, i_pool, i;

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

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

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

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

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

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

            InitializeCriticalSection(&(GPOOL.form_template_spinlock));

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

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

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

```

```

    }
}

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

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

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

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

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

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

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

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

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

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

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

#undef FORM_PAD
#undef GPOOL
}

```

```

void allocate_transaction_pool()
{
    int i, pool_size;

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

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

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

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

```

```

    if ((getcharvalue(ptr, '5', &ptr) == FALSE) || ((ld_id =
atoi(ptr) <= 0) || (ld_id > 10))
        return sendform_welcome(pECB, "Error: Invalid District
DID");
    /*
    Perform activities related to database logon etc.
    */
    return sendform_mainmenu(pECB, w_id, ld_id);
}

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

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

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

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

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

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

int send_response(EXTENSION_CONTROL_BLOCK *pECB, char *form, int
size)
{
    HSE_SEND_HEADER_EX_INFO info = { 0 };
    char szHeader[256];
    DWORD nOut=size;
    DWORD nHeader = sprintf(szHeader, "Content-Type: text/html\n"
"Content-Length: %d\n" "charset= ISO-8859-1\n\n", size);

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

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

    if (!pECB->WriteClient(pECB->ConnID, form, &nOut,
HSE_IO_SYNC))
    {
        userlog("WriteClient returns false");
        return FALSE;
    }

    /*
    char temp[1000];
    strncpy(temp, form, size);
    temp[strlen(temp)]='\0';
    userlog("send: from >>>%s<<<\n", temp);
    */

    return TRUE;
}

```

```

}

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

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

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

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

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

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

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

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

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

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

    pool = &txn_global_pool[SUBI];

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

```

```

    fill_number(form, WDID(w_id, ld_id),
form_index[SUBI][NO_TERMID].index,
    form_index[SUBI][NO_TERMID].length);
    fill_number(form, w_id, form_index[SUBI][NO_WID].index,
    form_index[SUBI][NO_WID].length);
    ret=send_response(pECB, form, strlen(form));
#endifdef NEW_ALLOCATE_FORM
    free_form(pool, form, index);
#endifif
    return ret;
#undef SUBI
}

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

    pool = &txn_global_pool[SUBI];

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

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

#ifdef NEW_ALLOCATE_FORM
    free_form(pool, form, index);
#endifif

    return ret;
#undef SUBI
}

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

    pool = &txn_global_pool[SUBI];

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

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

#ifdef NEW_ALLOCATE_FORM
    free_form(pool, form, index);
#endifif

    return ret;
#undef SUBI
}

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

    pool = &txn_global_pool[SUBI];

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

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

```

```

#ifdef NEW_ALLOCATE_FORM
    free_form(pool, form, index);
#endifif

    return ret;
#undef SUBI
}

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

    pool = &txn_global_pool[SUBI];

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

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

#ifdef NEW_ALLOCATE_FORM
    free_form(pool, form, index);
#endifif

    return ret;
#undef SUBI
}

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

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

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

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

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

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

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

```

```

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

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

}

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

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

    do {
        *pend = (avalue % 10) + '0';
        avalue = avalue / 10;
        if (--length) pend--;
    } while (length);

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

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

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

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

    return TRUE;
}

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

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

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

```

```

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

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

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

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

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

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

    return ret;
}

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

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

return ret;
}

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

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

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

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

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

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

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

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

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

return ret;
}

```

```

    }
    status = mod_tpcc_payment(pdata);
    ret=sendform_paymentoutput(status, pdata);
#endif NEW_ALLOCATE_FORM
    free_form(&txn_data_pool, (char *) pdata, index);
#endif
    return ret;
}

int mod_orderstatus_query(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id, char *ptr)
{
    T_orderstatus_data *pdata;
    int index = w_id*10+ld_id, ret;
    int status = SUCCESS;
#endif NEW_ALLOCATE_FORM
    pdata = (T_orderstatus_data
*)allocate_form_new(&txn_data_pool, index);
#else
    pdata = (T_orderstatus_data *)allocate_form(&txn_data_pool,
&index);
#endif
    pdata->w_id = w_id; pdata->ld_id = ld_id; pdata->context =
(void *)pECB;

    status = parse_orderstatus_query(ptr, pdata);
    if (status != SUCCESS) {
        ret=send_error_message(pECB, 0, status, "", w_id, ld_id,
0);
#endif NEW_ALLOCATE_FORM
        free_form(&txn_data_pool, (char *) pdata, index);
#endif
        return ret;
    }

    status = mod_tpcc_orderstatus(pdata);
    ret=sendform_orderstatusoutput(status, pdata);
#endif NEW_ALLOCATE_FORM
    free_form(&txn_data_pool, (char *) pdata, index);
#endif
    return ret;
}

int mod_stocklevel_query(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id, char *ptr)
{
    T_stocklevel_data *pdata;
    int index = w_id*10+ld_id, ret;
    int status = SUCCESS;
#endif NEW_ALLOCATE_FORM
    pdata = (T_stocklevel_data
*)allocate_form_new(&txn_data_pool, index);
#else
    pdata = (T_stocklevel_data *)allocate_form(&txn_data_pool,
&index);
#endif
    pdata->w_id = w_id; pdata->ld_id = ld_id; pdata->context =
(void *)pECB;

    status = parse_stocklevel_query(ptr, pdata);
    if (status != SUCCESS) {
        ret=send_error_message(pECB, 0, status, "", w_id, ld_id,
0);
#endif NEW_ALLOCATE_FORM
        free_form(&txn_data_pool, (char *) pdata, index);
#endif
        return ret;
    }

    status = mod_tpcc_stocklevel(pdata);
    ret=sendform_stockleveloutput(status, pdata);
#endif NEW_ALLOCATE_FORM
    free_form(&txn_data_pool, (char *) pdata, index);
#endif
    return ret;
}

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

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

```

```

    status = parse_query_string(iptr, NO_INPUT_MAX,
neworder_chars, value_ptr);
    if ((ptr = value_ptr[NO_INPUT_DID].value) == NULL) {
        return NEWORDER_MISSING_DID;
    }
    if ((status = get_number(ptr, &pdata->d_id)) == FALSE) {
        return NEWORDER_DISTRICT_INVALID;
    }
    if ((pdata->d_id > 10) || (pdata->d_id < 1)) {
        return NEWORDER_DISTRICT_RANGE;
    }
    if ((ptr = value_ptr[NO_INPUT_CID].value) == NULL) {
        return NEWORDER_CUSTOMER_KEY;
    }
    if ((status = get_number(ptr, &pdata->c_id)) == FALSE) {
        return NEWORDER_CUSTOMER_INVALID;
    }
    if ((pdata->c_id > 3000) || (pdata->c_id <= 0)) {
        return NEWORDER_CUSTOMER_RANGE;
    }

    pdata->o_all_local = 1;

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

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

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

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

```

```

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

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

    see_dot = FALSE;

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

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

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

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

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

    if ((ptr = value_ptr[DE_INPUT_QTIME].value) == NULL) {
        pdata->enqueue_time = (double) GetTickCount();
    }
}

```

```

        return SUCCESS;
    }

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

    see_dot = FALSE;

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

    pdata->enqueue_time = atof(ptr);
}

return SUCCESS;

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

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

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

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

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

int parse_stocklevel_query(char *iptr, T_stocklevel_data *pdata)
{
    value_index_entry value_ptr[SL_INPUT_MAX];
    char *ptr;
    int status = SUCCESS;

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

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

return SUCCESS;
}

```

```

}

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

int sendform_neworderoutput(int status, T_neworder_data *pdata)
{
    EXTENSION_CONTROL_BLOCK *pECB;
    int w_id, ld_id, ret;
    char *form, *form2;
    char blank[] = " ";

    int index = -1, formlen, strcount=0, shift=0, i, j,
    lineStart=15;
    int indexes[NO_FORMINDEX_SIZE], indLen[NO_FORMINDEX_SIZE],
    index2=-1;
    form_template_pool *pool;

#define SUBI POOL_TYPE_TXN_OUTPUT[TXN_TYPE_NEWORDER
    w_id = pdata->w_id; ld_id = pdata->ld_id;
    pECB = (EXTENSION_CONTROL_BLOCK *) pdata->context;

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

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

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

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

    formlen=strlen(form);

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

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

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

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

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

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

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

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

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

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

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

    if (!pdata->status) {

```

```

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

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

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

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

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

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

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

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

for (j=NO_SUPPW+i*8; j<NO_SUPPW+15*8; j++)
memcpy(form+form_index[SUBI][j].index, blank, form_index[SUBI][j].
length);

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

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

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

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

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

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

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

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

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

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

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

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

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

```



```

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

    return ret;
#undef SUBI
}

int sendform_paymentoutput(int status, T_payment_data *pdata)
{
    EXTENSION_CONTROL_BLOCK *pECB;
    int w_id, ld_id, ret;
    char *form, *form2;
    char blank[] = " ";

    int index = -1, formlen, strcount=0, shift=0, i=0,
    j, datalen;
    int indexes[PA_FORMINDEX_SIZE], indLen[PA_FORMINDEX_SIZE],
    index2=-1;
    form_template_pool *pool;

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

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

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

#define SUBI POOL_TYPE_TXN_OUTPUT[TXN_TYPE_PAYMENT
    pool = &txn_global_pool[SUBI];
    index=w_id*10+ld_id;

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

    formlen=strlen(form);

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

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

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

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

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

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

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

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

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

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

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

    fill_string(form, pdata->d_city,
    form_index[SUBI][PA_DCITY].index,

```

```

    form_index[SUBI][PA_DCITY].length, &shift);
    indexes[strcount]=form_index[SUBI][PA_DCITY].index;
    indLen[strcount++]=form_index[SUBI][PA_DCITY].length;

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

    fill_float(form, pdata->c_balance,
    form_index[SUBI][PA_CBAL].index,
    form_index[SUBI][PA_CBAL].length);

    fill_float(form, pdata->c_credit_lim,
    form_index[SUBI][PA_LIMIT].index,

```

```

        form_index[SUBI][PA_LIMIT].length);

    if (pdata->c_credit[0]=='B' && pdata->c_credit[1]=='C') {
        datalen=strlen(pdata->c_data);
        for (i=0; i<4; i++) {
            if (i * form_index[SUBI][PA_CUSTDATA+i].length >=
datalen) break;
            fill_string(form, pdata-
>c_data+(i*form_index[SUBI][PA_CUSTDATA+i].length),
                form_index[SUBI][PA_CUSTDATA+i].index,
                form_index[SUBI][PA_CUSTDATA+i].length,
                &shift);
            indexes[strcount]=form_index[SUBI][PA_CUSTDATA+i].index;
        }
        indLen[strcount++]=form_index[SUBI][PA_CUSTDATA+i].length;
    }

    for (j=i; j<4; j++)
        memcpy(form+form_index[SUBI][PA_CUSTDATA+j].index, blank,
            form_index[SUBI][PA_CUSTDATA+j].length);

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

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

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

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

    return ret;
#undef SUBI
}

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

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

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

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

#define SUBI POOL_TYPE_TXN_OUTPUT][TXN_TYPE_ORDERSTATUS
    pool = &txn_global_pool[SUBI];
    index=w_id*10+ld_id;

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

    formlen = strlen(form);

    fill_number(form, WID(w_id, ld_id),
form_index[SUBI][OS_TERMID].index,
                form_index[SUBI][OS_TERMID].length);
    fill_number(form, w_id, form_index[SUBI][OS_WID].index,
                form_index[SUBI][OS_WID].length);
    fill_number(form, pdata->d_id,
form_index[SUBI][OS_DID].index,
                form_index[SUBI][OS_DID].length);
    fill_number(form, pdata->c_id,
form_index[SUBI][OS_CID].index,
                form_index[SUBI][OS_CID].length);
    fill_string(form, pdata->c_first,
form_index[SUBI][OS_FIRST].index,
                form_index[SUBI][OS_FIRST].length, &shift);
    indexes[strcount]=form_index[SUBI][OS_FIRST].index;
    indLen[strcount++]=form_index[SUBI][OS_FIRST].length;

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

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

```

```

        indLen[strcount++]=form_index[SUBI][OS_LAST].length;

        fill_float(form, pdata->c_balance,
form_index[SUBI][OS_CBALANCE].index,
                form_index[SUBI][OS_CBALANCE].length);

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

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

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

        for (i=0; i < pdata->o_ol_cnt; i++) {
            fill_number(form, pdata->o_orderline[i].ol_supply_w_id,
form_index[SUBI][OS_SUPW+i*5].index,
                form_index[SUBI][OS_SUPW+i*5].length);

            fill_number(form, pdata->o_orderline[i].ol_i_id,
form_index[SUBI][OS_ITEMID+i*5].index,
                form_index[SUBI][OS_ITEMID+i*5].length);

            fill_number(form, pdata->o_orderline[i].ol_quantity,
form_index[SUBI][OS_QTY+i*5].index,
                form_index[SUBI][OS_QTY+i*5].length);

            fill_float(form, pdata->o_orderline[i].ol_amount,
form_index[SUBI][OS_AMOUNT+i*5].index,
                form_index[SUBI][OS_AMOUNT+i*5].length);

            fill_string(form, pdata-
>o_orderline[i].ol_delivery_d.DateString,
                form_index[SUBI][OS_DELDATE+i*5].index,
                form_index[SUBI][OS_DELDATE+i*5].length,
                &shift);
            indexes[strcount]=form_index[SUBI][OS_DELDATE+i*5].index;
        }
        indLen[strcount++]=form_index[SUBI][OS_DELDATE+i*5].length;

        for (lineStart=j=i; j<15; j++) {
            memcpy(form+form_index[SUBI][OS_SUPW+j*5].index, blank,
form_index[SUBI][OS_SUPW+j*5].length);
            memcpy(form+form_index[SUBI][OS_ITEMID+j*5].index, blank,
form_index[SUBI][OS_ITEMID+j*5].length);
            memcpy(form+form_index[SUBI][OS_QTY+j*5].index, blank,
form_index[SUBI][OS_QTY+j*5].length);
            memcpy(form+form_index[SUBI][OS_AMOUNT+j*5].index-1,
blank,
                form_index[SUBI][OS_AMOUNT+j*5].length+1);
            memcpy(form+form_index[SUBI][OS_DELDATE+j*5].index, blank,
form_index[SUBI][OS_DELDATE+j*5].length);
        }

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

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

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

        for (j=lineStart; j<15; j++)
            form[form_index[SUBI][OS_AMOUNT+j*5].index-1]='$';

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

    return ret;
#undef SUBI
}

int sendform_deliveryoutput(int status, T_delivery_data *pdata)
{
    EXTENSION_CONTROL_BLOCK *pECB;
    int w_id, ld_id;
    char *form;
    int index = -1, ret;
    form_template_pool *pool;

    w_id = pdata->w_id; ld_id = pdata->ld_id;
    pECB = (EXTENSION_CONTROL_BLOCK *) pdata->context;
    if (status != SUCCESS && status != DB_SUCCESS) {
        return send_error_message(pECB, 0, status, "", w_id,
ld_id, 0);
    }

#define SUBI POOL_TYPE_TXN_OUTPUT][TXN_TYPE_DELIVERY
    pool = &txn_global_pool[SUBI];
    index=w_id*10+ld_id;

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

```

```

    form = allocate_form(pool, &index);
#endif

    fill_number(form, WIDID(w_id, ld_id),
form_index[SUBI][DE_TERMID].index,
                form_index[SUBI][DE_TERMID].length);
    fill_number(form, w_id, form_index[SUBI][DE_WID].index,
                form_index[SUBI][DE_WID].length);
    fill_number(form, pdata->o_carrier_id,
form_index[SUBI][DE_CARID].index,
                form_index[SUBI][DE_CARID].length);

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

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

    return ret;
#undef SUBI
}

int sendform_stockleveloutput(int status, T_stocklevel_data
*pdata)
{
    EXTENSION_CONTROL_BLOCK *pECB;
    int w_id, ld_id;
    char *form;
    int index = -1, ret;
    form_template_pool *pool;

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

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

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

#define SUBI POOL_TYPE_TXN_OUTPUT[TXN_TYPE_STOCKLEVEL

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

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

    fill_number(form, WIDID(w_id, ld_id),
form_index[SUBI][SL_TERMID].index,
                form_index[SUBI][SL_TERMID].length);
    fill_number(form, w_id, form_index[SUBI][SL_WID].index,
                form_index[SUBI][SL_WID].length);
    fill_number(form, ld_id, form_index[SUBI][SL_DID].index,
                form_index[SUBI][SL_DID].length);
    fill_number(form, pdata->threshold,
form_index[SUBI][SL_THRESHOLD].index,
                form_index[SUBI][SL_THRESHOLD].length);
    fill_number(form, pdata->low_stock,
form_index[SUBI][SL_LOWSTOCK].index,
                form_index[SUBI][SL_LOWSTOCK].length);

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

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

    return ret;
#undef SUBI
}

int (FAR * mod_tpcc_neworder)(T_neworder_data *);
int (FAR * mod_tpcc_payment)(T_payment_data *);
int (FAR * mod_tpcc_delivery)(T_delivery_data *, int);
int (FAR * mod_tpcc_orderstatus)(T_orderstatus_data *);
int (FAR * mod_tpcc_stocklevel)(T_stocklevel_data *);
void (FAR * userlog)(char * str, ...);
void (FAR * initDelLog)(int);
void (FAR * endDelLog)(int);

-----
mod_tpcc_error.h
-----
/* Copyright (c) 2004, Oracle Corporation. All rights reserved.
*/

/*
NAME
    mod_tpcc_error.h - <one-line expansion of the name>

DESCRIPTION
    <short description of facility this file declares/defines>

RELATED DOCUMENTS

```

```

    <note any documents related to this facility>

EXPORT FUNCTION(S)
    <external functions declared for use outside package - one-
line descriptions>

INTERNAL FUNCTION(S)
    <other external functions declared - one-line descriptions>

EXAMPLES

NOTES
    <other useful comments, qualifications, etc.>

MODIFIED    (MM/DD/YY)
xnie        02/09/04 - to make it work with tuxedo
shuang      01/22/04 - shuang_rte
shuang      01/21/04 - Creation

*/

#define DB_SUCCESS          0
#define DB_ERROR           1
#define TRANSPORT_ERROR    2
#define DB_INTERFACE       3
#define DB_DEADLOCK_LIMIT  4
#define DB_NOT_COMMITTED   5
#define DB_DEAD            6
#define DB_PENDING         7
#define DB_NOT_LOGGED_IN   8
#define DB_LOGIN_FAILED    9
#define DB_USE_FAILED      10
#define DB_LOGOUT_FAILED   11
#define DB_TUXEDO_TPALLOC_ERROR 12
#define DB_TUXEDO_TPCALL_ERROR 13
#define DB_MAX_ERR         13
#define VALID_DB_ERR(err) ((err) >= DB_SUCCESS)&&((err) <=
DB_MAX_ERR)

#define SUCCESS             1000
#define COMMAND_UNDEFINED  1001
#define NOT_IMPLEMENTED_YET 1002
#define CANNOT_INIT_TERMINAL 1003
#define OUT_OF_MEMORY      1004
#define NEW_ORDER_NOT_PROCESSED 1005
#define PAYMENT_NOT_PROCESSED 1006
#define NO_SERVER_SPECIFIED 1007
#define ORDER_STATUS_NOT_PROCESSED 1008
#define W_ID_INVALID       1009
#define CAN_NOT_SET_MAX_CONNECTIONS 1010
#define UNKNOW_TRANSACTION_TYPE 1011
#define D_ID_INVALID       1012
#define MAX_CONNECT_PARAM  1013
#define INVALID_SYNC_CONNECTION 1014
#define INVALID_TERMID     1015
#define PAYMENT_INVALID_CUSTOMER 1016
#define SQL_OPEN_CONNECTION 1017

#define STOCKLEVEL_MISSING_THRESHOLD_KEY 1018
#define STOCKLEVEL_THRESHOLD_INVALID 1019
#define STOCKLEVEL_THRESHOLD_RANGE 1020
#define STOCKLEVEL_NOT_PROCESSED 1021
#define NEWORDER_MISSING_DID 1022
#define NEWORDER_DISTRICT_INVALID 1023
#define NEWORDER_DISTRICT_RANGE 1024
#define NEWORDER_CUSTOMER_KEY 1025
#define NEWORDER_CUSTOMER_INVALID 1026
#define NEWORDER_CUSTOMER_RANGE 1027
#define NEWORDER_MISSING_ID_KEY 1028
#define NEWORDER_ITEM_BLANK_LINES 1029
#define NEWORDER_ITEMID_INVALID 1030
#define NEWORDER_MISSING_SUPPW_KEY 1031
#define NEWORDER_SUPPW_INVALID 1032
#define NEWORDER_MISSING_QTY_KEY 1033
#define NEWORDER_QTY_INVALID 1034
#define NEWORDER_SUPPW_RANGE 1035
#define NEWORDER_ITEMID_RANGE 1036
#define NEWORDER_QTY_RANGE 1037
#define NEWORDER_SUPPW_WITHOUT_ITEMID 1039
#define NEWORDER_QTY_WITHOUT_ITEMID 1040
#define NEWORDER_NOITEMS_ENTERED 1041
#define PAYMENT_MISSING_DID_KEY 1042
#define PAYMENT_DISTRICT_INVALID 1043
#define PAYMENT_DISTRICT_RANGE 1043
#define PAYMENT_MISSING_CID_KEY 1044
#define PAYMENT_CUSTOMER_INVALID 1045
#define PAYMENT_MISSING_CLASTNAME 1046
#define PAYMENT_LAST_NAME_TO_LONG 1047
#define PAYMENT_CID_RANGE 1048
#define PAYMENT_CID_AND_CLASTNAME 1049
#define PAYMENT_MISSING_CDI_KEY 1050
#define PAYMENT_CDI_INVALID 1051
#define PAYMENT_CDI_RANGE 1052
#define PAYMENT_MISSING_CWI_KEY 1053
#define PAYMENT_CWI_INVALID 1054
#define PAYMENT_CWI_RANGE 1055
#define PAYMENT_MISSING_HAM_KEY 1056
#define PAYMENT_HAM_INVALID 1057
#define PAYMENT_HAM_RANGE 1058
#define ORDERSTATUS_MISSING_DID_KEY 1059
#define ORDERSTATUS_DID_INVALID 1060
#define ORDERSTATUS_DID_RANGE 1061
#define ORDERSTATUS_MISSING_CID_KEY 1062
#define ORDERSTATUS_MISSING_CLASTNAME_KEY 1063
#define ORDERSTATUS_CLASTNAME_RANGE 1064
#define ORDERSTATUS_CID_INVALID 1065
#define ORDERSTATUS_CID_RANGE 1066
#define ORDERSTATUS_CID_AND_CLASTNAME 1067

```

```

#define DELIVERY_MISSING_OCD_KEY 1068
#define DELIVERY_CARRIER_INVALID 1069
#define DELIVERY_CARRIER_ID_RANGE 1070

#define PAYMENT_MISSING_CLASTNAME_KEY 1071
#define CANT_FIND_TPCC_KEY 1072
#define CANT_FIND_INETINFO_KEY 1073
#define CANT_FIND_POOLTHREADLIMIT 1074
#define DB_DELIVERY_NOT_QUEUED 1075
#define DELIVERY_NOT_PROCESSED 1076
#define TERM_ALLOCATE_FAILED 1077
#define PENDING 1078
#define CANT_START_PRCDINIT_THREAD 1079
#define CANT_START_DELIVERY_THREAD 1080
#define GOVERNOR_VALUE_NOT_FOUND 1081
#define SERVER_MISMATCH 1082
#define DATABASE_MISMATCH 1083
#define USER_MISMATCH 1084
#define PASSWORD_MISMATCH 1085
#define CANT_CREATE_ALL_THREADS_EVENT 1086
#define CANT_CREATE_FORCE_THRED_STRT_EVENT 1087
#define CANT_ALLOCATE_THREAD_LOCAL_STORAGE 1088
#define CANT_SET_THREAD_LOCAL_STORAGE 1089
#define FORCE_CONNECT_THREAD_FAILED 1090
#define CANT_FIND_SERVER_VALUE 1091
#define NO_MESSAGE 1092
#define CANT_FIND_PATH_VALUE 1093
#define CANNOT_CREATE_RESULTS_FILE 1094
#define DELIVERY_PIPE_SECURITY 1095
#define DELIVERY_PIPE_CREATE 1096
#define DELIVERY_PIPE_OPEN 1097
#define DELIVERY_PIPE_READ 1098
#define DELIVERY_PIPE_DISCONNECT 1099
#define CANT_FIND_DATABASE_VALUE 1100
#define CANT_FIND_USER_VALUE 1101
#define CANT_FIND_PASSWORD_VALUE 1102
#define DELIVERY_OUTPUT_PIPE_WRITE 1103
#define DELIVERY_OUTPUT_PIPE_READ 1104
#define DELIVERY_MISSING_QUEUEITEM_KEY 1105
#define DELIVERY_QUEUEITEM_INVALID 1106
#define ALREADY_LOGGED_IN 1107
#define INVALID_FORM 1109
#define DELIVERY_MUST_CONNECTDB 1110
#define INVALID_FORM_AND_CMD_NOT_BEGIN 1111
#define MAX_CONNECTIONS_EXCEEDED 1112
#define CANNOT_FIND_CONNECTION 1113
#define CKPT_NOT_INITIALIZED 1114
#define PAYMENT_MISSING_CID_CLASTNAME 1115
#define CANT_FIND_MAXDBCONNECTIONS_VALUE 1116
#define PAYMENT_CUSTOMER_RANGE 1117

/* OCI return status */

#define DB_RETURN_OCI_SUCCESS 1118
#define DB_RETURN_OCI_SUCCESS_WITH_INFO 1119
#define DB_RETURN_OCI_NEED_DATA 1120
#define DB_RETURN_OCI_NO_DATA 1121
#define DB_RETURN_OCI_ERROR 1122
#define DB_RETURN_OCI_INVALID_HANDLE 1123
#define DB_RETURN_OCI_STILL_EXECUTING 1124
#define DB_RETURN_OCI_CONTINUE 1125

struct T_error_message
{
    int error_code;
    char error_mesg[80];
};
typedef struct T_error_message T_error_message;

T_error_message error_message [] =
{
    { SUCCESS, "Success, no error." },
    { NO_MESSAGE, "No message string available for the specified error code." },
    { COMMAND_UNDEFINED, "Command undefined." },
    { NOT_IMPLEMENTED_YET, "Not Implemented Yet." },
    { CANNOT_INIT_TERMINAL, "Cannot initialize client connection." },
    { OUT_OF_MEMORY, "Insufficient memory." },
    { NEW_ORDER_NOT_PROCESSED, "Cannot process new Order form." },
    { PAYMENT_NOT_PROCESSED, "Cannot process payment form." },
    { NO_SERVER_SPECIFIED, "No Server name specified." },
    { ORDER_STATUS_NOT_PROCESSED, "Cannot process order status form." },
    { W_ID_INVALID, "Invalid Warehouse ID." },
    { CAN_NOT_SET_MAX_CONNECTIONS, "Insufficient memory to allocate # connections." },
    { D_ID_INVALID, "Invalid District ID Must be 1 to 10." },
    { MAX_CONNECT_PARAM, "Max client connections exceeded, run install to increase." },
    { INVALID_SYNC_CONNECTION, "Invalid Terminal Sync ID." },
    { INVALID_TERMID, "Invalid Terminal ID." },
    { PAYMENT_INVALID_CUSTOMER, "Payment Form, No such Customer." },
    { SQL_OPEN_CONNECTION, "SQLOpenConnection API Failed." },
    { STOCKLEVEL_MISSING_THRESHOLD_KEY, "Stock Level missing Threshold key \"TT*\"." },
    { STOCKLEVEL_THRESHOLD_INVALID, "Stock Level Threshold invalid data type range = 1 - 99." },
    { STOCKLEVEL_THRESHOLD_RANGE, "Stock Level Threshold out of range, range must be 1 - 99." },
    { STOCKLEVEL_NOT_PROCESSED, "Stock Level not processed." },
    { NEWORDER_MISSING_DID, "New Order missing District key \"DID*\"." },
    { NEWORDER_DISTRICT_INVALID, "New Order District ID Invalid range 1 - 10." },
    { NEWORDER_DISTRICT_RANGE, "New Order District ID out of Range. Range = 1 - 10." },
    { NEWORDER_CUSTOMER_KEY, "New Order missing Customer key \"CID*\"." },
    { NEWORDER_CUSTOMER_INVALID, "New Order customer id invalid data type, range = 1 to 3000." },
    { NEWORDER_CUSTOMER_RANGE, "New Order customer id out of range, range = 1 to 3000." },
    { NEWORDER_MISSING_IID_KEY, "New Order missing Item Id key \"IID*\"." },
    { NEWORDER_ITEM_BLANK_LINES, "New Order blank order lines all orders must be continuous." },
    { NEWORDER_ITEMID_INVALID, "New Order Item Id is wrong data type, must be numeric." },
    { NEWORDER_MISSING_SUPPW_KEY, "New Order missing Supp_W key \"SP##*\"." },
    { NEWORDER_SUPPW_INVALID, "New Order Supp_W invalid data type must be numeric." },
    { NEWORDER_MISSING_QTY_KEY, "New Order Missing Qty key \"Qty##*\"." },
    { NEWORDER_QTY_INVALID, "New Order Qty invalid must be numeric range 1 - 99." },
    { NEWORDER_SUPPW_RANGE, "New Order Supp_W value out of range range = 1 - Max Warehouses." },
    { NEWORDER_ITEMID_RANGE, "New Order Item Id is out of range. Range = 1 to 999999." },
    { NEWORDER_QTY_RANGE, "New Order Qty is out of range. Range = 1 to 99." },
    { PAYMENT_DISTRICT_INVALID, "Payment District ID is invalid must be 1 - 10." },
    { NEWORDER_SUPPW_WITHOUT_ITEMID, "New Order Supp_W field entered without a corresponding Item.Id." },
    { NEWORDER_QTY_WITHOUT_ITEMID, "New Order Qty entered without a corresponding Item.Id." },
    { NEWORDER_NOITEMS_ENTERED, "New Order Blank Items between items, items must be continuous." },
    { PAYMENT_MISSING_DID_KEY, "Payment missing District key \"DID*\"." },
    { PAYMENT_DISTRICT_RANGE, "Payment District Out of range, range = 1 - 10." },
    { PAYMENT_MISSING_CID_KEY, "Payment missing Customer key \"CID*\"." },
    { PAYMENT_CUSTOMER_INVALID, "Payment Customer data type invalid, must be numeric." },
    { PAYMENT_MISSING_CLASTNAME, "Payment missing Customer Last Name key \"CLASTNAME*\"." },
    { PAYMENT_MISSING_CID_CLASTNAME, "Payment entered without Customer ID or last Name." },
    { PAYMENT_LAST_NAME_TO_LONG, "Payment Customer last name longer than 16 characters." },
    { PAYMENT_CUSTOMER_RANGE, "Payment Customer ID out of range, must be 1 to 3000." },
    { PAYMENT_CID_AND_CLASTNAME, "Payment Customer ID and Last Name entered must be one or other." },
    { PAYMENT_MISSING_CDI_KEY, "Payment missing Customer district key \"CDI*\"." },
    { PAYMENT_CDI_INVALID, "Payment Customer district invalid must be numeric." },
    { PAYMENT_CDI_RANGE, "Payment Customer district out of range must be 1 - 10." },
    { PAYMENT_MISSING_CWI_KEY, "Payment missing Customer Warehouse key \"CWI*\"." },
    { PAYMENT_CWI_INVALID, "Payment Customer Warehouse invalid must be numeric." },
    { PAYMENT_CWI_RANGE, "Payment Customer Warehouse out of range, 1 to Max Warehouses." },
    { PAYMENT_MISSING_HAM_KEY, "Payment missing Amount key \"HAM*\"." },
    { PAYMENT_HAM_INVALID, "Payment Amount invalid data type must be numeric." },
    { PAYMENT_HAM_RANGE, "Payment Amount out of range, 0 - 9999.99." },
    { ORDERSTATUS_MISSING_DID_KEY, "Order Status missing District key \"DID*\"." },
    { ORDERSTATUS_DID_INVALID, "Order Status District invalid, value must be numeric 1 - 10." },
    { ORDERSTATUS_DID_RANGE, "Order Status District out of range must be 1 - 10." },
    { ORDERSTATUS_MISSING_CID_KEY, "Order Status missing Customer key \"CID*\"." },
    { ORDERSTATUS_MISSING_CLASTNAME_KEY, "Order Status missing Customer Last Name key \"CLASTNAME*\"." },
    { ORDERSTATUS_CLASTNAME_RANGE, "Order Status Customer last name longer than 16 characters." },
    { ORDERSTATUS_CID_INVALID, "Order Status Customer ID invalid, range must be numeric 1 - 3000." },
    { ORDERSTATUS_CID_RANGE, "Order Status Customer ID out of range must be 1 - 3000." },
    { ORDERSTATUS_CID_AND_CLASTNAME, "Order Status Customer ID and LastName entered must be only one." },
    { DELIVERY_MISSING_OCD_KEY, "Delivery missing Carrier ID key \"OCD*\"." },
    { DELIVERY_CARRIER_INVALID, "Delivery Carrier ID invalid must be numeric 1 - 10." },
    { DELIVERY_CARRIER_ID_RANGE, "Delivery Carrier ID out of range must be 1 - 10." },
    { PAYMENT_MISSING_CLASTNAME_KEY, "Payment missing Customer Last Name key \"CLASTNAME*\"." },
    { DB_ERROR, "A Database error has occurred." },
    { DB_TUXEDO_TPALLOC_ERROR, "Tuxedo call tpalloc has failed." },
    { DB_TUXEDO_TPCALL_ERROR, "Tuxedo call tpcall has failed." },
    { DELIVERY_NOT_PROCESSED, "Delivery not processed." },
    { DB_DELIVERY_NOT_QUEUED, "Delivery not queued." },
    { CANT_FIND_TPCC_KEY, "TPCC key not found in registry." },
    { CANT_FIND_INETINFO_KEY, "inetinfo key not found in registry." },
    { CANT_FIND_POOLTHREADLIMIT, "PoolThreadLimit value not set in inetinfo\\Parameters key." },
}

```

```

{ TERM_ALLOCATE_FAILED, "Failed to allocate terminal data
structure." },
{ DELIVERY_PIPE_SECURITY, "Failed to initialize delivery pipe
security." },
{ DELIVERY_PIPE_CREATE, "Failed to create delivery pipe." },
{ DELIVERY_PIPE_OPEN, "Failed to open delivery pipe." },
{ DELIVERY_PIPE_READ, "Failed to read delivery pipe." },
{ DELIVERY_PIPE_DISCONNECT, "Failed to start delivery pipe
disconnect thread." },
{ PENDING, "Transaction pending." },
{ CANT_START_FRCDINIT_THREAD, "Can't start Forced
Initialization thread." },
{ CANT_START_DELIVERY_THREAD, "Can't start delivery thread."
},
{ GOVERNOR_VALUE_NOT_FOUND, "Governor value not found in
Registry." },
{ SERVER_MISMATCH, "Server does not match registry value." },
{ DATABASE_MISMATCH, "Database name does not match registry
value." },
{ USER_MISMATCH, "User name does not match registry value." },
{ PASSWORD_MISMATCH, "Password does not match registry value."
},
{ CANT_CREATE_ALL_THREADS_EVENT, "Can't create All Threads
Event." },
{ CANT_CREATE_FORCE_THRED_STRT_EVENT, "Can't create Force
Thread Start Event." },
{ CANT_ALLOCATE_THREAD_LOCAL_STORAGE, "Can't allocate thread
local storage" },
{ CANT_SET_THREAD_LOCAL_STORAGE, "Can't set thread local
storage." },
{ FORCE_CONNECT_THREAD_FAILED, "At least one database connect
call failed, check log files for specific error." },
{ CANT_FIND_SERVER_VALUE, "Server value not set in TPCC key."
},
{ CANT_FIND_PATH_VALUE, "PATH value not set in TPCC key." },
{ CANNOT_CREATE_RESULTS_FILE, "Cannot create results file." },
{ CANT_FIND_DATABASE_VALUE, "Database value not set in TPCC
key." },
{ CANT_FIND_USER_VALUE, "User value not set in TPCC key." },
{ CANT_FIND_PASSWORD_VALUE, "Password value not set in TPCC
key." },
{ DELIVERY_OUTPUT_PIPE_WRITE, "Failed to write output delivery
pipe." },
{ DELIVERY_OUTPUT_PIPE_READ, "Failed to read output delivery
pipe." },
{ DELIVERY_MISSING_QUEUEUETIME_KEY, "Delivery queue time missing
from query." },
{ DELIVERY_QUEUEUETIME_INVALID, "Delivery queue time is
invalid." },
{ ALREADY_LOGGED_IN, "TPCCConnectDB has already been called."
},
{ DB_NOT_LOGGED_IN, "TPCCConnectDB has not yet been called."
},
{ INVALID_FORM, "The FORM field is missing or invalid." },
{ DELIVERY_MUST_CONNECTDB, "Synchronous transport requires
delivery server connect to database." },
{ INVALID_FORM_AND_CMD_NOT_BEGIN, "The FORM field is missing
and CMD is not Begin." },
{ MAX_CONNECTIONS_EXCEEDED, "The maximum number of connections
has been exceeded." },
{ CANT_FIND_MAXDBCONNECTIONS_VALUE, "MaxDBConnections value
not set in TPCC key." },
{ CANNOT_FIND_CONNECTION, "Transport layer unable to find a
DBContext corresponding to the CallersContext." },
{ CKPT_NOT_INITIALIZED, "The checkpoint subsystem has not been
started." },
{ DB_RETURN_OCI_SUCCESS, "OCI SUCCESS" },
{ DB_RETURN_OCI_SUCCESS_WITH_INFO, "OCI SUCCESS WITH INFO" },
{ DB_RETURN_OCI_NEED_DATA, "OCI NEED DATA" },
{ DB_RETURN_OCI_NO_DATA, "OCI NO DATA" },
{ DB_RETURN_OCI_ERROR, "OCI ERROR" },
{ DB_RETURN_OCI_INVALID_HANDLE, "OCI INVALID HANDLE" },
{ DB_RETURN_OCI_STILL_EXECUTING, "OCI STILL EXECUTING" },
{ DB_RETURN_OCI_CONTINUE, "OCI CONTINUE" },
{ 0, "" }
};

```

mod_tpcc.h

```

/* Copyright (c) 2004, Oracle Corporation. All rights reserved.
*/
/*
NAME
mod_tpcc.h - <one-line expansion of the name>
DESCRIPTION
<short description of facility this file declares/defines>
RELATED DOCUMENTS
<note any documents related to this facility>
EXPORT FUNCTION(S)
<external functions declared for use outside package - one-
line descriptions>
INTERNAL FUNCTION(S)
<other external functions declared - one-line descriptions>
EXAMPLES
NOTES
<other useful comments, qualifications, etc.>
MODIFIED (MM/DD/YY)

```

```

xnie 01/30/04 - the real mod_tpcc.h
shuang 01/22/04 - shuang_rte
shuang 01/21/04 - Creation

```

```

*/
#include <httpext.h>

#define CMD_PROCESS(p) (p[0] == 'P') && (p[1] == 'r')
#define CMD_NEWORDER(p) (p[0] == 'N')
#define CMD_PAYMENT(p) (p[0] == 'P') && (p[1] == 'a')
#define CMD_DELIVERY(p) (p[0] == 'D')
#define CMD_ORDERSTATUS(p) (p[0] == 'O')
#define CMD_STOCKLEVEL(p) (p[0] == 'S')
#define CMD_EXIT(p) (p[0] == 'E')
#define CMD_MENU(p) (p[0] == 'M')
#define CMD_BEGIN(p) (p[0] == 'B')

#define TXN_TYPE_DELIVERY 0
#define TXN_TYPE_STOCKLEVEL 1
#define TXN_TYPE_NEWORDER 2
#define TXN_TYPE_ORDERSTATUS 3
#define TXN_TYPE_PAYMENT 4
#define TXN_TYPE_MAX 5

#define POOL_TYPE_TXN_INPUT 0
#define POOL_TYPE_TXN_OUTPUT 1
#define POOL_TYPE_TXN_MAX 2

#define MAX_FORM_INDEX 164
#define BUF_SIZE 4096
#define FILENAMESIZE 128
#define MYLOGFILE "/tmp/mod_tpcc.log"
#define WDDID(w_id,d_id) (10 * w_id + (d_id - 1))

#define MAX(a, b) ((a > b) ? a : b)
#define MIN(a, b) ((a > b) ? b : a)
#define STRING_UPPERCASE(x) \
{ \
int str_pos; \
int len = strlen(x); \
for (str_pos=0; str_pos < len; str_pos++) \
x[str_pos] = toupper(x[str_pos]); \
}

struct value_index_entry
{
char *value;
int length;
};
typedef struct value_index_entry value_index_entry;

struct form_index_entry
{
int index;
int length;
};
typedef struct form_index_entry form_index_entry;

struct form_template_pool
{
CRITICAL_SECTION form_template_spinlock;
/* mutex for
serialization */
int form_template_length; /* Length
of each form */
int form_template_size; /* Number of
form in the pool */
char *form_template_storage; /* The space allocated for
the whole pool */
int free_slot;
int *free_list;
};
typedef struct form_template_pool form_template_pool;

//static int tpcc_handler(request_rec *r);
//static int tpcc_post_config(apr_pool_t *p, apr_pool_t *pl,
// apr_pool_t *pt, server_rec *s);
//static void tpcc_child_init(apr_pool_t *p, server_rec *s);
//static void tpcc_register_hooks(apr_pool_t *p);

void allocate_response_pool();
void allocate_transaction_pool();
void allocate_template_pool();

int sendform_mainmenu(EXTENSION_CONTROL_BLOCK *pECB, int w_id,
int ld_id);
int sendform_welcome(EXTENSION_CONTROL_BLOCK *, char *);
int sendform_neworderinput(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id);
int sendform_paymentinput(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id);
int sendform_orderstatusinput(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id);
int sendform_deliveryinput(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id);
int sendform_stocklevelinput(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id);

int mod_neworder_query(EXTENSION_CONTROL_BLOCK *pECB, int w_id,
int ld_id, char *ptr);
int mod_delivery_query(EXTENSION_CONTROL_BLOCK *pECB, int w_id,
int ld_id, char *ptr);
int mod_payment_query(EXTENSION_CONTROL_BLOCK *pECB, int w_id,
int ld_id, char *ptr);
int mod_orderstatus_query(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id, char *ptr);

```

```

int mod_stocklevel_query(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id, char *ptr);
int process_query(EXTENSION_CONTROL_BLOCK *);
int mod_begin_cmd(EXTENSION_CONTROL_BLOCK *);
int mod_menu_cmd(EXTENSION_CONTROL_BLOCK *, int, int);
int mod_exit_cmd(EXTENSION_CONTROL_BLOCK *);
int send_error_message(EXTENSION_CONTROL_BLOCK *, int, int, char
*, int, int, void *);

int get_wid_did(char *iptr, int *wid, int *did, char **optr);
int getcharvalue(char *iptr, char key, char **optr);
char *allocate_form(form_template_pool *pool, int *index);
char *allocate_form_new(form_template_pool *pool, int index);
void free_form(form_template_pool *pool, char *form_template,
int index);
void make_txn_form_template(char *, char *, char *, char *,
int);
int build_form_index(char *form, char *form_template,
form_index_entry *f_index, int length);
int send_response(EXTENSION_CONTROL_BLOCK *, char *, int);
void fill_number(char *form, int value, int index, int length);
void fill_float(char *form, double value, int index, int
length);
void fill_string(char *form, char *string, int index, int
length, int *shift);
void adjust_form(char *form, int *indexes, int *length, int
size, int formlen, int totalshift);
int get_number(char *ptr, int *value);
int parse_query_string(char *iptr, int max_cnt, char *txn_chars,
value_index_entry *txn_vals);

#define mod_neworder_cmd(rec, w_id, ld_id)
sendform_neworderinput(rec, w_id, ld_id)
#define mod_delivery_cmd(rec, w_id, ld_id)
sendform_deliveryinput(rec, w_id, ld_id)
#define mod_payment_cmd(rec, w_id, ld_id)
sendform_paymentinput(rec, w_id, ld_id)
#define mod_orderstatus_cmd(rec, w_id, ld_id)
sendform_orderstatusinput(rec, w_id, ld_id)
#define mod_stocklevel_cmd(rec, w_id, ld_id)
sendform_stocklevelinput(rec, w_id, ld_id)

/* -----
The following defines the form layout of the different
screens (forms).

NAME=1 - Command.

VALUE = NewOrder - neworder bring out new order input
form
Delivery - delivery bring out delivery input
form
OrderStatus - order status bring out order status
input form
Payment - payment bring out payment input
form
StockLevel - stock level bring out stock level
input form
Menu - display main menu
Process - perform the specified transaction
after providing input
Begin - send wid and did

NAME=2 - Form Type.

VALUE = d,n,p,s,o [D,N,P,S,O] output/input. Plus terminal
ID.
= W logon
= M main menu

Delivery
3 - district number.

Order Status
3 - district number.
4 - customer id.
5 - customer last name.

Payment
3 - district number.
4 - customer id.
5 - customer warehouse.
6 - customer district.
7 - name
8 - amount paid

Stock Level
3 - stock level threshold

New Order
3 - district number.
4 - customer number.
----- */

#define TRANSACTION_MENU \
"<HR>" \
"<INPUT TYPE=submit NAME=0 VALUE=NewOrder>" \
"<INPUT TYPE=submit NAME=0 VALUE=Payment>" \
"<INPUT TYPE=submit NAME=0 VALUE=Delivery>" \
"<INPUT TYPE=submit NAME=0 VALUE=StockLevel>" \
"<INPUT TYPE=submit NAME=0 VALUE=OrderStatus>" \
"<INPUT TYPE=submit NAME=0 VALUE=Exit>"

```

```

"<BODY><FORM ACTION=%s METHOD=GET>"
"<INPUT TYPE=hidden NAME=2 VALUE=B000>"
"%s. Please provide your warehouse ID and district ID.<BR>"
"Warehouse ID <INPUT NAME=3 SIZE=7><BR>"
"District ID <INPUT NAME=4 SIZE=2><BR>"
"<HR>"
"<INPUT TYPE=submit NAME=1 VALUE=Begin>"
"</FORM></BODY>"; */
static char WelcomeForm [] =
"<BODY><FORM ACTION=%s METHOD=GET>"
"<INPUT TYPE=hidden NAME=3 VALUE=W000>"
"%s. Please provide your warehouse ID and district ID.<BR>"
"Warehouse ID <INPUT NAME=4 SIZE=7><BR>"
"District ID <INPUT NAME=5 SIZE=2><BR>"
"<HR>"
"<INPUT TYPE=submit NAME=0 VALUE=Begin>"
"</FORM></BODY>";

static char FormHeader [] =
"<BODY><FORM ACTION=%s METHOD=GET>";

#define FORM_BEGIN "<BODY><FORM ACTION=%s METHOD=GET>"
#define FORM_END "</FORM></BODY>"
#define FORM_SUBMIT "<INPUT TYPE=submit NAME=0 VALUE=Process>"
#define FORM_MENU "<INPUT TYPE=submit NAME=0 VALUE=Menu>"

static char MainForm [] =
FORM_BEGIN
"<INPUT TYPE=hidden NAME=3 VALUE=M%07d>"
"%60s<BR>"
"Please Select the Next Transaction.<BR>"
TRANSACTION_MENU
FORM_END;

static char ErrorForm [] =
FORM_BEGIN
"<INPUT TYPE=hidden NAME=3 VALUE=e%06d>"
"Error: %d %d %40s %s<BR>"
TRANSACTION_MENU
FORM_END;

/*
static char ErrorForm [] =
FORM_BEGIN
"<INPUT TYPE=hidden NAME=3 VALUE=e%06d>"
"Error: %d (%s): %s<BR>"
TRANSACTION_MENU
FORM_END;
*/
#define DE_EXTRA_ID 0
#define DE_INPUT_DID DE_EXTRA_ID + 1
#define DE_INPUT_QTIME DE_INPUT_DID + 1
#define DE_INPUT_MAX DE_INPUT_QTIME + 1

#define OS_INPUT_DID 0
#define OS_INPUT_CID OS_INPUT_DID + 1
#define OS_INPUT_NAME OS_INPUT_CID + 1
#define OS_INPUT_MAX OS_INPUT_NAME + 1

#define PA_INPUT_DID 0
#define PA_INPUT_CID PA_INPUT_DID + 1
#define PA_INPUT_CWID PA_INPUT_CID + 1
#define PA_INPUT_CDID PA_INPUT_CWID + 1
#define PA_INPUT_NAME PA_INPUT_CDID + 1
#define PA_INPUT_AMT PA_INPUT_NAME + 1
#define PA_INPUT_MAX PA_INPUT_AMT + 1

#define SL_INPUT_THRESHOLD 0
#define SL_INPUT_MAX SL_INPUT_THRESHOLD + 1

#define NO_INPUT_DID 0
#define NO_INPUT_CID NO_INPUT_DID + 1
#define NO_INPUT_SPW00 NO_INPUT_CID + 1
#define NO_INPUT_ID00 NO_INPUT_SPW00 + 1
#define NO_INPUT_QTY00 NO_INPUT_ID00 + 1
#define NO_INPUT_SPW01 NO_INPUT_QTY00 + 1
#define NO_INPUT_ID01 NO_INPUT_SPW01 + 1
#define NO_INPUT_QTY01 NO_INPUT_ID01 + 1
#define NO_INPUT_SPW02 NO_INPUT_QTY01 + 1
#define NO_INPUT_ID02 NO_INPUT_SPW02 + 1
#define NO_INPUT_QTY02 NO_INPUT_ID02 + 1
#define NO_INPUT_SPW03 NO_INPUT_QTY02 + 1
#define NO_INPUT_ID03 NO_INPUT_SPW03 + 1
#define NO_INPUT_QTY03 NO_INPUT_ID03 + 1
#define NO_INPUT_SPW04 NO_INPUT_QTY03 + 1
#define NO_INPUT_ID04 NO_INPUT_SPW04 + 1
#define NO_INPUT_QTY04 NO_INPUT_ID04 + 1
#define NO_INPUT_SPW05 NO_INPUT_QTY04 + 1
#define NO_INPUT_ID05 NO_INPUT_SPW05 + 1
#define NO_INPUT_QTY05 NO_INPUT_ID05 + 1
#define NO_INPUT_SPW06 NO_INPUT_QTY05 + 1
#define NO_INPUT_ID06 NO_INPUT_SPW06 + 1
#define NO_INPUT_QTY06 NO_INPUT_ID06 + 1
#define NO_INPUT_SPW07 NO_INPUT_QTY06 + 1
#define NO_INPUT_ID07 NO_INPUT_SPW07 + 1
#define NO_INPUT_QTY07 NO_INPUT_ID07 + 1
#define NO_INPUT_SPW08 NO_INPUT_QTY07 + 1
#define NO_INPUT_ID08 NO_INPUT_SPW08 + 1
#define NO_INPUT_QTY08 NO_INPUT_ID08 + 1
#define NO_INPUT_SPW09 NO_INPUT_QTY08 + 1
#define NO_INPUT_ID09 NO_INPUT_SPW09 + 1
#define NO_INPUT_QTY09 NO_INPUT_ID09 + 1
#define NO_INPUT_SPW10 NO_INPUT_QTY09 + 1
#define NO_INPUT_ID10 NO_INPUT_SPW10 + 1
#define NO_INPUT_QTY10 NO_INPUT_ID10 + 1
#define NO_INPUT_SPW11 NO_INPUT_QTY10 + 1
#define NO_INPUT_ID11 NO_INPUT_SPW11 + 1

```



```

extern int (FAR * mod_tpcc_neworder)(T_neworder_data *);
extern int (FAR * mod_tpcc_payment)(T_payment_data *);
extern int (FAR * mod_tpcc_delivery)(T_delivery_data *, int);
extern int (FAR * mod_tpcc_orderstatus)(T_orderstatus_data *);
extern int (FAR * mod_tpcc_stocklevel)(T_stocklevel_data *);
extern void (FAR *userlog)(char * str, ...);
extern void (FAR *initDelLog)(int);
extern void (FAR *endDelLog)(int);

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

DWORD TlsPointer;
char DllPath[MAXLEN];
char LogFile[MAXLEN];
char InitFile[MAXLEN];
char DllFile[MAXLEN];
char origin[MAXLEN];
CRITICAL_SECTION critical_initDelQueue;
CRITICAL_SECTION critical_memory;
CRITICAL_SECTION critical_DelQueue_free;
CRITICAL_SECTION critical_DelQueue_work;
HANDLE waitAvailableDelQueue;
HANDLE waitDelWork;
HANDLE DelThreadRunning;
HINSTANCE dllinstance;
int seddel=0;
int DBConnections;
int Maxterms;
int DeliveryQueues;
int DeliveryThreads;
int modtpcc_ready=0;
int memory_ready=0;
int queue_ready=0;
int DeliveryThreadstop=0;
int StartTerm=1;
DelQueue_info *DelQueue_begin = NULL;
DelQueue_info *DelQueue_end = NULL;
DelQueue_info *DelQueue_free = NULL;

static form_index_entry
form_index[POOL_TYPE_TXN_MAX][TXN_TYPE_MAX][MAX_FORM_INDEX];
static form_template_pool
txn_global_pool[POOL_TYPE_TXN_MAX][TXN_TYPE_MAX];
static form_template_pool txn_data_pool;
static form_template_pool resp_global_pool;

char delivery_chars [] = {'6', '7'};
char orderstatus_chars [] = {'8', '9', 'Y'};
char payment_chars [] = {'8', '9', 'Z', 'v', 'Y', 'w'};
char stocklevel_chars [] = {'x'};
char neworder_chars [] = {'8', '9',
'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R',
'S', 'T', 'U', 'V', 'W', 'X', 'a', 'b', 'c',
'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l',
'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u'};

-----
paynz.sql
-----
DECLARE /* paynz */
not_serializable EXCEPTION;
PRAGMA EXCEPTION_INIT(not_serializable,-8177);
deadlock EXCEPTION;
PRAGMA EXCEPTION_INIT(deadlock,-60);
snapshot_too_old EXCEPTION;
PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
BEGIN
LOOP BEGIN
UPDATE ware
SET w_ytd = w_ytd + :h_amount
WHERE w_id = :w_id
RETURNING w_name, w_street_1, w_street_2, w_city,
w_state, w_zip
: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
:c_last, :c_street_1,
:c_street_2, :c_city, :c_state, :c_zip,
:c_phone,
:c_since, :c_credit, :c_credit_lim,
:c_discount, :c_balance;
IF SQL%NOTFOUND THEN
raise NO_DATA_FOUND;
END IF;

```

```

IF :c_credit = 'BC' THEN
UPDATE cust
SET c_data = substr ((to_char (:c_id) || ' ' ||
to_char (:c_d_id) || ' ' ||
to_char (:c_w_id) || ' ' ||
to_char (:d_id) || ' ' ||
to_char (:w_id) || ' ' ||
to_char (:h_amount/100,
'9999.99') || ' ' | ')
|| c_data, 1, 500)
WHERE rowid = inittpcc.cust_rowid
RETURNING substr(c_data,1, 200)
INTO :c_data;

END IF;

UPDATE dist
SET d_ytd = d_ytd + :h_amount
WHERE d_id = :d_id
AND d_w_id = :w_id
RETURNING d_name, d_street_1, d_street_2,
d_city,d_state, d_zip
INTO
inittpcc.dist_name,:d_street_1,:d_street_2,:d_city,:d_state,
:d_zip;
IF SQL%NOTFOUND THEN
raise NO_DATA_FOUND;
END IF;

INSERT INTO hist (h_c_id, h_c_d_id, h_c_w_id, h_d_id,
h_w_id,
h_amount, h_date, h_data)
VALUES
(:c_id, :c_d_id, :c_w_id, :d_id, :w_id, :h_amount,
:c_date, inittpcc.ware_name || ' ' ||
inittpcc.dist_name);
EXIT;

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

END LOOP;
END;

-----
payz.sql
-----
DECLARE /* payz */
not_serializable EXCEPTION;
PRAGMA EXCEPTION_INIT(not_serializable,-8177);
deadlock EXCEPTION;
PRAGMA EXCEPTION_INIT(deadlock,-60);
snapshot_too_old EXCEPTION;
PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
BEGIN
LOOP BEGIN
UPDATE ware
SET w_ytd = w_ytd+ :h_amount
WHERE w_id = :w_id
RETURNING w_name,
w_street_1, w_street_2, w_city, w_state,
w_zip
INTO inittpcc.ware_name,
:w_street_1, :w_street_2, :w_city, :w_state,
:w_zip;

SELECT rowid
BULK COLLECT INTO inittpcc.row_id
FROM cust
WHERE c_d_id = :c_d_id AND c_w_id = :c_w_id AND c_last
= :c_last
ORDER BY c_last, c_d_id, c_w_id, c_first;
inittpcc.c_num := sql%rowcount;
inittpcc.cust_rowid := inittpcc.row_id((inittpcc.c_num) /
2);

UPDATE cust
SET c_balance = c_balance - :h_amount,
c_ytd_payment = c_ytd_payment+ :h_amount,
c_payment_cnt = c_payment_cnt+1
WHERE rowid = inittpcc.cust_rowid
RETURNING
c_id, c_first, c_middle, c_last, c_street_1,
c_street_2,
c_city, c_state, c_zip, c_phone,
c_since, c_credit, c_credit_lim,
c_discount, c_balance
INTO :c_id, :c_first, :c_middle, :c_last,
:c_street_1, :c_street_2, :c_city, :c_state,
:c_zip, :c_phone, :c_since, :c_credit,
:c_credit_lim, :c_discount, :c_balance;

:c_data := ' ';
IF :c_credit = 'BC' THEN
UPDATE cust
SET c_data = substr ((to_char (:c_id) || ' ' ||
to_char (:c_d_id) || ' ' ||

```

```

        to_char (:c_w_id) || ' ' ||
        to_char (:d_id) || ' ' ||
        to_char (:w_id) || ' ' ||
        to_char (:h_amount/100,
'9999.99') || ' ' || ')
        || c_data, 1, 500)
        WHERE rowid = inittpcc.cust_rowid
        RETURNING substr(c_data,1, 200)
        INTO :c_data;

END IF;

UPDATE dist
SET d_ytd = d_ytd+h_amount
WHERE d_id = :d_id
AND d_w_id = :w_id
RETURNING d_name, d_street_1, d_street_2, d_city,
d_state, d_zip
INTO inittpcc.dist_name, :d_street_1, :d_street_2,
:d_city,
:d_state, :d_zip;

IF SQL%NOTFOUND
THEN
raise NO_DATA_FOUND;
END IF;

INSERT INTO hist (h_c_id, h_c_d_id, h_c_w_id, h_d_id,
h_w_id,
h_amount, h_date, h_data)
VALUES (:c_id, :c_d_id, :c_w_id, :d_id, :w_id,
:h_amount,
:cr_date, inittpcc.ware_name || ' ' ||
inittpcc.dist_name);

EXIT;

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

END LOOP;
END;

```

Stdafx.cpp

```

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

#include "stdafx.h"

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

```

Stdafx.h

```

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

#ifdef AFX_STDAFX_H__1D53560F_AAD5_4CEE_A8CC_651C9688A6DF__INC
LUDED_
#define
AFX_STDAFX_H__1D53560F_AAD5_4CEE_A8CC_651C9688A6DF__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

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

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

// TODO: reference additional headers your program requires here
//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.

#endif //
#ifdef AFX_STDAFX_H__1D53560F_AAD5_4CEE_A8CC_651C9688A6DF__INC
LUDED_

```

tkvcin.in.sql

-- The initnew package for storing variables used in the
-- New Order anonymous block

CREATE OR REPLACE PACKAGE inittpcc
AS

```

TYPE intarray IS TABLE OF INTEGER INDEX BY BINARY_INTEGER;
TYPE distarray IS TABLE OF VARCHAR(24) INDEX BY BINARY_INTEGER;
nulldate DATE;
TYPE rowidarray IS TABLE OF ROWID INDEX BY PLS_INTEGER;
s_dist distarray;
idxlarr intarray;
s_remote intarray;
dist intarray;
row_id rowidarray;
cust_rowid rowid;
dist_name VARCHAR2(11);
ware_name VARCHAR2(11);
c_num PLS_INTEGER;

```

```

PROCEDURE init_no(idxarr intarray);
PROCEDURE init_del;
PROCEDURE init_pay;
END inittpcc;
/
show errors;

```

CREATE OR REPLACE PACKAGE BODY inittpcc AS

```

PROCEDURE init_no (idxarr intarray)
IS
BEGIN
-- initialize null date
nulldate := TO_DATE('01-01-1811', 'MM-DD-YYYY');
idxlarr := idxarr;
END init_no;

```

```

PROCEDURE init_del
IS
BEGIN
FOR i IN 1 .. 10 LOOP
dist(i) := i;
END LOOP;
END init_del;

```

```

PROCEDURE init_pay IS
BEGIN
NULL;
END init_pay;

```

```

END inittpcc;
/
show errors
exit

```

tkvcpdel.sql

```

declare
TYPE numarray IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
TYPE numlist is varray (10) of number;
dist numarray;
amt numarray ;
cnt pls_integer;

not_serializable EXCEPTION;
PRAGMA EXCEPTION_INIT(not_serializable, -8177);
deadlock EXCEPTION;
PRAGMA EXCEPTION_INIT(deadlock, -60);
snapshot_too_old EXCEPTION;
PRAGMA EXCEPTION_INIT(snapshot_too_old, -1555);

BEGIN
LOOP BEGIN
FORALL d IN 1..10
DELETE FROM nord N
WHERE no_d_id = inittpcc.dist(d)
AND no_w_id = :w_id
AND no_o_id = (select min (no_o_id)
from nord
where no_d_id = N.no_d_id
and no_w_id = N.no_w_id)
RETURNING no_d_id, no_o_id BULK COLLECT INTO :d_id,
:order_id;

:ordcnt := SQL%ROWCOUNT;

FORALL o in 1.. :ordcnt
UPDATE ordr SET o_carrier_id = :carrier_id
WHERE o_id = :order_id (o)
AND o_d_id = :d_id(o)
AND o_w_id = :w_id
RETURNING o_c_id BULK COLLECT INTO :o_c_id;

FORALL o in 1.. :ordcnt
UPDATE ordl SET ol_delivery_d = :now
WHERE ol_w_id = :w_id
AND ol_d_id = :d_id(o)
AND ol_o_id = :order_id(o)
RETURNING sum(ol_amount) BULK COLLECT INTO :sums;

FORALL c IN 1.. :ordcnt
UPDATE cust
SET c_balance = c_balance + :sums(c),
c_delivery_cnt = c_delivery_cnt + 1
WHERE c_w_id = :w_id
AND c_d_id = :d_id(c)

```

```

        AND c_id = :o_c_id(c);
    COMMIT;
    EXIT;
    EXCEPTION
    WHEN not_serializable OR deadlock OR snapshot_too_old
    THEN
        ROLLBACK;
        :retry := :retry + 1;
    END;

END LOOP; -- for retry
END;

-----
tkvcpnew.sql
-----

-- New Order Anonymous block

DECLARE
    idx                PLS_INTEGER;
    dummy_local        PLS_INTEGER;
    cache_ol_cnt       PLS_INTEGER;
    not_serializable   EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_serializable,-8177);
    deadlock            EXCEPTION;
    PRAGMA EXCEPTION_INIT(deadlock,-60);
    snapshot_too_old   EXCEPTION;
    PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);

PROCEDURE u1 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
        UPDATE stock_item
        SET s_order_cnt = s_order_cnt + 1,
            s_ytd = s_ytd + :ol_quantity(idx),
            s_remote_cnt = s_remote_cnt + :s_remote(idx),
            s_quantity = (CASE WHEN s_quantity < :ol_quantity
(idx) + 10
                THEN s_quantity +91
                ELSE s_quantity
                END) - :ol_quantity(idx)
        WHERE i_id = :ol_i_id(idx)
        AND s_w_id = :ol_supply_w_id(idx)
        RETURNING i_price, i_name, s_quantity, s_dist_01,
            i_price*ol_quantity(idx),
            CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE 'B'
            END)
        END
        BULK COLLECT INTO :i_price, :i_name, :s_quantity,
inittpcc.s_dist,
            :ol_amount, :brand_generic;

    END u1;

PROCEDURE u2 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
        UPDATE stock_item
        SET s_order_cnt = s_order_cnt + 1,
            s_ytd = s_ytd + :ol_quantity(idx),
            s_remote_cnt = s_remote_cnt + :s_remote(idx),
            s_quantity = (CASE WHEN s_quantity < :ol_quantity
(idx) + 10
                THEN s_quantity +91
                ELSE s_quantity
                END) - :ol_quantity(idx)
        WHERE i_id = :ol_i_id(idx)
        AND s_w_id = :ol_supply_w_id(idx)
        RETURNING i_price, i_name, s_quantity, s_dist_02,
            i_price*ol_quantity(idx),
            CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE 'B'
            END)
        END
        BULK COLLECT INTO :i_price, :i_name, :s_quantity,
inittpcc.s_dist,
            :ol_amount, :brand_generic;

    END u2;

PROCEDURE u3 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
        UPDATE stock_item
        SET s_order_cnt = s_order_cnt + 1,
            s_ytd = s_ytd + :ol_quantity(idx),
            s_remote_cnt = s_remote_cnt + :s_remote(idx),
            s_quantity = (CASE WHEN s_quantity < :ol_quantity
(idx) + 10
                THEN s_quantity +91
                ELSE s_quantity
                END) - :ol_quantity(idx)
        WHERE i_id = :ol_i_id(idx)
        AND s_w_id = :ol_supply_w_id(idx)
        RETURNING i_price, i_name, s_quantity, s_dist_03,
            i_price*ol_quantity(idx),
            CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
            THEN 'G'

```

```

        ELSE 'B'
        END)
    END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity,
inittpcc.s_dist,
            :ol_amount, :brand_generic;

    END u3;

PROCEDURE u4 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
        UPDATE stock_item
        SET s_order_cnt = s_order_cnt + 1,
            s_ytd = s_ytd + :ol_quantity(idx),
            s_remote_cnt = s_remote_cnt + :s_remote(idx),
            s_quantity = (CASE WHEN s_quantity < :ol_quantity
(idx) + 10
                THEN s_quantity +91
                ELSE s_quantity
                END) - :ol_quantity(idx)
        WHERE i_id = :ol_i_id(idx)
        AND s_w_id = :ol_supply_w_id(idx)
        RETURNING i_price, i_name, s_quantity, s_dist_04,
            i_price*ol_quantity(idx),
            CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE 'B'
            END)
        END
        BULK COLLECT INTO :i_price, :i_name, :s_quantity,
inittpcc.s_dist,
            :ol_amount, :brand_generic;

    END u4;

PROCEDURE u5 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
        UPDATE stock_item
        SET s_order_cnt = s_order_cnt + 1,
            s_ytd = s_ytd + :ol_quantity(idx),
            s_remote_cnt = s_remote_cnt + :s_remote(idx),
            s_quantity = (CASE WHEN s_quantity < :ol_quantity
(idx) + 10
                THEN s_quantity +91
                ELSE s_quantity
                END) - :ol_quantity(idx)
        WHERE i_id = :ol_i_id(idx)
        AND s_w_id = :ol_supply_w_id(idx)
        RETURNING i_price, i_name, s_quantity, s_dist_05,
            i_price*ol_quantity(idx),
            CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE 'B'
            END)
        END
        BULK COLLECT INTO :i_price, :i_name, :s_quantity,
inittpcc.s_dist,
            :ol_amount, :brand_generic;

    END u5;

PROCEDURE u6 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
        UPDATE stock_item
        SET s_order_cnt = s_order_cnt + 1,
            s_ytd = s_ytd + :ol_quantity(idx),
            s_remote_cnt = s_remote_cnt + :s_remote(idx),
            s_quantity = (CASE WHEN s_quantity < :ol_quantity
(idx) + 10
                THEN s_quantity +91
                ELSE s_quantity
                END) - :ol_quantity(idx)
        WHERE i_id = :ol_i_id(idx)
        AND s_w_id = :ol_supply_w_id(idx)
        RETURNING i_price, i_name, s_quantity, s_dist_06,
            i_price*ol_quantity(idx),
            CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE 'B'
            END)
        END
        BULK COLLECT INTO :i_price, :i_name, :s_quantity,
inittpcc.s_dist,
            :ol_amount, :brand_generic;

    END u6;

PROCEDURE u7 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
        UPDATE stock_item
        SET s_order_cnt = s_order_cnt + 1,
            s_ytd = s_ytd + :ol_quantity(idx),
            s_remote_cnt = s_remote_cnt + :s_remote(idx),
            s_quantity = (CASE WHEN s_quantity < :ol_quantity
(idx) + 10
                THEN s_quantity +91
                ELSE s_quantity
                END) - :ol_quantity(idx)
        WHERE i_id = :ol_i_id(idx)
        AND s_w_id = :ol_supply_w_id(idx)
        RETURNING i_price, i_name, s_quantity, s_dist_07,
            i_price*ol_quantity(idx),

```

```

CASE WHEN i_data NOT LIKE '%ORIGINAL%'
THEN 'G'
ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
THEN 'G'
ELSE 'B'
END)
END
BULK COLLECT INTO :i_price, :i_name, :s_quantity,
inittppc.s_dist,
:ol_amount, :brand_generic;
END u7;
PROCEDURE u8 IS
BEGIN
FORALL idx IN 1 .. cache_ol_cnt
UPDATE stock_item
SET s_order_cnt = s_order_cnt + 1,
s_ytd = s_ytd + :ol_quantity(idx),
s_remote_cnt = s_remote_cnt + :s_remote(idx),
s_quantity = (CASE WHEN s_quantity < :ol_quantity
(idx) + 10
THEN s_quantity + 91
ELSE s_quantity
END) - :ol_quantity(idx)
WHERE i_id = :ol_i_id(idx)
AND s_w_id = :ol_supply_w_id(idx)
RETURNING i_price, i_name, s_quantity, s_dist_08,
i_price*ol_quantity(idx),
CASE WHEN i_data NOT LIKE '%ORIGINAL%'
THEN 'G'
ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
THEN 'G'
ELSE 'B'
END)
END)
BULK COLLECT INTO :i_price, :i_name, :s_quantity,
inittppc.s_dist,
:ol_amount, :brand_generic;
END u8;
PROCEDURE u9 IS
BEGIN
FORALL idx IN 1 .. cache_ol_cnt
UPDATE stock_item
SET s_order_cnt = s_order_cnt + 1,
s_ytd = s_ytd + :ol_quantity(idx),
s_remote_cnt = s_remote_cnt + :s_remote(idx),
s_quantity = (CASE WHEN s_quantity < :ol_quantity
(idx) + 10
THEN s_quantity + 91
ELSE s_quantity
END) - :ol_quantity(idx)
WHERE i_id = :ol_i_id(idx)
AND s_w_id = :ol_supply_w_id(idx)
RETURNING i_price, i_name, s_quantity, s_dist_09,
i_price*ol_quantity(idx),
CASE WHEN i_data NOT LIKE '%ORIGINAL%'
THEN 'G'
ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
THEN 'G'
ELSE 'B'
END)
END)
BULK COLLECT INTO :i_price, :i_name, :s_quantity,
inittppc.s_dist,
:ol_amount, :brand_generic;
END u9;
PROCEDURE u10 IS
BEGIN
FORALL idx IN 1 .. cache_ol_cnt
UPDATE stock_item
SET s_order_cnt = s_order_cnt + 1,
s_ytd = s_ytd + :ol_quantity(idx),
s_remote_cnt = s_remote_cnt + :s_remote(idx),
s_quantity = (CASE WHEN s_quantity < :ol_quantity
(idx) + 10
THEN s_quantity + 91
ELSE s_quantity
END) - :ol_quantity(idx)
WHERE i_id = :ol_i_id(idx)
AND s_w_id = :ol_supply_w_id(idx)
RETURNING i_price, i_name, s_quantity, s_dist_10,
i_price*ol_quantity(idx),
CASE WHEN i_data NOT LIKE '%ORIGINAL%'
THEN 'G'
ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
THEN 'G'
ELSE 'B'
END)
END)
BULK COLLECT INTO :i_price, :i_name, :s_quantity,
inittppc.s_dist,
:ol_amount, :brand_generic;
END u10;
PROCEDURE fix_items IS
rows_lost PLS_INTEGER;
max_index PLS_INTEGER;
temp_index PLS_INTEGER;
BEGIN
idx := 1;
rows_lost := 0;
max_index := dummy_local;
WHILE (max_index != cache_ol_cnt) LOOP
WHILE (idx <= sql%rowcount AND

```

```

sql%bulk_rowcount(idx + rows_lost) = 1)
LOOP
idx := idx + 1;
END LOOP;
temp_index := max_index;
WHILE (temp_index >= idx + rows_lost) LOOP
:ol_amount(temp_index + 1) :=
:ol_amount(temp_index);
:i_price(temp_index + 1) :=
:i_price(temp_index);
:i_name(temp_index + 1) :=
:i_name(temp_index);
:s_quantity(temp_index + 1) :=
:s_quantity(temp_index);
inittppc.s_dist(temp_index + 1) :=
inittppc.s_dist(temp_index);
:brand_generic(temp_index + 1) :=
:brand_generic(temp_index);
temp_index := temp_index - 1;
END LOOP;
IF (idx + rows_lost <= cache_ol_cnt) THEN
:i_price(idx + rows_lost) := 0;
:i_name(idx + rows_lost) := 'NO ITEM';
:s_quantity(idx + rows_lost) := 0;
inittppc.s_dist(idx + rows_lost) := NULL;
:brand_generic(idx + rows_lost) := ' ';
:ol_amount(idx + rows_lost) := 0;
rows_lost := rows_lost + 1;
max_index := max_index + 1;
END IF;
END LOOP;
END fix_items;
BEGIN
LOOP BEGIN
cache_ol_cnt := :o_ol_cnt;
UPDATE dist SET d_next_o_id = d_next_o_id + 1
WHERE d_id = :d_id AND d_w_id = :w_id
RETURNING d_tax, d_next_o_id-1
INTO :d_tax, :o_id;
SELECT c_discount, c_last, c_credit, 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 ordx (o_id,o_d_id, o_w_id, o_c_id,
o_entry_d, o_carrier_id, o_ol_cnt,
o_all_local)
VALUES (:o_id, :d_id, :w_id, :c_id,
:cr_date, 11, :o_ol_cnt, :o_all_local);
dummy_local := :d_id;
IF (dummy_local < 6) THEN
IF (dummy_local < 3) THEN
IF (dummy_local = 1) THEN
u1;
ELSE
u2;
END IF;
ELSE
IF (dummy_local = 3) THEN
u3;
ELSIF (dummy_local = 4) then
u4;
ELSE
u5;
END IF;
END IF;
ELSE
IF (dummy_local < 8) THEN
IF (dummy_local = 6) THEN
u6;
ELSE
u7;
END IF;
ELSE
IF (dummy_local = 8) THEN
u8;
ELSIF (dummy_local = 9) then
u9;
ELSE
u10;
END IF;
END IF;
END IF;
dummy_local := sql%rowcount;
IF (dummy_local != cache_ol_cnt ) THEN fix_items; END
IF;
FORALL idx IN 1..dummy_local
INSERT INTO ordl

```

```

        (ol_o_id, ol_d_id, ol_w_id, ol_number,
ol_delivery_d, ol_i_id,
        ol_supply_w_id,
ol_quantity, ol_amount, ol_dist_info)
VALUES (:o_id, :d_id, :w_id, inittpcclarr(idx),
inittpcclnulldate,
        :ol_i_id(idx), :ol_supply_w_id(idx),
        :ol_quantity(idx), :ol_amount(idx),
inittpccl_s_dist(idx));

        IF (dummy_local != :o_ol_cnt) THEN
        :o_ol_cnt := dummy_local;
        ROLLBACK;
        END IF;

EXIT;

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

```

```

tpccflags.h
-----
//#define USE_IEEE_NUMBER
-----

tpccpl.h
-----
#ifndef TPCCPL_H
#define TPCCPL_H

//#include "tpcc.h"

#include <oratypes.h>
#include <oci.h>
#include <ocidfn.h>
#include <time.h>
#include <io.h>
#include "tpccflags.h"

#ifndef TUX
#define DELRT 5.0
#else
#define DELRT 80.0
#endif

#ifndef DISCARD
#define DISCARD (void)
#endif

#ifndef sword
#define sword int
#endif

#define VER7 2

#define NA -1 /* ANSI SQL NULL */
#define NLT 1 /* length for string null
terminator */
#define DEADLOCK 60 /* ORA-00060: deadlock */
#define NO_DATA_FOUND 1403 /* ORA-01403: no data found */
#define NOT_SERIALIZABLE 8177 /* ORA-08177: transaction not
serializable */
#define SNAPSHOT_TOO_OLD 1555 /* ORA-01555: snapshot too old
*/

/* Error codes */

#define RECOVER -10
#define IRRECERR -20
#define NOERR 111
#define DEL_ERROR -666
#define DEL_DATE_LEN 7
#define NDISTS 10
#define NITEMS 15
#define SQL_BUF_SIZE 8192

#define FULLDATE "dd-mon-yy.hh24:mi:ss"
#define SHORTDATE "dd-mm-yyyy"

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

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

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

#define OCIERROR(errp, function) \
ocierror(__FILE__, __LINE__, (errp), (function));

#define OCIBND(stmp, bndp, errp, sqlvar, progvl, ftype) \

```

```

ocierror(__FILE__, __LINE__, (errp), \
OCIHandleAlloc((stmp), (dvoid **)&(bndp), OCI_HTYPE_BIND, 0, (dvoid**
0)); \
ocierror(__FILE__, __LINE__, (errp), \
OCIBindByName((stmp), &(bndp), (errp), \
(text *) (sqlvar), strlen((sqlvar)), \
(progvl), (progvl), (ftype), 0, 0, 0, 0, OCI_DEFAULT));

#define
OCIBNDRA(stmp, bndp, errp, sqlvar, progvl, progvl, ftype, indp, alen, arco
de) \
ocierror(__FILE__, __LINE__, (errp), \
OCIHandleAlloc((stmp), (dvoid **)&(bndp), OCI_HTYPE_BIND, 0, (dvoid**
0)); \
ocierror(__FILE__, __LINE__, (errp), \
OCIBindByName((stmp), &(bndp), (errp), (text
*)(sqlvar), strlen((sqlvar)), \
(progvl), (progvl), (ftype), (indp), (alen), (arcode), 0, 0, OCI_DEFAULT
));
#define
OCIBNDRAD(stmp, bndp, errp, sqlvar, progvl, ftype, indp, ctp, cbf_nodata
a, cbf_data) \
ocierror(__FILE__, __LINE__, (errp), \
OCIHandleAlloc((stmp), (dvoid **)&(bndp), OCI_HTYPE_BIND, 0, (dvoid**
0)); \
ocierror(__FILE__, __LINE__, (errp), \
OCIBindByName((stmp), &(bndp), (errp), (text
*)(sqlvar), strlen((sqlvar)), \
indp, 0, 0, 0, OCI_DATA_AT_EXEC)); \
ocierror(__FILE__, __LINE__, (errp), \
OCIBindDynamic((bndp), (errp), (ctp), (cbf_nodata), (ctp), (cbf_dat
a)));

#define
OCIBNDR(stmp, bndp, errp, sqlvar, progvl, progvl, ftype, indp, alen, arco
de) \
ocierror(__FILE__, __LINE__, (errp), \
OCIHandleAlloc((stmp), (dvoid **)&(bndp), OCI_HTYPE_BIND, 0, (dvoid**
0)); \
ocierror(__FILE__, __LINE__, (errp), \
OCIBindByName((stmp), &(bndp), (errp), (text
*)(sqlvar), strlen((sqlvar)), \
(progvl), (progvl), (ftype), (indp), (alen), (arcode), 0, 0, OCI_DEFAULT
));
#define
OCIBNDRAA(stmp, bndp, errp, sqlvar, progvl, progvl, ftype, indp, alen, arco
de, ms, cu) \
ocierror(__FILE__, __LINE__, (errp), \
OCIHandleAlloc((stmp), &(bndp), OCI_HTYPE_BIND, 0, (dvoid**)0)); \
ocierror(__FILE__, __LINE__, (errp), \
OCIBindByName((stmp), &(bndp), (errp), (text
*)(sqlvar), strlen((sqlvar)), \
(progvl), (progvl), (ftype), (indp), (alen), (arcode), (ms), (cu), OCI_DE
FAULT));

#define OCIDEFINE(stmp, dfnp, errp, pos, progvl, progvl, ftype) \
OCIDefineByPos((stmp), &(dfnp), (errp), (pos), (progvl), (progvl), (ftyp
e), \
0, 0, 0, OCI_DEFAULT);

#define OCIDEF(stmp, dfnp, errp, pos, progvl, progvl, ftype) \
OCIHandleAlloc((stmp), (dvoid **)&(dfnp), OCI_HTYPE_DEFINE, 0, \
(dvoid**)0); \
OCIDefineByPos((stmp), &(dfnp), (errp), (pos), (progvl), (progvl), \
(ftype), NULL, NULL, NULL, OCI_DEFAULT); \

#define
OCIDFNRA(stmp, dfnp, errp, pos, progvl, progvl, ftype, indp, alen, arco
de) \
OCIHandleAlloc((stmp), (dvoid **)&(dfnp), OCI_HTYPE_DEFINE, 0, \
(dvoid**)0); \
OCIDefineByPos((stmp), &(dfnp), (errp), (pos), (progvl), \
(progvl), (ftype), (indp), (alen), \
(arcode), OCI_DEFAULT); \

#define
OBNDRV(lda, cursor, sqlvar, progvl, progvl, ftype) \
if
(obndrv((cursor), (text *) (sqlvar), NA, (ub1 *) (progvl), (progvl), (ftyp
e), NA, \
(sb2 *) 0, (text *) 0, NA, NA)) \
{errrpt(lda, cursor); return(-1);} \
else \
DISCARD 0

#define
OBNDRA(lda, cursor, sqlvar, progvl, progvl, ftype, indp, alen, arco
de) \
if
(obndra((cursor), (text *) (sqlvar), NA, (ub1 *) (progvl), (progvl), (ftyp
e), NA, \
(indp), (alen), (arcode), (ub4 *) 0, (ub4 *) 0, (text *) 0, NA, NA)) \
{errrpt(lda, cursor); return(-1);} \

```

```

else\
    DISCARD 0
#define
OBNDRAA(lda, cursor, sqlvar, progvl, progvl, ftype, indp, alen, arcode, ms, cs)\
    if
    (obndra((cursor), (text*)(sqlvar), NA, (ub1*)(progvl), (progvl), (ftype), NA, \
    (indp), (alen), (arcod), (ub4)(ms), (ub4*)(cs), (text*)0, NA, NA))\
        {errrpt(lda, cursor); return(-1);} \
    else\
        DISCARD 0
#define
ODEFIN(lda, cursor, pos, buf, buf1, ftype, scale, indp, fmt, fmt1, ffmt, r1, en, rcode)\
    if
    (odefin((cursor), (pos), (ub1*)(buf), (buf1), (ftype), (scale), (indp), \
    (text*)(fmt), (fmt1), (ffmt), (rlen), (rcode)))\
        {errrpt(lda, cursor); return(-1);} \
    else\
        DISCARD 0
#define
OEXPET(lda, cursor, nrows, cancel, exact)\
    if (oexpet((cursor), (nrows), (cancel), (exact)))\
        {if ((cursor)->rc == 1403) \
        {i=errrpt(lda, cursor); orol(lda); return(-1);} \
        else if (errrpt(lda, cursor)==RECOVER) \
        {orol(lda); return(RECOVER);} \
        else{orol(lda); return(-1);}} \
    else\
        DISCARD 0
#define
OOPEN(lda, cursor)\
    if (oopen((cursor), (lda), (text*)0, NA, NA, (text*)0, NA))\
        {errrpt(lda, cursor); return(-1);} \
    else\
        DISCARD 0
#define
OPARSE(lda, cursor, sqlstm, sql1, defflg, lngflg)\
    if
    (oparse((cursor), (sqlstm), (sb4)(sql1), (defflg), (ub4)(lngflg)))\
        {errrpt(lda, cursor); return(-1);} \
    else\
        DISCARD 0
#define
OFEN(lda, cursor, nrows)\
    if (ofen((cursor), (nrows)))\
        {if (errrpt(lda, cursor)==RECOVER) \
        {orol(lda); return(RECOVER);} \
        else{orol(lda); return(-1);}} \
    else\
        DISCARD 0
#define
OEXEC(lda, cursor)\
    if (oexec((cursor))\
        {if (errrpt(lda, cursor)==RECOVER) \
        {orol(lda); return(RECOVER);} \
        else{orol(lda); return(-1);}} \
    else\
        DISCARD 0
#define
OCOM(lda, cursor)\
    if (ocom((lda)) \
        {errrpt(lda, cursor); orol(lda); return(-1);} \
    else\
        DISCARD 0
#define
OEXN(lda, cursor, iters, rowoff)\
    if (oexn((cursor), (iters), (rowoff)) \
        {if (errrpt(lda, cursor)==RECOVER) \
        {orol(lda); return(RECOVER);} \
        else{orol(lda); return(-1);}} \
    else\
        DISCARD 0
/* bind in/out for plsql without indicator and rcode */
#define
OCIBNDPL(stmp, bndp, errp, sqlvar, progvl, progvl, ftype, alen)\
    \
    DISCARD ocierror(__FILE__, __LINE__, (errp), \
OCIHandleAlloc((stmp), (dvoid*)&(bndp), OCI_HTYPE_BIND, 0, (dvoid**\
)0); \
    DISCARD ocierror(__FILE__, __LINE__, (errp), \
OCIBindByName((stmp), &(bndp), (errp), (const text
*)(sqlvar), \
        (sb4)strlen((const char*)(sqlvar)),
(dvoid*)(progvl), (progvl), (ftype), \
        NULLP(dvoid), (alen), NULLP(ub2),
0, NULLP(ub4), OCI_DEFAULT));
/* bind in/out for plsql arrays without indicator and rcode */
#define
OCIBNDPLA(stmp, bndp, errp, sqlvar, progvl, progvl, ftype, alen, ms, cu) \
    DISCARD ocierror(__FILE__, __LINE__, (errp), \
OCIHandleAlloc((stmp), (dvoid*)&(bndp), OCI_HTYPE_BIND, 0, (dvoid**\
)0)); \
    DISCARD ocierror(__FILE__, __LINE__, (errp), \
OCIBindByName((stmp), &(bndp), (errp), (CONST text
*)(sqlvar), \

```

```

        (sb4)strlen((CONST char*)(sqlvar)), (void
*)(progvl), \
        (progvl), (ftype), NULL, (alen), NULL, (ms), (cu), OCI_DEFAULT));
#define
OCIDEFINE(stmp, dfnp, errp, pos, progvl, progvl, ftype)\
OCIDefineByPos((stmp), &(dfnp), (errp), (pos), (progvl), (progvl), (ftyp
e), \
        0, 0, 0, OCI_DEFAULT);
#define
OCIDEF(stmp, dfnp, errp, pos, progvl, progvl, ftype) \
OCIHandleAlloc((stmp), (dvoid*)&(dfnp), OCI_HTYPE_DEFINE, 0, \
(dvoid**)0); \
OCIDefineByPos((stmp), &(dfnp), (errp), (pos), (progvl), (progvl), \
(ftype), NULL, NULL, NULL, OCI_DEFAULT); \
#define
OCIDFNRA(stmp, dfnp, errp, pos, progvl, progvl, ftype, indp, alen, arcode)\
OCIHandleAlloc((stmp), (dvoid*)&(dfnp), OCI_HTYPE_DEFINE, 0, \
(dvoid**)0); \
OCIDefineByPos((stmp), &(dfnp), (errp), (pos), (progvl), \
(progvl), (ftype), (indp), (alen), \
(arcod), OCI_DEFAULT);
#define
OCIDFNDYN(stmp, dfnp, errp, pos, progvl, progvl, 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), (progvl),
(progvl), (ftype), \
(indp), NULL, NULL,
OCI_DYNAMIC_FETCH)); \
ocierror(__FILE__, __LINE__, (errp), \
OCIDefineDynamic((dfnp), (errp), (ctp), (cbf_data)));
#endif
-----
tpcc_struct.h
-----
/* Copyright (c) 2004, Oracle Corporation. All rights reserved.
*/
/*
NAME
    tpcc_struct.h - <one-line expansion of the name>
DESCRIPTION
    <short description of facility this file declares/defines>
RELATED DOCUMENTS
    <note any documents related to this facility>
EXPORT FUNCTION(S)
    <external functions declared for use outside package - one-
line descriptions>
INTERNAL FUNCTION(S)
    <other external functions declared - one-line descriptions>
EXAMPLES
NOTES
    <other useful comments, qualifications, etc.>
MODIFIED    (MM/DD/YY)
xnie        02/09/04 - add status field to carry error status
shuang      01/22/04 - shuang_rte
shuang      01/21/04 - Creation
*/
#define
MAX_ORDERLINE 15
#define
SMALL_BUF_SIZE 32
#define
TXN_COMMON_DATA \
    int w_id; \
    int ld_id; \
    int txn_status; \
    int db_status; \
    void *context
struct
T_connect_data
{
    TXN_COMMON_DATA;
};
typedef
struct
T_connect_data
T_connect_data;
struct
T_date
{
    char DateString[20];
};

```

```

typedef struct T_date T_date;

struct T_delivery_data
{
    TXN_COMMON_DATA;
    double    enqueue_time;
    double    dequeue_time;
    double    complete_time;
    int       o_carrier_id;
    int       o_id[10];
};
typedef struct T_delivery_data T_delivery_data;

struct T_orderline
{
    int       ol_i_id;
    int       ol_supply_w_id;
    int       ol_quantity;
    char      i_name[25];
    int       s_quantity;
    char      b_g[2];
    double    i_price;
    double    ol_amount;
};
typedef struct T_orderline T_orderline;

struct T_neworder_data
{
    TXN_COMMON_DATA;
    int       d_id;
    int       c_id;
    int       o_ol_cnt;
    int       o_all_local;
    T_orderline o_orderline[MAX_ORDERLINE];
    T_date    o_entry_d;
    char      c_last[17];
    char      c_credit[3];
    double    c_discount;
    double    w_tax;
    double    d_tax;
    int       o_id;
    double    total_amount;
    int       status;
};
typedef struct T_neworder_data T_neworder_data;

struct T_stocklevel_data
{
    TXN_COMMON_DATA;
    int       threshold;
    int       low_stock;
};
typedef struct T_stocklevel_data T_stocklevel_data;

struct T_orderline_status
{
    int       ol_supply_w_id;
    int       ol_i_id;
    int       ol_quantity;
    double    ol_amount;
    T_date    ol_delivery_d;
};
typedef struct T_orderline_status T_orderline_status;

struct T_orderstatus_data
{
    TXN_COMMON_DATA;
    int       by_last_name;
    int       d_id;
    int       c_id;
    char      c_last[17];
    char      c_first[17];
    char      c_middle[3];
    double    c_balance;
    int       o_id;
    T_date    o_entry_d;
    int       o_carrier_id;
    int       o_ol_cnt;
    T_orderline_status o_orderline[MAX_ORDERLINE];
};
typedef struct T_orderstatus_data T_orderstatus_data;

struct T_payment_data
{
    TXN_COMMON_DATA;
    int       by_last_name;
    int       d_id;
    int       c_id;
    char      c_last[17];
    int       c_w_id;
    int       c_d_id;
    double    h_amount;
    T_date    h_date;
    char      w_street_1[21];
    char      w_street_2[21];
    char      w_city[21];
    char      w_state[3];
    char      w_zip[10];
    char      d_street_1[21];
    char      d_street_2[21];
    char      d_city[21];
    char      d_state[3];
    char      d_zip[10];
    char      c_first[17];
    char      c_middle[3];
    char      c_street_1[21];
    char      c_street_2[21];
    char      c_city[21];

```

```

    char      c_state[3];
    char      c_zip[10];
    char      c_phone[17];
    T_date    c_since;
    char      c_credit[3];
    double    c_credit_lim;
    double    c_discount;
    double    c_balance;
    char      c_data[201];
};
typedef struct T_payment_data T_payment_data;

struct T_transaction_data
{
    int       txn_type;
    union {
        T_delivery_data delivery_data;
        T_payment_data payment_data;
        T_neworder_data neworder_data;
        T_stocklevel_data stocklevel_data;
        T_orderstatus_data orderstatus_data;
    } txn_data;
};
typedef struct T_transaction_data T_transaction_data;

struct T_login_data
{
    TXN_COMMON_DATA;
    char      server[SMALL_BUF_SIZE];
    char      database[SMALL_BUF_SIZE];
    char      user[SMALL_BUF_SIZE];
    char      password[SMALL_BUF_SIZE];
    char      application[SMALL_BUF_SIZE];
};
typedef struct T_login_data T_login_data;

-----
tpccstruct.h
-----

#define NITEMS 15
#define ROWIDLEN 20
#define OCIROWLEN 20

struct newctx {

    ub2 nol_i_id_len[NITEMS];
    ub2 nol_supply_w_id_len[NITEMS];
    ub2 nol_quantity_len[NITEMS];
    ub2 nol_amount_len[NITEMS];
    ub2 s_quantity_len[NITEMS];
    ub2 i_name_len[NITEMS];
    ub2 i_price_len[NITEMS];
    ub2 s_dist_info_len[NITEMS];
    ub2 ol_o_id_len[NITEMS];
    ub2 ol_number_len[NITEMS];
    ub2 s_remote_len[NITEMS];
    ub2 s_quant_len[NITEMS];
    ub2 ol_dist_info_len[NITEMS];
    ub2 s_bg_len[NITEMS];

    int ol_o_id[NITEMS];
    int ol_number[NITEMS];

#ifdef USE_IEEE_NUMBER
    float s_remote[NITEMS];
#else
    int s_remote[NITEMS];
#endif
    char s_dist_info[NITEMS][25];
    OCISmt *curn1;
    OCIBind *ol_i_id_bp;
    OCIBind *ol_supply_w_id_bp;
    OCIBind *i_price_bp;
    OCIBind *i_name_bp;
    OCIBind *s_bg_bp;
    ub4 nol_i_count;
    ub4 nol_s_count;
    ub4 nol_q_count;
    ub4 nol_item_count;
    ub4 nol_name_count;
    ub4 nol_qty_count;
    ub4 nol_bg_count;
    ub4 nol_am_count;
    ub4 s_remote_count;
    OCISmt *curn2;
    OCIBind *ol_quantity_bp;
    OCIBind *s_remote_bp;
    OCIBind *s_quantity_bp;
    OCIBind *w_id_bp;
    OCIBind *d_id_bp;
    OCIBind *c_id_bp;
    OCIBind *o_all_local_bp;
    OCIBind *o_all_cnt_bp;
    OCIBind *w_tax_bp;
    OCIBind *d_tax_bp;
    OCIBind *o_id_bp;
    OCIBind *c_discount_bp;
    OCIBind *c_credit_bp;
    OCIBind *c_last_bp;

```

```

OCIBind *retries_bp;
OCIBind *cr_date_bp;
OCIBind *ol_o_id_bp;
OCIBind *ol_amount_bp;

ub2 w_id_len;
ub2 d_id_len;
ub2 c_id_len;
ub2 o_all_local_len;
ub2 o_ol_cnt_len;
ub2 w_tax_len;
ub2 d_tax_len;
ub2 o_id_len;
ub2 c_discount_len;
ub2 c_credit_len;
ub2 c_last_len;
ub2 retries_len;
ub2 cr_date_len;
};

typedef struct newctx newctx;

#define NDISTS 10
#define ROWIDLEN 20

struct delctx {
    sb2 del_o_id_ind[NDISTS];
    sb2 d_id_ind[NDISTS];
    sb2 c_id_ind[NDISTS];
    sb2 del_date_ind[NDISTS];
    sb2 carrier_id_ind[NDISTS];
    sb2 amt_ind[NDISTS];

    ub4 del_o_id_len[NDISTS];
    ub4 c_id_len[NDISTS];
    int oid_ctx;
    int cid_ctx;
    OCIBind *olamt_bp;

    ub2 w_id_len[NDISTS];
    ub2 d_id_len[NDISTS];
    ub2 del_date_len[NDISTS];
    ub2 carrier_id_len[NDISTS];
    ub2 amt_len[NDISTS];

    ub2 del_o_id_rcode[NDISTS];
    ub2 cons_rcode[NDISTS];
    ub2 w_id_rcode[NDISTS];
    ub2 d_id_rcode[NDISTS];
    ub2 c_id_rcode[NDISTS];
    ub2 del_date_rcode[NDISTS];
    ub2 carrier_id_rcode[NDISTS];
    ub2 amt_rcode[NDISTS];

    int del_o_id[NDISTS];
    int del_d_id[NDISTS];
    int cons[NDISTS];
    int w_id[NDISTS];
    int d_id[NDISTS];
    int c_id[NDISTS];
    int carrier_id[NDISTS];
    int amt[NDISTS];
    ub4 del_o_id_rcnt;
    int retry;
    OCIRowid *no_rowid_ptr[NDISTS];
    OCIRowid *o_rowid_ptr[NDISTS];
    OCIDate del_date[NDISTS];
    OCISTmt *curd0;
    OCISTmt *curd1;
    OCISTmt *curd2;
    OCISTmt *curd3;
    OCISTmt *curd4;
    OCISTmt *curd5;
    OCISTmt *curd6;
    OCISTmt *curdtest;

    OCIBind *w_id_bp;
    OCIBind *w_id_bp3;
    OCIBind *w_id_bp4;
    OCIBind *w_id_bp5;
    OCIBind *w_id_bp6;
    OCIBind *d_id_bp;
    OCIBind *d_id_bp3;
    OCIBind *d_id_bp4;
    OCIBind *d_id_bp6;
    OCIBind *o_id_bp;
    OCIBind *cr_date_bp;
    OCIBind *c_id_bp;
    OCIBind *c_id_bp3;
    OCIBind *no_rowid_bp;
    OCIBind *carrier_id_bp;
    OCIBind *o_rowid_bp;
    OCIBind *del_o_id_bp;
    OCIBind *del_o_id_bp3;
    OCIBind *amt_bp;
    OCIBind *bstr1_bp[10];
    OCIBind *bstr2_bp[10];
    OCIBind *retry_bp;
    OCIDefine *inum_dp;
    OCIDefine *d_id_dp;
    OCIDefine *del_o_id_dp;
    OCIDefine *no_rowid_dp;
    OCIDefine *c_id_dp;
    OCIDefine *o_rowid_dp;
    OCIDefine *cons_dp;

```

```

OCIDefine *amt_dp;

};
int norow;

typedef struct delctx delctx;
struct pldelctx {

    ub2 del_d_id_len[NDISTS];
    ub2 del_o_id_len[NDISTS];

    ub2 w_id_len;
    ub2 d_id_len[NDISTS];
    ub2 o_c_id_len[NDISTS];
    ub2 sums_len[NDISTS];
    ub2 carrier_id_len;
    ub2 ordcnt_len;
    ub2 del_date_len;

    int del_o_id[NDISTS];
    int del_d_id[NDISTS];
    int o_c_id[NDISTS];
#ifdef USE_IEEE_NUMBER
    float sums[NDISTS];
#else
    int sums[NDISTS];
#endif
#endif
    OCIDate del_date;
    int carrier_id;
    int ordcnt;

    ub4 del_o_id_rcnt;
    ub4 del_d_id_rcnt;
    ub4 o_c_id_rcnt;
    ub4 sums_rcnt;

    int retry;
    OCISTmt *curp1;
    OCISTmt *curp2;
    OCIBind *w_id_bp;
    OCIBind *d_id_bp;
    OCIBind *o_id_bp;
    OCIBind *o_c_id_bp;
    OCIBind *ordcnt_bp;
    OCIBind *sums_bp;
    OCIBind *del_date_bp;
    OCIBind *carrier_id_bp;
    OCIBind *retry_bp;

    int norow;
};

typedef struct pldelctx pldelctx;

struct amtctx {
    int ol_amt[NITEMS];
    sb2 ol_amt_ind[NITEMS];
    ub4 ol_amt_len[NITEMS];
    ub2 ol_amt_rcode[NITEMS];
    int ol_cnt;
};

typedef struct amtctx amtctx;

struct ordctx {

    ub2 c_rowid_len[100];
    ub2 ol_supply_w_id_len[NITEMS];
    ub2 ol_i_id_len[NITEMS];
    ub2 ol_quantity_len[NITEMS];
    ub2 ol_amount_len[NITEMS];
    ub2 ol_delivery_d_len[NITEMS];
    ub2 ol_w_id_len;
    ub2 ol_d_id_len;
    ub2 ol_o_id_len;

    ub4 ol_supply_w_id_csize;
    ub4 ol_i_id_csize;
    ub4 ol_quantity_csize;
    ub4 ol_amount_csize;
    ub4 ol_delivery_d_csize;
    ub4 ol_w_id_csize;
    ub4 ol_d_id_csize;
    ub4 ol_o_id_csize;

    OCISTmt *curo0;
    OCISTmt *curo1;
    OCISTmt *curo2;
    OCISTmt *curo3;
    OCISTmt *curo4;
    OCIBind *c_id_bp;
    OCIBind *w_id_bp[4];
    OCIBind *d_id_bp[4];
    OCIBind *c_last_bp[2];
    OCIBind *o_id_bp;
    OCIBind *c_rowid_bp;
    OCIDefine *c_rowid_dp;
    OCIDefine *c_last_dp[2];
    OCIDefine *c_id_dp;
    OCIDefine *c_first_dp[2];
    OCIDefine *c_middle_dp[2];
    OCIDefine *c_balance_dp[2];
    OCIDefine *o_id_dp[2];
    OCIDefine *o_entry_d_dp[2];
    OCIDefine *o_cr_id_dp[2];
    OCIDefine *o_ol_cnt_dp[2];

```



```

OCIDefine *ol_d_dp;
OCIDefine *ol_i_id_dp;
OCIDefine *ol_supply_w_id_dp;
OCIDefine *ol_quantity_dp;
OCIDefine *ol_amount_dp;
OCIDefine *ol_d_base_dp;
OCIDefine *c_count_dp;
OCIRowid *c_rowid_ptr[100];
OCIRowid *c_rowid_cust;
int cs;
int cust_idx;
int norow;
int rcount;
int somerows;
};

typedef struct ordctx ordctx;

struct defctx
{
    boolean reexec;
    ub4 count;
};
typedef struct defctx defctx;

struct payctx {
    OCIStmt *curpi;
    OCIStmt *curp0;
    OCIStmt *curp1;
    OCIBind *w_id_bp[2];
    ub2 w_id_len;

    OCIBind *d_id_bp[2];
    ub2 d_id_len;

    OCIBind *c_w_id_bp[2];
    ub2 c_w_id_len;

    OCIBind *c_d_id_bp[2];
    ub2 c_d_id_len;

    OCIBind *c_id_bp[2];
    ub2 c_id_len;

    OCIBind *h_amount_bp[2];
    ub2 h_amount_len;

    OCIBind *c_last_bp[2];
    ub2 c_last_len;

    OCIBind *w_street_1_bp[2];
    ub2 w_street_1_len;

    OCIBind *w_street_2_bp[2];
    ub2 w_street_2_len;

    OCIBind *w_city_bp[2];
    ub2 w_city_len;

    OCIBind *w_state_bp[2];
    ub2 w_state_len;

    OCIBind *w_zip_bp[2];
    ub2 w_zip_len;

    OCIBind *d_street_1_bp[2];
    ub2 d_street_1_len;

    OCIBind *d_street_2_bp[2];
    ub2 d_street_2_len;

    OCIBind *d_city_bp[2];
    ub2 d_city_len;

    OCIBind *d_state_bp[2];
    ub2 d_state_len;

    OCIBind *d_zip_bp[2];
    ub2 d_zip_len;

    OCIBind *c_first_bp[2];
    ub2 c_first_len;

    OCIBind *c_middle_bp[2];
    ub2 c_middle_len;

    OCIBind *c_street_1_bp[2];
    ub2 c_street_1_len;

    OCIBind *c_street_2_bp[2];
    ub2 c_street_2_len;

    OCIBind *c_city_bp[2];
    ub2 c_city_len;

    OCIBind *c_state_bp[2];
    ub2 c_state_len;

    OCIBind *c_zip_bp[2];
    ub2 c_zip_len;

    OCIBind *c_phone_bp[2];
    ub2 c_phone_len;

    OCIBind *c_since_bp[2];
    ub2 c_since_len;
};

```

```

OCIBind *c_credit_bp[2];
ub2 c_credit_len;

OCIBind *c_credit_lim_bp[2];
ub2 c_credit_lim_len;

OCIBind *c_discount_bp[2];
ub2 c_discount_len;

OCIBind *c_balance_bp[2];
ub2 c_balance_len;

OCIBind *c_data_bp[2];
ub2 c_data_len;

OCIBind *h_date_bp[2];
ub2 h_date_len;

OCIBind *retries_bp[2];
ub2 retries_len;

OCIBind *cr_date_bp[2];
ub2 cr_date_len;

OCIBind *byln_bp[2];
ub2 byln_len;
};

typedef struct payctx payctx;

struct stoctx {
    OCIStmt *curs;
    OCIBind *w_id_bp;
    OCIBind *d_id_bp;
    OCIBind *threshold_bp;
#ifdef PLSQLSTO
    OCIBind *low_stock_bp;
#else
    OCIDefine *low_stock_bp;
#endif
    int norow;
};

typedef struct stoctx stoctx;

/* New order */

struct newinstruct {
    int w_id;
    int d_id;
    int c_id;
    int ol_i_id[15];
    int ol_supply_w_id[15];
    int ol_quantity[15];
};

struct newoutstruct {
    int terror;
    int o_id;
    int o_ol_cnt;
    char c_last[17];
    char c_credit[3];
    float c_discount;
    float w_tax;
    float d_tax;
    char o_entry_d[20];
    float total_amount;
    char i_name[15][25];
    int s_quantity[15];
    char brand_generic[15];
    float i_price[15];
    float ol_amount[15];
    char status[26];
    int retry;
};

struct newstruct {
    struct newinstruct newin;
    struct newoutstruct newout;
};

/* Payment */

struct payinstruct {
    int w_id;
    int d_id;
    int c_w_id;
    int c_d_id;
    int c_id;
    int bylastname;
    int h_amount;
    char c_last[17];
};

struct payoutstruct {
    int terror;
    char w_street_1[21];
    char w_street_2[21];
    char w_city[21];
    char w_state[3];
    char w_zip[10];
    char d_street_1[21];
    char d_street_2[21];
    char d_city[21];
};

```

```

char d_state[3];
char d_zip[10];
int c_id;
char c_first[17];
char c_middle[3];
char c_last[17];
char c_street_1[21];
char c_street_2[21];
char c_city[21];
char c_state[3];
char c_zip[10];
char c_phone[17];
char c_since[11];
char c_credit[3];
double c_credit_lim;
float c_discount;
double c_balance;
char c_data[201];
char h_date[20];
int retry;
};

struct paystruct {
    struct payinstruct payin;
    struct payoutstruct payout;
};

/* Order status */

struct ordinstruct {
    int w_id;
    int d_id;
    int c_id;
    int bylastname;
    char c_last[17];
};

struct ordoutstruct {
    int terror;
    int c_id;
    char c_last[17];
    char c_first[17];
    char c_middle[3];
    double c_balance;
    int o_id;
    char o_entry_d[20];
    int o_carrier_id;
    int o_ol_cnt;
    int ol_supply_w_id[15];
    int ol_i_id[15];
    int ol_quantity[15];
    float ol_amount[15];
    char ol_delivery_d[15][11];
    int retry;
};

struct ordstruct {
    struct ordinstruct ordin;
    struct ordoutstruct ordout;
};

/* Delivery */

struct delinstruct {
    int w_id;
    int o_carrier_id;
    double qtime;
    int in_timing_int;
    int plsqflag;
};

struct deloutstruct {
    int terror;
    int retry;
};

struct delstruct {
    struct delinstruct delin;
    struct deloutstruct delout;
};

/* Stock level */

```

```

struct stoinstruct {
    int w_id;
    int d_id;
    int threshold;
};

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

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

-----
views.sql
-----
connect tpcc/tpcc;
set echo on;

create or replace view wh_cust
(w_id, w_tax, c_id, c_d_id, c_w_id, c_discount, c_last,
c_credit)
as select w.w_id, w.w_tax,
        c.c_id, c.c_d_id, c.c_w_id, c.c_discount, c.c_last,
        c.c_credit
    from cust c, ware w
    where w.w_id = c.c_w_id;

create or replace view wh_dist
(w_id, d_id, d_tax, d_next_o_id, w_tax )
as select w.w_id, d.d_id, d.d_tax, d.d_next_o_id, w.w_tax
    from dist d, ware w
    where w.w_id = d.d_w_id;

create or replace view stock_item
(i_id, s_w_id, i_price, i_name, i_data, s_data, s_quantity,
s_order_cnt, s_ytd, s_remote_cnt,
s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10)
as
select /*+ leading(s) use_nl(i) */
    i.i_id, s.w_id, i.i_price, i.i_name, i.i_data, s_data,
    s_quantity,
    s_order_cnt, s_ytd, s_remote_cnt,
    s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
    s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10
    from stok s, item i
    where i.i_id = s.s_i_id;

set echo off;

# PRTE COMMAND FILE
# C_LAST is the constant value used for customer last
names.
database.set network_variable C_LAST 87

```

Appendix B:

Database Design

```
-----
analyze.sql
-----
spool analyze.log;
set echo on;

connect tpcc/tpcc

dbms_stats.gather_table_stats('tpcc','stok','',10);
dbms_stats.gather_table_stats('tpcc','cust','',10);
dbms_stats.gather_table_stats('tpcc','ordr','',10);
dbms_stats.gather_table_stats('tpcc','ordl','',10);
dbms_stats.gather_table_stats('tpcc','nord','',10);
dbms_stats.gather_table_stats('tpcc','hist','',10);
dbms_stats.gather_table_stats('tpcc','dist','',100);
dbms_stats.gather_table_stats('tpcc','item','',100);
dbms_stats.gather_table_stats('tpcc','ware','',100);

set echo off;
spool off;

exit sql.sqlcode;

-----
assigntemp.sql
-----
spool assigntemp.log;

set echo on;

alter user tpcc temporary tablespace temp_0;

set echo off;
spool off;

exit ;

-----
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
rem 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
rem uses 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
rem below, please
rem replace svrmgrl with sqldb lmode=y.
rem
rem Modification History:
rem
rem wbatist      16-Jun-1996      Create two additional views to
rem keep
rem track of the number of clones
rem in each
rem tablespace.
rem
rem wbatist      24-May-1995      Add the state check for the
rem cbf view
rem to ensure that cloned blocks
rem are not
rem counted.
rem
rem
rem connect $oracle_dba/$oracle_dba_password;
rem set echo on;
rem drop view cbf;
rem create view cbf as
rem select distinct(dbarfil) file#, count(1) blocks
rem from x$bh
rem where dbarfil > 0 and state <> 3
rem group by dbarfil;
rem drop view cbt;
rem create view cbt as
rem select ts$.name name,sum(cbf.blocks) buffers
rem from cbf, file$, ts$
rem where cbf.file#=file$.file# and file$.ts#=ts$.ts#
rem group by file$.ts#, ts$.name;
rem drop view cbfcln;
rem create view cbfcln as
rem select distinct(dbarfil) file#, count(1) blocks
rem from x$bh
rem where dbarfil > 0
rem group by dbarfil;
```

```
drop view cbtcln;
create view cbtcln as
select ts$.name name,sum(cbfcln.blocks) buffers
from cbfcln, file$, ts$
where cbfcln.file#=file$.file# and file$.ts#=ts$.ts#
group by file$.ts#, ts$.name;

set echo off;

-----
createdb.sql
-----
/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreatedb.sh Fri Jul 2
16:04:42 CDT 2004 */
spool createdb.log

set echo on

shutdown abort

startup pfile=p_create.ora nomount
create database tpcc
controlfile reuse
maxinstances 1
datafile
'/home/oracle/dev/system_1' size 400M reuse
logfile '/home/oracle/dev/log_1_1' size 7910M reuse,
'/home/oracle/dev/log_1_2' size 7910M reuse
sysaux datafile '/home/oracle/dev/tpccaux' size 120M reuse ;

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

set echo off
exit sql.sqlcode

-----
createindex_icust1.sql
-----
/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreateindex.sh Fri Jul 2
16:04:54 CDT 2004 */
set timing on
set sqlblanklines on
spool createindex_icust1.log ;
set echo on ;
drop index icust1 ;
create unique index icust1 on cust ( c_w_id
, c_d_id
, c_id )
pctfree 1 initrans 3
storage ( buffer_pool default )
parallel 2
compute statistics
tablespace icust1_0 ;
set echo off
spool off
exit sql.sqlcode;

-----
createindex_icust2.sql
-----
/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreateindex.sh Fri Jul 2
16:04:55 CDT 2004 */
set timing on
set sqlblanklines on
spool createindex_icust2.log ;
set echo on ;
drop index icust2 ;
create unique index icust2 on cust ( c_last
, c_w_id
, c_d_id
, c_first
, c_id )
pctfree 1 initrans 3
storage ( buffer_pool default )
compute statistics
tablespace icust2_0 ;
set echo off
spool off
exit sql.sqlcode;

-----
createindex_idist.sql
-----
```

```

/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreateindex.sh Fri Jul 2
16:04:56 CDT 2004 */
set timing on
set sqlblanklines on
spool createindex_idist.log ;
set echo on ;
drop index idist ;
create unique index idist on dist ( d_w_id
, d_id )
pctfree 5 initrans 3
storage ( buffer_pool default )
compute statistics
tablespace idist_0 ;
set echo off
spool off
exit sql.sqlcode;

```

```
-----
createindex_iitem.sql
-----
```

```

/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreateindex.sh Fri Jul 2
16:04:57 CDT 2004 */
set timing on
set sqlblanklines on
spool createindex_iitem.log ;
set echo on ;
drop index iitem ;
create unique index iitem on item ( i_id )
pctfree 5 initrans 4
storage ( buffer_pool default )

compute statistics
tablespace iitem_0 ;
set echo off
spool off
exit sql.sqlcode;

```

```
-----
createindex_inord.sql
-----
```

```

/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreateindex.sh Fri Jul 2
16:04:59 CDT 2004 */
set timing on
exit 0;

```

```
-----
createindex_iordl.sql
-----
```

```

/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreateindex.sh Fri Jul 2
16:04:59 CDT 2004 */
set timing on
exit 0;

```

```
-----
createindex_iordr1.sql
-----
```

```

/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreateindex.sh Fri Jul 2
16:04:57 CDT 2004 */
set timing on
exit 0;

```

```
-----
createindex_iordr2.sql
-----
```

```

/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreateindex.sh Fri Jul 2
16:04:58 CDT 2004 */
set timing on
set sqlblanklines on
spool createindex_iordr2.log ;
set echo on ;
drop index iordr2 ;
create unique index iordr2 on ordr ( o_c_id
, o_d_id
, o_w_id
, o_id )
pctfree 25 initrans 4
storage ( buffer_pool default )
compute statistics
tablespace iordr2_0 ;
set echo off
spool off
exit sql.sqlcode;

```

```
-----
createindex_istok.sql
-----
```

```

/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreateindex.sh Fri Jul 2
16:04:56 CDT 2004 */
set timing on

```

```

set sqlblanklines on
spool createindex_istok.log ;
set echo on ;
drop index istok ;
create unique index istok on stok ( s_i_id
, s_w_id )
pctfree 1 initrans 3
storage ( buffer_pool default )
compute statistics
tablespace istok_0 ;
set echo off
spool off
exit sql.sqlcode;

```

```
-----
createindex_iware.sql
-----
```

```

/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreateindex.sh Fri Jul 2
16:04:54 CDT 2004 */
set timing on
set sqlblanklines on
spool createindex_iware.log ;
set echo on ;
drop index iware ;
create unique index iware on ware ( w_id )
pctfree 1 initrans 3
storage ( buffer_pool default )
compute statistics
tablespace iware_0 ;
set echo off
spool off
exit sql.sqlcode;

```

```
-----
createspacestats.sql
-----
```

```

@space_init
@space_get 12 10
@space_rpt
spool off
exit sql.sqlcode;

```

```
-----
createstoredprocs.sql
-----
```

```

spool createstoredprocs.log
@tkvcin.in.sql
spool off
exit sql.sqlcode;

```

```
-----
createtable_cust.sql
-----
```

```

/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreatetable.sh Fri Jul 2
16:04:44 CDT 2004 */
set timing on
set sqlblanklines on
spool createtable_cust.log
set echo on
drop cluster custcluster including tables ;

create cluster custcluster (
c_id number
, c_d_id number
, c_w_id number
)
single table
hashkeys 129600000
hash is ( (c_id * ( 4320 * 10 ) + c_w_id * 10 + c_d_id ) )
size 180
pctfree 0 initrans 3
storage ( buffer_pool recycle ) parallel ( degree 2 )
tablespace cust_0;

```

```
create table cust (
```

```

c_id number
, c_d_id number
, c_w_id number
, c_discount number
, c_credit char(2)
, c_last varchar2(16)
, c_first varchar2(16)
, c_credit_lim number
, c_balance number
, c_ytd_payment number
, c_payment_cnt number
, c_delivery_cnt number
, c_street_1 varchar2(20)
, c_street_2 varchar2(20)
, c_city varchar2(20)
, c_state char(2)
, c_zip char(9)
, c_phone char(16)
, c_since date
, c_middle char(2)
, c_data char(500)
)
cluster custcluster (

```

```

c_id
, c_d_id
, c_w_id
);
set echo off
spool off
exit sql.sqlcode;

-----
createtable_dist.sql
-----
/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreatetable.sh Fri Jul 2
16:04:46 CDT 2004 */
set timing on
set sqlblanklines on
spool createtable_dist.log
set echo on
drop cluster distcluster including tables ;

create cluster distcluster (
d_id number
, d_w_id number
)
single table
hashkeys 43200
hash is ( ((d_w_id * 10) + d_id )
size 1448
intrans 4
storage ( buffer_pool default )
tablespace dist_0;

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

-----
createtable_hist.sql
-----
/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreatetable.sh Fri Jul 2
16:04:47 CDT 2004 */
set timing on
set sqlblanklines on
spool createtable_hist.log
set echo on
drop table hist ;

create table hist (
h_c_id number
, h_c_d_id number
, h_c_w_id number
, h_d_id number
, h_w_id number
, h_date date
, h_amount number
, h_data varchar2(24)
)
pctfree 5 intrans 4
storage ( buffer_pool recycle )
tablespace hist_0 ;
set echo off
spool off
exit sql.sqlcode;

-----
createtable_item.sql
-----
/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreatetable.sh Fri Jul 2
16:04:50 CDT 2004 */
set timing on
set sqlblanklines on
spool createtable_item.log
set echo on
drop cluster itemcluster including tables ;

create cluster itemcluster (
i_id number(6,0)
)
single table
hashkeys 100000
hash is ( (i_id) )
size 120
pctfree 0 intrans 3

```

```

storage ( buffer_pool keep )
tablespace item_0;

create table item (
i_id number(6,0)
, i_name varchar2(24)
, i_price number
, i_data varchar2(50)
, i_im_id number
)
cluster itemcluster (
i_id
);
set echo off
spool off
exit sql.sqlcode;

-----
createtable_nord.sql
-----
/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreatetable.sh Fri Jul 2
16:04:52 CDT 2004 */
set timing on
set sqlblanklines on
spool createtable_nord.log
set echo on
drop cluster nordcluster_queue including tables ;

create cluster nordcluster_queue (
no_w_id number
, no_d_id number
, no_o_id number SORT
)
hashkeys 43200
hash is ( (no_w_id - 1) * 10 + no_d_id - 1 )
size 190
tablespace nord_0;

create table nord (
no_w_id number
, no_d_id number
, no_o_id number sort
, constraint nord_uk primary key ( no_w_id
, no_d_id
, no_o_id )
)
cluster nordcluster_queue (
no_w_id
, no_d_id
, no_o_id
);
set echo off
spool off
exit sql.sqlcode;

-----
createtable_ordl.sql
-----
/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreatetable.sh Fri Jul 2
16:04:52 CDT 2004 */
set timing on
set sqlblanklines on
spool createtable_ordl.log
set echo on
create table ordl (
ol_w_id number
, ol_d_id number
, ol_o_id number sort
, ol_number number sort
, ol_i_id number
, ol_delivery_d date
, ol_amount number
, ol_supply_w_id number
, ol_quantity number
, ol_dist_info char(24)
, constraint ordl_uk primary key (ol_w_id, ol_d_id, ol_o_id,
ol_number ) CLUSTER ordcluster_queue(ol_w_id, ol_d_id,
ol_o_id, ol_number) ;
set echo off
spool off
exit sql.sqlcode;

-----
createtable_ordr.sql
-----
/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreatetable.sh Fri Jul 2
16:04:50 CDT 2004 */
set timing on
set sqlblanklines on
spool createtable_ordr.log
set echo on
drop cluster ordcluster_queue including tables ;

create cluster ordcluster_queue (
o_w_id number
, o_d_id number
, o_id number SORT
, o_number number SORT

```

```

)
hashkeys 43200
hash is ( (o_w_id - 1) * 10 + o_d_id - 1 )
size 1490
tablespace ordr_0;

create table ordr (
  o_id number sort
, o_w_id number
, o_d_id number
, o_c_id number
, o_carrier_id number
, o_ol_cnt number
, o_all_local number
, o_entry_d date
, constraint ordr_uk primary key ( o_w_id
, o_d_id
, o_id )
)
cluster ordrcluster_queue (
  o_w_id
, o_d_id
, o_id
);
set echo off
spool off
exit sql.sqlcode;

-----
createtable_stok.sql
-----
/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreatetable.sh Fri Jul 2
16:04:48 CDT 2004 */
set timing on
set sqlblanklines on
spool createtable_stok.log
set echo on
drop cluster stokcluster including tables ;

create cluster stokcluster (
  s_i_id number
, s_w_id number
)
single table
hashkeys 432000000
hash is ( (s_i_id * 4320 + s_w_id) )
size 256
pctfree 0 intrans 2 maxtrans 2
storage ( buffer_pool keep ) parallel ( degree 2 )
tablespace stok_0;

create table stok (
  s_i_id number
, s_w_id number
, s_quantity number
, s_ytd number
, s_order_cnt number
, s_remote_cnt number
, s_data varchar2(50)
, s_dist_01 varchar2(24)
, s_dist_02 varchar2(24)
, s_dist_03 varchar2(24)
, s_dist_04 varchar2(24)
, s_dist_05 varchar2(24)
, s_dist_06 varchar2(24)
, s_dist_07 varchar2(24)
, s_dist_08 varchar2(24)
, s_dist_09 varchar2(24)
, s_dist_10 varchar2(24)
)
cluster stokcluster (
  s_i_id
, s_w_id
);
set echo off
spool off
exit sql.sqlcode;

-----
createtable_ware.sql
-----
/* created automatically by
/home/oracle/tpcc_96000/scripts/buildcreatetable.sh Fri Jul 2
16:04:43 CDT 2004 */
set timing on
set sqlblanklines on
spool createtable_ware.log
set echo on
drop cluster warecluster including tables ;

create cluster warecluster (
  w_id number
)
single table
hashkeys 4320
hash is ( (w_id - 1) )
size 1448
intrans 2
storage ( buffer_pool default )
tablespace ware_0;

create table ware (

```

```

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

-----
createts.sh
-----
#created automatically by
/home/oracle/tpcc_96000/scripts/buildcreatets.sh Fri Jul 2
16:04:30 CDT 2004

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

$tpcc_createts ware 1 1      20M 11264K unix 0      0 2 auto t
if expr $? != 0 > /dev/null; then
  echo Creating tablespace for ware failed.  Exiting.
  exit 0
fi

# Tablespace cust, ts size 112860M (115568640K)
# each file 6270M (6420480K)
# extents 103500K (103500K)
# 18 files

$tpcc_createts cust 18 1      6270M 103500K unix 0      1 2
auto t
if expr $? != 0 > /dev/null; then
  echo Creating tablespace for cust failed.  Exiting.
  exit 0
fi

# Tablespace dist, ts size 100M (102400K)
# each file 100M (102400K)
# extents 90880K (90880K)
# 1 files

$tpcc_createts dist 1 1      100M 90880K unix 0      19 2 auto
t
if expr $? != 0 > /dev/null; then
  echo Creating tablespace for dist failed.  Exiting.
  exit 0
fi

# Tablespace hist, ts size 11280M (11550720K)
# each file 1880M (1925120K)
# extents 100936K (100936K)
# 6 files

$tpcc_createts hist 6 1      1880M 100936K unix 0      20 2
auto t
if expr $? != 0 > /dev/null; then
  echo Creating tablespace for hist failed.  Exiting.
  exit 0
fi

# Tablespace stok, ts size 126720M (129761280K)
# each file 7040M (7208960K)
# extents 104378K (104378K)
# 18 files

$tpcc_createts stok 18 1      7040M 104378K unix 0      26 2
auto t
if expr $? != 0 > /dev/null; then
  echo Creating tablespace for stok failed.  Exiting.
  exit 0
fi

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

$tpcc_createts item 1 1      20M 16892K unix 0      44 2 auto
t
if expr $? != 0 > /dev/null; then
  echo Creating tablespace for item failed.  Exiting.
  exit 0
fi

# Tablespace ordr, ts size 153540M (157224960K)
# each file 25590M (26204160K)
# extents 103136K (103136K)
# 6 files

$tpcc_createts ordr 6 1      25590M 103136K unix 0      45 2
16K t
if expr $? != 0 > /dev/null; then
  echo Creating tablespace for ordr failed.  Exiting.
  exit 0
fi

```

```

# Tablespace nord, ts size 1480M (1515520K)
# each file 1480M (1515520K)
# extents 150784K (150784K)
# 1 files

$tpcc_createts nord 1 1      1480M 150784K unix 0      51 2
auto t
  if expr $? != 0 > /dev/null; then
    echo Creating tablespace for nord failed.  Exiting.
    exit 0
  fi

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

$tpcc_createts iware 1 1      20M 11264K unix 0      52 2 auto
t
  if expr $? != 0 > /dev/null; then
    echo Creating tablespace for iware failed.  Exiting.
    exit 0
  fi

# Tablespace icust1, ts size 3260M (3338240K)
# each file 3260M (3338240K)
# extents 1667310K (1667310K)
# 1 files

$tpcc_createts icust1 1 1      3260M 1667310K unix 0      53 2
auto t
  if expr $? != 0 > /dev/null; then
    echo Creating tablespace for icust1 failed.  Exiting.
    exit 0
  fi

# Tablespace icust2, ts size 7620M (7802880K)
# each file 1270M (1300480K)
# extents 1297024K (1297024K)
# 6 files

$tpcc_createts icust2 6 1      1270M 1297024K unix 0      54 2
auto t
  if expr $? != 0 > /dev/null; then
    echo Creating tablespace for icust2 failed.  Exiting.
    exit 0
  fi

# Tablespace idist, ts size 30M (30720K)
# each file 30M (30720K)
# extents 22624K (22624K)
# 1 files

$tpcc_createts idist 1 1      30M 22624K unix 0      60 2 auto
t
  if expr $? != 0 > /dev/null; then
    echo Creating tablespace for idist failed.  Exiting.
    exit 0
  fi

# Tablespace istok, ts size 9620M (9850880K)
# each file 4810M (4925440K)
# extents 1638076K (1638076K)
# 2 files

$tpcc_createts istok 2 1      4810M 1638076K unix 0      61 2
auto t
  if expr $? != 0 > /dev/null; then
    echo Creating tablespace for istok failed.  Exiting.
    exit 0
  fi

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

$tpcc_createts iitem 1 1      20M 11264K unix 0      63 2 auto
t
  if expr $? != 0 > /dev/null; then
    echo Creating tablespace for iitem failed.  Exiting.
    exit 0
  fi

# Tablespace iordr2, ts size 6780M (6942720K)
# each file 1130M (1157120K)
# extents 95510K (95510K)
# 6 files

$tpcc_createts iordr2 6 1      1130M 95510K unix 0      64 2
auto t
  if expr $? != 0 > /dev/null; then
    echo Creating tablespace for iordr2 failed.  Exiting.
    exit 0
  fi

# Tablespace temp, ts size 24120M (24698880K)
# each file 4020M (416480K)
# extents 205656K (205656K)
# 6 files

$tpcc_createts temp 6 1      4020M 205656K unix 1      70 2
auto t
  if expr $? != 0 > /dev/null; then
    echo Creating tablespace for temp failed.  Exiting.
    exit 0
  fi

```

```
-----
createuser.sql
-----
```

```
spool createusertpcc.log;
```

```
set echo on;
```

```
create user tpcc identified by tpcc;
```

```
grant dba to tpcc;
```

```
set echo off;
```

```
spool off;
```

```
exit ;
```

```
-----
files.dat
-----
```

```
#generated automatically by files.dat Fri Jul 2 16:05:00 CDT
2004
```

```

/home/oracle/dev//ware_0_0 20M
/home/oracle/dev//cust_0_0 6270M
/home/oracle/dev//cust_0_1 6270M
/home/oracle/dev//cust_0_2 6270M
/home/oracle/dev//cust_0_3 6270M
/home/oracle/dev//cust_0_4 6270M
/home/oracle/dev//cust_0_5 6270M
/home/oracle/dev//cust_0_6 6270M
/home/oracle/dev//cust_0_7 6270M
/home/oracle/dev//cust_0_8 6270M
/home/oracle/dev//cust_0_9 6270M
/home/oracle/dev//cust_0_10 6270M
/home/oracle/dev//cust_0_11 6270M
/home/oracle/dev//cust_0_12 6270M
/home/oracle/dev//cust_0_13 6270M
/home/oracle/dev//cust_0_14 6270M
/home/oracle/dev//cust_0_15 6270M
/home/oracle/dev//cust_0_16 6270M
/home/oracle/dev//cust_0_17 6270M
/home/oracle/dev//dist_0_0 100M
/home/oracle/dev//hist_0_0 1880M
/home/oracle/dev//hist_0_1 1880M
/home/oracle/dev//hist_0_2 1880M
/home/oracle/dev//hist_0_3 1880M
/home/oracle/dev//hist_0_4 1880M
/home/oracle/dev//hist_0_5 1880M
/home/oracle/dev//stok_0_0 7040M
/home/oracle/dev//stok_0_1 7040M
/home/oracle/dev//stok_0_2 7040M
/home/oracle/dev//stok_0_3 7040M
/home/oracle/dev//stok_0_4 7040M
/home/oracle/dev//stok_0_5 7040M
/home/oracle/dev//stok_0_6 7040M
/home/oracle/dev//stok_0_7 7040M
/home/oracle/dev//stok_0_8 7040M
/home/oracle/dev//stok_0_9 7040M
/home/oracle/dev//stok_0_10 7040M
/home/oracle/dev//stok_0_11 7040M
/home/oracle/dev//stok_0_12 7040M
/home/oracle/dev//stok_0_13 7040M
/home/oracle/dev//stok_0_14 7040M
/home/oracle/dev//stok_0_15 7040M
/home/oracle/dev//stok_0_16 7040M
/home/oracle/dev//stok_0_17 7040M
/home/oracle/dev//item_0_0 20M
/home/oracle/dev//ordr_0_0 25590M
/home/oracle/dev//ordr_0_1 25590M
/home/oracle/dev//ordr_0_2 25590M
/home/oracle/dev//ordr_0_3 25590M
/home/oracle/dev//ordr_0_4 25590M
/home/oracle/dev//ordr_0_5 25590M
/home/oracle/dev//nord_0_0 1480M
/home/oracle/dev//iware_0_0 20M
/home/oracle/dev//icust1_0_0 3260M
/home/oracle/dev//icust2_0_0 1270M
/home/oracle/dev//icust2_0_1 1270M
/home/oracle/dev//icust2_0_2 1270M
/home/oracle/dev//icust2_0_3 1270M
/home/oracle/dev//icust2_0_4 1270M
/home/oracle/dev//icust2_0_5 1270M
/home/oracle/dev//idist_0_0 30M
/home/oracle/dev//istok_0_0 4810M
/home/oracle/dev//istok_0_1 4810M
/home/oracle/dev//iitem_0_0 20M
/home/oracle/dev//iordr2_0_0 1130M
/home/oracle/dev//iordr2_0_1 1130M
/home/oracle/dev//iordr2_0_2 1130M
/home/oracle/dev//iordr2_0_3 1130M
/home/oracle/dev//iordr2_0_4 1130M
/home/oracle/dev//iordr2_0_5 1130M
/home/oracle/dev//temp_0_0 4020M
/home/oracle/dev//temp_0_1 4020M
/home/oracle/dev//temp_0_2 4020M
/home/oracle/dev//temp_0_3 4020M
/home/oracle/dev//temp_0_4 4020M
/home/oracle/dev//temp_0_5 4020M
/home/oracle/dev//tpccaux 120M
/home/oracle/dev//control_001 25M
/home/oracle/dev//control_002 25M
/home/oracle/dev//sp_0 1024M
/home/oracle/dev//log_1_1 7910M
/home/oracle/dev//log_1_2 7910M

```

```
/home/oracle/dev/system_1 400M
/home/oracle/dev/roll1 8096M
```

```
-----
loadcust.sh
-----
```

```
#created automatically by
/home/oracle/tpcc_96000/scripts/evenload.sh Fri Jul 2 16:04:53
CDT 2004
rm -f loadcust*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 4320 -C -l 1 -m 750 >> loadcust0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 4320 -C -l 751 -m 1500 >> loadcust1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 4320 -C -l 1501 -m 2250 >> loadcust2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 4320 -C -l 2251 -m 3000 >> loadcust3.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
    wait $curproc
    error=`expr $? + $error`
done
exit `expr $error != 0`
```

```
-----
loaddist.sh
-----
```

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -d > loaddist.log 2>&1
```

```
-----
loadhist.sh
-----
```

```
#created automatically by
/home/oracle/tpcc_96000/scripts/evenload.sh Fri Jul 2 16:04:53
CDT 2004
rm -f loadhist*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 4320 -h -b 1 -e 1080 >> loadhist0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 4320 -h -b 1081 -e 2160 >> loadhist1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 4320 -h -b 2161 -e 3240 >> loadhist2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 4320 -h -b 3241 -e 4320 >> loadhist3.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
    wait $curproc
    error=`expr $? + $error`
done
exit `expr $error != 0`
```

```
-----
loaditem.sh
-----
```

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -i > loaditem.log 2>&1
```

```
-----
loadnord.sh
-----
```

```
#created automatically by
/home/oracle/tpcc_96000/scripts/evenload.sh Fri Jul 2 16:04:53
CDT 2004
rm -f loadnord*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 4320 -n -b 1 -e 4320 >> loadnord0.log 2>&1 &
allprocs="$allprocs ${!}"
#$tpcc_load -M 4320 -n -b 1081 -e 2160 >> loadnord1.log 2>&1 &
allprocs="$allprocs ${!}"
#$tpcc_load -M 4320 -n -b 2161 -e 3240 >> loadnord2.log 2>&1 &
allprocs="$allprocs ${!}"
#$tpcc_load -M 4320 -n -b 3241 -e 4320 >> loadnord3.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
    wait $curproc
    error=`expr $? + $error`
done
exit `expr $error != 0`
```

```
-----
loadordrord1.sh
-----
```

```
#created automatically by
/home/oracle/tpcc_96000/scripts/evenload.sh Fri Jul 2 16:04:53
CDT 2004
rm -f loadordrord1*.log
cd $tpcc_bench
allprocs=
```

```
$tpcc_load -M 4320 -o ${tpcc_disks_location}dummy0.dat -b 1 -e
1080 >> loadordrord10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 4320 -o ${tpcc_disks_location}dummy1.dat -b 1081 -
e 2160 >> loadordrord11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 4320 -o ${tpcc_disks_location}dummy2.dat -b 2161 -
e 3240 >> loadordrord12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 4320 -o ${tpcc_disks_location}dummy3.dat -b 3241 -
e 4320 >> loadordrord13.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
    wait $curproc
    error=`expr $? + $error`
done
exit `expr $error != 0`
```

```
-----
loadstok.sh
-----
```

```
#created automatically by
/home/oracle/tpcc_96000/scripts/evenload.sh Fri Jul 2 16:04:53
CDT 2004
rm -f loadstok*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 4320 -S -j 1 -k 25000 >> loadstok0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 4320 -S -j 25001 -k 50000 >> loadstok1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 4320 -S -j 50001 -k 75000 >> loadstok2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 4320 -S -j 75001 -k 100000 >> loadstok3.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
    wait $curproc
    error=`expr $? + $error`
done
exit `expr $error != 0`
```

```
-----
loadware.sh
-----
```

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -w > loadware.log 2>&1
```

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

```
compatible = 10.1.0.0.0
db_name = tpcc
control_files = /home/oracle/dev/control_001
parallel_max_servers = 100
recovery_parallelism = 40
db_files = 176
db_cache_size = 8192M
db_8k_cache_size = 3072M
db_16k_cache_size = 8192M
dml_locks = 500
statistics_level = basic
log_buffer = 1048576
processes = 150
sessions = 150
transactions = 150
shared_pool_size = 1536M
cursor_space_for_time = TRUE
db_block_size = 2048
undo_management = auto
undo_retention = 2
_in_memory_undo=false
_undo_autotune=false
plssql_optimize_level=2
UNDO_TABLESPACE = undo_1
db_4k_cache_size = 20M
_column_compression_factor=175
```

```
-----
p_create.ora
-----
```

```
compatible = 10.1.0.0.0
db_name = tpcc
control_files = /home/oracle/dev/control_001
db_block_size = 2048
db_cache_size = 8192M
db_8k_cache_size = 3072M
log_buffer = 1048576
db_16k_cache_size = 8192M
undo_management = manual
statistics_level = basic
shared_pool_size = 1536M
_in_memory_undo=false
plssql_optimize_level=2
_column_compression_factor=175
db_4k_cache_size = 20M
```



```

-----
space_get.sql
-----
REM=====
=====
REM          Copyright (c) 1995 Oracle Corp, Redwood Shores, CA
REM
REM          OPEN SYSTEMS PERFORMANCE GROUP
REM
REM          All Rights Reserved
REM=====
=====
REM FILENAME
REM          space_get.sql
REM DESCRIPTION
REM          Get sizes of tables, indexes and tablespaces.
REM Usage: sqlplus 'sys/change_on_install as sysdba' @space_get
[<tpm> <# of warehouses>]
REM=====
=====*/

set echo on;
delete from tpcc_data;
delete from tpcc_space;
delete from tpcc_totSPACE;

insert into tpcc_data
select substr(segment_name,1,18), substr(segment_type,1,15),
sum(blocks), t.block_size,
round(sum(blocks) * 0.05), 0,
sum(blocks) + round(sum(blocks) * 0.05)
from dba_extents e, dba_tablespaces t
where owner = 'TPCC' AND ( segment_type = 'INDEX' OR
segment_type = 'INDEX PARTITION' OR segment_type =
'CLUSTER'
OR segment_type = 'TABLE' OR segment_type = 'TABLE
PARTITION')
AND e.tablespace_name <> 'SYSTEM' AND
e.tablespace_name <> 'SP_0'
AND e.tablespace_name = t.tablespace_name
group by segment_name, segment_type, t.block_size;

insert into tpcc_data
select 'SYSTEM', 'SYS', sum(blocks), t.block_size, 0, 0,
sum(blocks)
from dba_data_files f, dba_tablespaces t
where f.tablespace_name = 'SYSTEM' and t.tablespace_name
= f.tablespace_name
group by t.block_size;

insert into tpcc_data
select 'SYSAUX', 'SYS', sum(blocks), t.block_size, 0, 0,
sum(blocks)
from dba_data_files f, dba_tablespaces t
where f.tablespace_name = 'SYSAUX' and t.tablespace_name
= f.tablespace_name
group by t.block_size;

insert into tpcc_data
select 'ROLL_SEG', 'SYS', sum(blocks), t.block_size, 0, 0,
sum(blocks)
from dba_data_files f, dba_tablespaces t
where f.tablespace_name like '%UNDO_TS%' and
f.tablespace_name = t.tablespace_name
group by f.tablespace_name, t.block_size;

insert into tpcc_data
select 'DB_STAT', 'SYS', sum(blocks), t.block_size, 0, 0,
sum(blocks)
from dba_data_files f, dba_tablespaces t
where f.tablespace_name like '%SP_0%' and
f.tablespace_name = t.tablespace_name
group by f.tablespace_name, t.block_size;

update tpcc_data
set five_pct = 0,
daily_grow = round(blocks * &&1 / 62.5 / &&2),
total = blocks + round(blocks * &&1 / 62.5 / &&2)
where segment = 'HIST' OR segment = 'ORDRCLUSTER_QUEUE' OR
segment = 'IORDL';

insert into tpcc_space
select substr(ex$.name,1,18), sum(sp$.sz_blocks),
sp$.block_size, 0, 0, 0, 0
from
(select f.tablespace_name , sum(blocks) sz_blocks,
t.block_size block_size
from dba_data_files f, dba_tablespaces t
where f.tablespace_name <> 'SYSTEM' and f.tablespace_name
= t.tablespace_name
group by f.tablespace_name, t.block_size
) sp$,
(select distinct tablespace_name, segment_name name
from dba_extents
where owner = 'TPCC'
and (segment_type = 'CLUSTER' or segment_type = 'TABLE'
or segment_type = 'TABLE PARTITION' or segment_type =
'INDEX'
or segment_type = 'INDEX PARTITION')
and tablespace_name <> 'SYSTEM'
) ex$
where sp$.tablespace_name = ex$.tablespace_name
group by ex$.name, sp$.block_size;

insert into tpcc_space

```

```

select substr(f.tablespace_name,1,18), sum(blocks),
t.block_size, 0, 0, 0, 0
from dba_data_files f, dba_tablespaces t
where (f.tablespace_name = 'SYSTEM' or f.tablespace_name =
'SYSAUX')
and f.tablespace_name = t.tablespace_name
group by f.tablespace_name, t.block_size;

insert into tpcc_space
select 'ROLL_SEG', sum(blocks), t.block_size, 0, 0, 0, 0
from dba_data_files f, dba_tablespaces t
where f.tablespace_name = 'UNDO_TS' and f.tablespace_name =
t.tablespace_name
group by f.tablespace_name, t.block_size;

insert into tpcc_space
select 'DB_STAT', sum(blocks), t.block_size, 0, 0, 0, 0
from dba_data_files f, dba_tablespaces t
where f.tablespace_name = 'SP_0' and f.tablespace_name =
t.tablespace_name
group by f.tablespace_name, t.block_size;

update tpcc_space
set required =
(
select sum(total)
from tpcc_data
where tpcc_data.segment = tpcc_space.segment
)
where segment in
(
select segment from tpcc_data
);

update tpcc_space
set static =
(
select sum(total)
from tpcc_data
where tpcc_data.segment = tpcc_space.segment
)
where segment in
(
select segment from tpcc_data
);

update tpcc_space
set static = 0,
dynamic =
(
select sum(blocks)
from tpcc_data
where tpcc_data.segment = tpcc_space.segment
)
where segment in ('HIST', 'ORDRCLUSTER_QUEUE', 'IORDL');

update tpcc_space
set oversize = blocks - required;

insert into tpcc_totSPACE
select &&1, &&2, sum(static * block_size)/1024,
sum(dynamic * block_size)/1024, sum(oversize * block_size)/1024,
0, 0, 0
from tpcc_space;

update tpcc_totSPACE
set daily_grow =
(
select sum(daily_grow * block_size)/1024
from tpcc_data
);

update tpcc_totSPACE
set space60 = static + 60 * daily_grow;
set echo off;

-----
space_init.sql
-----
REM=====
=====
REM FILENAME
REM          space_init.sql
REM DESCRIPTION
REM          Create tables for space calculations.
REM Usage: sqlplus 'sys/change_on_install as sysdba'
@space_init.sql
REM=====
=====*/

set echo on;
drop table tpcc_data;
drop table tpcc_space;
drop table tpcc_totSPACE;
create table tpcc_data (
segment varchar2(18),
type varchar2(15),
blocks number,
block_size number,
five_pct number,
daily_grow number,
total number
);
create table tpcc_space (
segment varchar2(18),
blocks number,

```



```

    fprintf(stderr, "\t-m <cid#> :\tending cid number (with -
C)\n");
    fprintf(stderr, "\t-g :\tgenerate rows to standard
output\n");
    fprintf(stderr, "\t $tpcc_bench must be set to the location
of the kit\n");
    fprintf(stderr, "\n");
    exit(1);
}

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

    sprintf(realfile, "%s", fnam);
    fd = fopen(realfile, "r");
    if (!fd)
    {
        return (0);
    }
    while (fgets((char *)linebuf+nulpt, SQL_BUF_SIZE, fd))
    {
        nulpt = strlen((char *)linebuf);
    }
    return(nulpt);
}

void quit()
{
    OCIERROR(errhp, OCISessionEnd ( tpcsvc, errhp, tpcusr,
OCI_DEFAULT));
    OCIERROR(errhp, OCIserverDetach ( tpcsrv, errhp,
OCI_DEFAULT));
    OCIHandleFree((dvoid *)tpcusr, OCI_HTYPE_SESSION);
    OCIHandleFree((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX);
    OCIHandleFree((dvoid *)errhp, OCI_HTYPE_ERROR);
    OCIHandleFree((dvoid *)tpcsrv, OCI_HTYPE_SERVER);
    OCIHandleFree((dvoid *)tpcenv, OCI_HTYPE_ENV);
}

void main (argc, argv)
int argc;
char *argv[];
{
    char *uid="tpcc";
    char *pwd="tpcc";
    int scale=0;
    int i, j;
    int loop;
    int loopcount;
    int cid;
    int dwid;
    int cdid;
    int cwid;
    int sid;
    int swid;
    int olcnt;
    int nrows;
    int row;

    int w_id;
    char w_name[11];
    char w_street_1[21];
    char w_street_2[21];
    char w_city[21];
    char w_state[2];
    char w_zip[9];
    float w_tax;

    int d_id[10];
    int d_w_id[10];
    char d_name[10][11];
    char d_street_1[10][21];
    char d_street_2[10][21];
    char d_city[10][21];
    char d_state[10][2];
    char d_zip[10][9];
    float d_tax[10];

    int s_c_id;
    int s_c_d_id;
    int s_c_w_id;
    int s_c_count;

    int c_id[100];
    int c_d_id[100];
    int c_w_id[100];
    char c_first[100][17];
    char c_last[100][17];
    char c_street_1[100][21];
    char c_street_2[100][21];
    char c_city[100][21];
    char c_state[100][2];
    char c_zip[100][9];
    char c_phone[100][16];
    char c_credit[100][2];
    float c_discount[100];
    char c_data[100][501];

    int i_id[100];
    int i_im_id[100];
    int i_price[100];
    char i_name[100][25];
    char i_data[100][51];

```

```

int s_s_count;
int s_s_i_id;
int s_s_w_id;

int s_i_id[100];
int s_w_id[100];
int s_quantity[100];
char s_dist_01[100][25];
char s_dist_02[100][25];
char s_dist_03[100][25];
char s_dist_04[100][25];
char s_dist_05[100][25];
char s_dist_06[100][25];
char s_dist_07[100][25];
char s_dist_08[100][25];
char s_dist_09[100][25];
char s_dist_10[100][25];
char s_data[100][51];

int h_w_id[100];
int h_d_id[100];
int h_c_id[100];
char h_data[100][25];

int o_id[100];
int o_d_id[100];
int o_w_id[100];
int o_c_id[100];
int o_carrier_id[100];
int o_ol_cnt[100];

int ol_o_id[1500];
int ol_d_id[1500];
int ol_w_id[1500];
int ol_number[1500];
int ol_i_id[1500];
int ol_supply_w_id[1500];
int ol_amount[1500];
char ol_dist_info[1500][24];
int ol_cnt;
int ol_cnt;

ub2 ol_o_id_len[1500];
ub2 ol_d_id_len[1500];
ub2 ol_w_id_len[1500];
ub2 ol_number_len[1500];
ub2 ol_i_id_len[1500];
ub2 ol_supply_w_id_len[1500];
ub2 ol_dist_info_len[1500];
ub2 ol_amount_len[1500];

ub4 ol_o_id_clen;
ub4 ol_d_id_clen;
ub4 ol_w_id_clen;
ub4 ol_number_clen;
ub4 ol_i_id_clen;
ub4 ol_supply_w_id_clen;
ub4 ol_dist_info_clen;
ub4 ol_amount_clen;

ub2 o_id_len[100];
ub2 o_d_id_len[100];
ub2 o_w_id_len[100];
ub2 o_c_id_len[100];
ub2 o_carrier_id_len[100];
ub2 o_ol_cnt_len[100];

ub4 o_id_clen;
ub4 o_d_id_clen;
ub4 o_w_id_clen;
ub4 o_c_id_clen;
ub4 o_carrier_id_clen;
ub4 o_ol_cnt_clen;

text stmbuf[16*1024];

int no_o_id[100];
int no_d_id[100];
int no_w_id[100];

char sdate[30];

#ifdef ORA_NT
clock_t begin_time, end_time;
clock_t begin_cpu, end_cpu;

char *arg_ptr, **end_args;
#else
double begin_time, end_time;
double begin_cpu, end_cpu;
double gettime(), getcpu();

extern int getopt();
extern char *optarg;
extern int optind, opterr;
int opt;
#endif

char *argstr="M:AwdcCisShno:b:e:j:k:l:m:g";
int do_A=0;
int do_w=0;
int do_d=0;
int do_i=0;
int do_c=0;
int do_C=0;
int do_s=0;
int do_S=0;

```

```

int do_h=0;
int do_o=0;
int do_n=0;
int gen=0;
int bware=1;
int aware=0;
int bitem=1;
int eitem=0;
int bcid=1;
int ecid=0;

FILE *olfp=NULL;
char olfname[100];
char* basename;
int status;
#ifdef ORA_NT
char fname[100];
FILE *logfile;
#endif /* ORA_NT */

/*-----
+ | Parse command line -- look for scale factor. |
+-----
*/

if (argc == 1) {
    myusage ();
}

#ifdef ORA_NT
end_args = argv + argc;
for (++argv; argv < end_args; )
{
    arg_ptr = *argv++;

    if (*arg_ptr != '-')
    {
        myusage ();
    }
    else
    {
        switch (arg_ptr[1]) {
        case '?': myusage ();
            break;
        case 'M': scale = atoi (*argv++);
            break;
        case 'A': do_A = 1;
            break;
        case 'w': do_w = 1;
            break;
        case 'd': do_d = 1;
            break;
        case 'c': do_c = 1;
            break;
        case 'C': do_C = 1;
            break;
        case 'i': do_i = 1;
            break;
        case 's': do_s = 1;
            break;
        case 'S': do_S = 1;
            break;
        case 'h': do_h = 1;
            break;
        case 'n': do_n = 1;
            break;
        case 'o': do_o = 1;
            strcpy (olfname, *argv++);
            break;
        case 'b': bware = atoi (*argv++);
            break;
        case 'e': aware = atoi (*argv++);
            break;
        case 'j': bitem = atoi (*argv++);
            break;
        case 'k': eitem = atoi (*argv++);
            break;
        case 'l': bcid = atoi (*argv++);
            break;
        case 'm': ecid = atoi (*argv++);
            break;
        case 'g': gen = 1;
            strcpy (fname, *argv++);
            break;
        case 'l': logfile=fopen(*argv++,"w");
            break;
        default: fprintf (stderr, "THIS SHOULD NEVER
HAPPEN!!!\n");
            fprintf (stderr, "(reached default case in
getopt ())\n");
            myusage ();
        }
    }
}

#else

while ((opt = getopt (argc, argv, argstr)) != -1) {
    switch (opt) {
    case '?': myusage ();
        break;
    case 'M': scale = atoi (optarg);
        break;
    case 'A': do_A = 1;
        break;
    case 'w': do_w = 1;
        break;

```

```

        case 'd': do_d = 1;
            break;
        case 'c': do_c = 1;
            break;
        case 'C': do_C = 1;
            break;
        case 'i': do_i = 1;
            break;
        case 's': do_s = 1;
            break;
        case 'S': do_S = 1;
            break;
        case 'h': do_h = 1;
            break;
        case 'n': do_n = 1;
            break;
        case 'o': do_o = 1;
            strcpy (olfname, optarg);
            break;
        case 'b': bware = atoi (optarg);
            break;
        case 'e': aware = atoi (optarg);
            break;
        case 'j': bitem = atoi (optarg);
            break;
        case 'k': eitem = atoi (optarg);
            break;
        case 'l': bcid = atoi (optarg);
            break;
        case 'm': ecid = atoi (optarg);
            break;
        case 'g': gen = 1;
            break;
        default: fprintf (stderr, "THIS SHOULD NEVER
HAPPEN!!!\n");
            fprintf (stderr, "(reached default case in
getopt ())\n");
            myusage ();
    }
}

#endif /* ORA_NT */

/*-----
* | Rudimentary error checking |
*-----
*/

if (scale < 1) {
    fprintf (stderr, "Invalid scale factor: '%d'\n", scale);
    myusage ();
}

if ((do_A || do_w || do_d || do_c || do_C || do_i || do_s ||
do_S || do_h || do_o || do_n)) {
    fprintf (stderr, "What should I load???\n");
    myusage ();
}

if (gen && (do_A || (do_w + do_d + do_c + do_C + do_i + do_s
+ do_S + do_h + do_o + do_n > 1))) {
    fprintf (stderr, "Can only generate table one at a
time\n");
    myusage ();
}

if (do_S && (do_A || do_s)) {
    fprintf (stderr, "Cluster stock table around s_w_id or
s_i_id?\n");
    myusage ();
}

if (do_C && (do_A || do_c)) {
    fprintf (stderr, "Cluster cust table around c_w_id or
c_id?\n");
    myusage ();
}

if (aware <= 0)
    aware = scale;
if (ecid <= 0)
    ecid = CUSTFAC;
if (eitem <= 0)
    eitem = STOCFAC;

if (do_C) {
    if ((bcid < 1) || (bcid > CUSTFAC)) {
        fprintf (stderr, "Invalid beginning cid number:
'%d'\n", bcid);
        myusage ();
    }

    if ((ecid < bcid) || (ecid > CUSTFAC)) {
        fprintf (stderr, "Invalid ending cid number: '%d'\n",
ecid);
        myusage ();
    }
}

if (do_S) {
    if ((bitem < 1) || (bitem > STOCFAC)) {
        fprintf (stderr, "Invalid beginning item number:
'%d'\n", bitem);
        myusage ();
    }
}

```

```

        if ((eitem < bitem) || (eitem > STOCFAC)) {
            fprintf (stderr, "Invalid ending item number: '%d'\n",
item);
            myusage ();
        }
        if (do_o) {
            if ((basename = getenv ("tpcc_bench")) == NULL)
            {
                fprintf (stderr, "$tpcc_bench is not set");
                myusage ();
            }
        }
        if ((bware < 1) || (bware > scale)) {
            fprintf (stderr, "Invalid beginning warehouse number:
'd'\n", bware);
            myusage ();
        }
        if ((eware < bware) || (eware > scale)) {
            fprintf (stderr, "Invalid ending warehouse number:
'd'\n", eware);
            myusage ();
        }
        if (gen && do_o) {
            if ((olfp = fopen (olfname, "w")) == NULL) {
                fprintf (stderr, "Can't open '%s' for writing order
lines\n", olfname);
                myusage ();
            }
        }
    }
    /*-----
    + | Prepare to insert into database. |
    +-----*/
    sysdate (sdate);
    if (!gen) {
        /* log on to Oracle */
        OCIInitialize(OCI_DEFAULT|OCI_OBJECT,(dvoid *)0,0,0,0);
        OCIEnvInit(&tpcenv, OCI_DEFAULT, 0, (dvoid **)0);
        OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsrv,
OCI_HTYPE_SERVER, 0, (dvoid **)0);
        OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&errhp,
OCI_HTYPE_ERROR, 0, (dvoid **)0);
        OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsvc,
OCI_HTYPE_SVCCTX, 0, (dvoid **)0);
        OCIAttach(tpcsrv, errhp, (text *)0,0,OCI_DEFAULT);
        OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX, (dvoid
*)tpcsrv,
            (ub4)0,OCI_ATTR_SERVER, errhp);
        OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr,
OCI_HTYPE_SESSION, 0, (dvoid **)0);
        OCIAttrSet((dvoid *)tpcsrv, OCI_HTYPE_SESSION, (dvoid
*)uid,
            (ub4)strlen(uid),OCI_ATTR_USERNAME, errhp);
        OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid
*)pwd, (ub4)strlen(pwd),
            OCI_ATTR_PASSWORD, errhp);
        OCIERROR(errhp, OCISESSIONBegin(tpcsvc, errhp, tpcusr,
OCI_CRED_RDBMS, OCI_DEFAULT));
        OCIAttrSet(tpcsvc, OCI_HTYPE_SVCCTX, tpcusr, 0,
OCI_ATTR_SESSION, errhp);
        fprintf (stderr, "\nConnected to Oracle userid
'%s/%s'\n", uid, pwd);
        /* open cursors and parse statement */
        if (do_A || do_w) {
            OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)&curw),
OCI_HTYPE_STMT, 0, (dvoid **)0);
            OCIERROR(errhp,OCIStmtPrepare(curw, errhp, (text
*)SQLTXTW,
                strlen((char *)SQLTXTW), (ub4)
OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
        }
        if (do_A || do_d) {
            OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)&curd),
OCI_HTYPE_STMT, 0, (dvoid **)0);
            OCIERROR(errhp,OCIStmtPrepare(curd, errhp, (text
*)SQLXTD,
                strlen((char *)SQLXTD), (ub4)
OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
        }
        if (do_A || do_c || do_C) {
            OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)&curc),
OCI_HTYPE_STMT, 0, (dvoid **)0);
            OCIERROR(errhp,OCIStmtPrepare(curc, errhp, (text
*)SQLTXTC,
                strlen((char *)SQLTXTC), (ub4)
OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
            OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid
**)(&curcs), OCI_HTYPE_STMT, 0, (dvoid **)0);
            OCIERROR(errhp,OCIStmtPrepare(curcs, errhp, (text
*)SQLTXTCQUERY,
                strlen((char *)SQLTXTCQUERY), (ub4)
OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));

```

```

        }
        if (do_A || do_h) {
            OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)&curh),
OCI_HTYPE_STMT, 0, (dvoid **)0);
            OCIERROR(errhp,OCIStmtPrepare(curh, errhp, (text
*)SQLTXTH,
                strlen((char *)SQLTXTH), (ub4)
OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
        }
        if (do_A || do_s || do_S) {
            OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)&curS),
OCI_HTYPE_STMT, 0, (dvoid **)0);
            OCIERROR(errhp,OCIStmtPrepare(curS, errhp, (text
*)SQLXTS,
                strlen((char *)SQLXTS), (ub4)
OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
            OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid
**)(&curss), OCI_HTYPE_STMT, 0, (dvoid **)0);
            OCIERROR(errhp,OCIStmtPrepare(curss, errhp, (text
*)SQLXTXSQUERY,
                strlen((char *)SQLXTXSQUERY), (ub4)
OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
        }
        if (do_A || do_i) {
            OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)&curi),
OCI_HTYPE_STMT, 0, (dvoid **)0);
            OCIERROR(errhp,OCIStmtPrepare(curI, errhp, (text
*)SQLXTXI,
                strlen((char *)SQLXTXI), (ub4)
OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
        }
        if (do_A || do_o) {
            int stat;
            char fname[160];
            OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid
**)(&curol), OCI_HTYPE_STMT, 0, (dvoid **)0);
            DISCARD strcpy(fname,basename);
            DISCARD strcat(fname, "/");
            DISCARD strcat(fname,
"benchrun/blocks/load_ordordl.sql");
            stat = sqlfile(fname, stmbuf);
            if (!stat)
            {
                fprintf (stderr, "unable to open %s \n",fname);
                quit();
                exit(1);
            }
            OCIERROR(errhp,OCIStmtPrepare(curo1, errhp, stmbuf,
                strlen((char *)stmbuf), (ub4)
OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
        }
        if (do_A || do_n) {
            OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid
**)(&curno), OCI_HTYPE_STMT, 0, (dvoid **)0);
            OCIERROR(errhp,OCIStmtPrepare(curno, errhp, (text
*)SQLXTNO,
                strlen((char *)SQLXTNO), (ub4)
OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
        }
        /* bind variables */
        /* warehouse */
        if (do_A || do_w) {
            OCIERROR(errhp, OCIBindByName(curw, &w_id_bp, errhp,
(text *)("w_id"), strlen("w_id"),
                (ub1 *)&w_id, sizeof(w_id), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
            OCIERROR(errhp, OCIBindByName(curw, &w_name_bp,
errhp,(text *)"w_name", strlen("w_name"),
                (ub1 *)w_name, 11, SQLT_STR, (dvoid *) 0,
(ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
            OCIERROR(errhp, OCIBindByName(curw, &w_street1_bp,
errhp, (text *)"w_street_1",
                strlen("w_street_1"), (ub1 *)w_street_1, 21,
SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
            OCIERROR(errhp, OCIBindByName(curw, &w_street2_bp,
errhp, (text *)"w_street_2",
                strlen("w_street_2"), (ub1 *)w_street_2, 21,
SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
            OCIERROR(errhp, OCIBindByName(curw, &w_city_bp, errhp,
(text *)"w_city",
                strlen("w_city"), (ub1 *)w_city, 21, SQLT_STR,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
            OCIERROR(errhp, OCIBindByName(curw, &w_state_bp, errhp,
(text *)"w_state",
                strlen("w_state"), (ub1 *)w_state, 2, SQLT_CHR,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

```

```

OCIERROR(errhp, OCIBindByName(curw, &w_zip_bp, errhp,
(text *)":w_zip",
    strlen(":w_zip"), (ub1 *)w_zip, 9, SQLT_CHR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curw, &w_tax_bp, errhp,
(text *)":w_tax",
    strlen(":w_tax"), (ub1 *) &w_tax, sizeof(w_tax),
SQLT_FLT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
}
/* district */
if (do_A || do_d) {
OCIERROR(errhp, OCIBindByName(curd, &d_id_bp, errhp,
(text *)":d_id",
    strlen(":d_id"), (ub1 *)d_id, sizeof(int), SQLT_INT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curd, &d_w_id_bp, errhp,
(text *)":d_w_id",
    strlen(":d_w_id"), (ub1 *)d_w_id, sizeof(int),
SQLT_INT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curd, &d_name_bp, errhp,
(text *)":d_name",
    strlen(":d_name"), (ub1 *)d_name, 11, SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curd, &d_street1_bp,
errhp, (text *)":d_street_1",
    strlen(":d_street_1"), (ub1 *)d_street_1, 21,
SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curd, &d_street2_bp,
errhp, (text *)":d_street_2",
    strlen(":d_street_2"), (ub1 *)d_street_2, 21,
SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curd, &d_city_bp, errhp,
(text *)":d_city",
    strlen(":d_city"), (ub1 *)d_city, 21, SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curd, &d_state_bp, errhp,
(text *)":d_state",
    strlen(":d_state"), (ub1 *)d_state, 2,
SQLT_CHR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curd, &d_zip_bp, errhp,
(text *)":d_zip",
    strlen(":d_zip"), (ub1 *)d_zip, 9,
SQLT_CHR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curd, &d_tax_bp, errhp,
(text *)":d_tax",
    strlen(":d_tax"), (ub1 *)d_tax,
sizeof(float), SQLT_FLT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
}
/* customer */
if (do_A || do_c || do_i) {
OCIERROR(errhp, OCIBindByName(curcs, &s_c_id_bp, errhp,
(text *)":s_c_id",
    strlen(":s_c_id"), (ub1 *)&s_c_id,
sizeof(int), SQLT_INT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curcs, &s_c_w_id_bp,
errhp, (text *)":s_c_w_id",
    strlen(":s_c_w_id"), (ub1 *)&s_c_w_id,
sizeof(int), SQLT_INT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curcs, &s_c_d_id_bp,
errhp, (text *)":s_c_d_id",
    strlen(":s_c_d_id"), (ub1 *)&s_c_d_id,
sizeof(int), SQLT_INT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIDefineByPos(curcs, &s_c_ret_bp, errhp, 1, &s_c_count, sizeof(int),
SQLT_INT,
    0, 0, 0, OCI_DEFAULT);

```

```

OCIERROR(errhp, OCIBindByName(curc, &c_id_bp, errhp,
(text *)":c_id",
    strlen(":c_id"), (ub1 *)c_id, sizeof(int),
SQLT_INT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curc, &c_d_id_bp, errhp,
(text *)":c_d_id",
    strlen(":c_d_id"), (ub1 *)c_d_id,
sizeof(int), SQLT_INT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curc, &c_w_id_bp, errhp,
(text *)":c_w_id",
    strlen(":c_w_id"), (ub1 *)c_w_id,
sizeof(int), SQLT_INT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curc, &c_first_bp, errhp,
(text *)":c_first",
    strlen(":c_first"), (ub1 *)c_first, 17, SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curc, &c_last_bp, errhp,
(text *)":c_last",
    strlen(":c_last"), (ub1 *)c_last, 17, SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curc, &c_street1_bp,
errhp, (text *)":c_street_1",
    strlen(":c_street_1"), (ub1 *)c_street_1,
21, SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curc, &c_street2_bp,
errhp, (text *)":c_street_2",
    strlen(":c_street_2"), (ub1 *)c_street_2,
21, SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curc, &c_city_bp, errhp,
(text *)":c_city",
    strlen(":c_city"), (ub1 *)c_city, 21,
SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curc, &c_state_bp, errhp,
(text *)":c_state",
    strlen(":c_state"), (ub1 *)c_state, 2,
SQLT_CHR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curc, &c_zip_bp, errhp,
(text *)":c_zip",
    strlen(":c_zip"), (ub1 *)c_zip, 9,
SQLT_CHR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curc, &c_phone_bp, errhp,
(text *)":c_phone",
    strlen(":c_phone"), (ub1 *)c_phone, 16,
SQLT_CHR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curc, &c_credit_bp,
errhp, (text *)":c_credit",
    strlen(":c_credit"), (ub1 *)c_credit, 2,
SQLT_CHR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curc, &c_discount_bp,
errhp, (text *)":c_discount",
    strlen(":c_discount"), (ub1 *)c_discount,
sizeof(float), SQLT_FLT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curc, &c_data_bp, errhp,
(text *)":c_data",
    strlen(":c_data"), (ub1 *)c_data, 501,
SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
}
/* item */
if (do_A || do_i) {
OCIERROR(errhp, OCIBindByName(curi, &i_id_bp, errhp,
(text *)":i_id",
    strlen(":i_id"), (ub1 *)i_id, sizeof(int),
SQLT_INT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

```

```

OCIERROR(errhp, OCIBindByName(curs, &i_im_id_bp, errhp,
(text *)":i_im_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &i_name_bp, errhp,
(text *)":i_name",
sizeof(int), SQLT_STR,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &i_price_bp, errhp,
(text *)":i_price",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &i_data_bp, errhp,
(text *)":i_data",
sizeof(int), SQLT_STR,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
}
/* stock */
if (do_A || do_s || do_S) {
OCIERROR(errhp, OCIBindByName(curs, &s_s_i_id_bp,
errhp, (text *)":s_s_i_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &s_s_w_id_bp,
errhp, (text *)":s_s_w_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIDefineByPos(curs, &s_s_ret_bp, errhp, 1, &s_s_count, sizeof(int),
SQLT_INT,
0, 0, 0, OCI_DEFAULT);
OCIERROR(errhp, OCIBindByName(curs, &s_i_id_bp, errhp,
(text *)":s_i_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &s_w_id_bp, errhp,
(text *)":s_w_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &s_quantity_bp,
errhp, (text *)":s_quantity",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &s_dist_01_bp,
errhp, (text *)":s_dist_01",
24, SQLT_CHR,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &s_dist_02_bp,
errhp, (text *)":s_dist_02",
24, SQLT_CHR,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &s_dist_03_bp,
errhp, (text *)":s_dist_03",
24, SQLT_CHR,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &s_dist_04_bp,
errhp, (text *)":s_dist_04",
24, SQLT_CHR,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &s_dist_05_bp,
errhp, (text *)":s_dist_05",
24, SQLT_CHR,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &s_dist_06_bp,
errhp, (text *)":s_dist_06",
24, SQLT_CHR,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &s_dist_07_bp,
errhp, (text *)":s_dist_07",
24, SQLT_CHR,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &s_dist_08_bp,
errhp, (text *)":s_dist_08",
24, SQLT_CHR,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &s_dist_09_bp,
errhp, (text *)":s_dist_09",
24, SQLT_CHR,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &s_dist_10_bp,
errhp, (text *)":s_dist_10",
24, SQLT_CHR,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &s_data_bp, errhp,
(text *)":s_data",
sizeof(int), SQLT_STR,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
}
/* history */
if (do_A || do_h) {
OCIERROR(errhp, OCIBindByName(curs, &h_c_id_bp, errhp,
(text *)":h_c_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &h_c_d_id_bp,
errhp, (text *)":h_c_d_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &h_c_w_id_bp,
errhp, (text *)":h_c_w_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &h_d_id_bp, errhp,
(text *)":h_d_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &h_w_id_bp, errhp,
(text *)":h_w_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIBindByName(curs, &h_data_bp, errhp,
(text *)":h_data",
sizeof(int), SQLT_STR,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4 *) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
}
/* order and order_line (delivered) */
if (do_A || do_o) {
for (i = 0; i < ORDEARR; i++) {
o_id_len[i] = sizeof(int);
o_d_id_len[i] = sizeof(int);
o_w_id_len[i] = sizeof(int);
o_c_id_len[i] = sizeof(int);
o_carrier_id_len[i] = sizeof(int);
o_ol_cnt_len[i] = sizeof(int);
}
}

```



```

    }
    OCIERROR(errhp, OCIBindByName(curo1, &o1_o_id_bp,
errhp, (text *)":o1_o_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)o1_o_id_len, (ub2 *)0,
(ub4) OCI_DEFAULT));
    OCIERROR(errhp, OCIBindByName(curo1, &o1_d_id_bp,
errhp, (text *)":o1_d_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)o1_d_id_len, (ub2 *)0,
(ub4) OCI_DEFAULT));
    OCIERROR(errhp, OCIBindByName(curo1, &o1_w_id_bp,
errhp, (text *)":o1_w_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)o1_w_id_len, (ub2 *)0,
(ub4) OCI_DEFAULT));
    OCIERROR(errhp, OCIBindByName(curo1, &o1_number_bp,
errhp, (text *)":o1_number",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)o1_number_len, (ub2
*)0,
(ub4) OCI_DEFAULT));
    OCIERROR(errhp, OCIBindByName(curo1, &o1_i_id_bp,
errhp, (text *)":o1_i_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)o1_i_id_len, (ub2 *)0,
(ub4) OCI_DEFAULT));
    OCIERROR(errhp, OCIBindByName(curo1,
&o1_supply_w_id_bp, errhp, (text *)":o1_supply_w_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)o1_supply_w_id_len,
(ub2 *)0,
(ub4) OCI_DEFAULT));
    OCIERROR(errhp, OCIBindByName(curo1, &o1_dist_info_bp,
errhp, (text *)":o1_dist_info",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)o1_dist_info_len, (ub2
*)0,
(ub4) OCI_DEFAULT));
    OCIERROR(errhp, OCIBindByName(curo1, &o1_amount_bp,
errhp, (text *)":o1_amount",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)o1_amount_len, (ub2
*)0,
(ub4) OCI_DEFAULT));
    OCIERROR(errhp, OCIBindByName(curo1, &o_id_bp, errhp,
(text *)":o_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)o_id_len, (ub2 *)0,
(ub4) OCI_DEFAULT));
    OCIERROR(errhp, OCIBindByName(curo1, &o_d_id_bp, errhp,
(text *)":o_d_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)o_d_id_len, (ub2 *)0,
(ub4) OCI_DEFAULT));
    OCIERROR(errhp, OCIBindByName(curo1, &o_w_id_bp, errhp,
(text *)":o_w_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)o_w_id_len, (ub2 *)0,
(ub4) OCI_DEFAULT));
    OCIERROR(errhp, OCIBindByName(curo1, &o_c_id_bp, errhp,
(text *)":o_c_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)o_c_id_len, (ub2 *)0,
(ub4) OCI_DEFAULT));
    OCIERROR(errhp, OCIBindByName(curo1, &o_carrier_id_bp,
errhp, (text *)":o_carrier_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)o_carrier_id_len, (ub2
*)0,

```

```

(ub4) OCI_DEFAULT));
    OCIERROR(errhp, OCIBindByName(curo1, &o_ol_cnt_bp,
errhp, (text *)":o_ol_cnt",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)o_ol_cnt_len, (ub2
*)0,
(ub4) OCI_DEFAULT));
    OCIERROR(errhp, OCIBindByName(curo1, &o_cnt_bp, errhp,
(text *)":order_rows",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4) OCI_DEFAULT));
    OCIERROR(errhp, OCIBindByName(curo1, &o_olcnt_bp,
errhp, (text *)":ordl_rows",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4) OCI_DEFAULT));
}
/* new order */
if (do_A || do_n) {
    OCIERROR(errhp, OCIBindByName(curno, &no_o_id_bp,
errhp, (text *)":no_o_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4) OCI_DEFAULT));
    OCIERROR(errhp, OCIBindByName(curno, &no_d_id_bp,
errhp, (text *)":no_d_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4) OCI_DEFAULT));
    OCIERROR(errhp, OCIBindByName(curno, &no_w_id_bp,
errhp, (text *)":no_w_id",
sizeof(int), SQLT_INT,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4) OCI_DEFAULT));
}
}
/*-----
+
| Initialize random number generator |
+-----*/
srand (SEED);
#ifdef ORANT
srand48 (SEED);
#endif
initperm ();
/*-----
+
| Load the WAREHOUSE table. |
+-----*/
if (do_A || do_w) {
    nrows = aware - bware + 1;
    fprintf (stderr, "Loading/generating warehouse: %d - %d
(%d rows)\n",
bware, aware, nrows);
    begin_time = gettime ();
    begin_cpu = getcpu ();
    for (loop = bware; loop <= aware; loop++) {
        w_tax = (float) ((lrand48 () % 2001) * 0.0001);
        randstr (w_name, 6, 10);
        randstr (w_street_1, 10, 20);
        randstr (w_street_2, 10, 20);
        randstr (w_city, 10, 20);
        randstr (str2, 2, 2);
        randnum (num9, 9);
        num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';
        if (gen) {
            printf ("%d 30000000 %6.4f %s %s %s %s %s %s\n",
loop, w_tax,
w_name, w_street_1, w_street_2, w_city,
str2, num9);
            fflush (stdout);
        }
        else {
            w_id = loop;
            strncpy (w_state, str2, 2);
            strncpy (w_zip, num9, 9);
            status = OCIStmtExecute(tpcsvc, curw, errhp, (ub4) 1,
(ub4) 0,

```

```

0, (CONST OCISnapshot*) 0, (OCISnapshot*)
OCI_COMMIT_ON_SUCCESS);
    if (status != OCI_SUCCESS) {
        fprintf (stderr, "Error at ware %d\n", loop);
        OCIERROR(errhp, status);
        quit ();
        exit (1);
    }
}
    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done. %d rows loaded/generated in
%10.2f sec. (%10.2f cpu)\n\n",
nrows, end_time - begin_time, end_cpu -
begin_cpu);
/*-----
+ | Load the DISTRICT table. |
+-----*/
    if (do_A || do_d) {
        nrows = (eware - bware + 1) * DISTFAC;
        fprintf (stderr, "Loading/generating district: w%d - w%d
(%d rows)\n",
bware, eware, nrows);
        begin_time = gettime ();
        begin_cpu = getcpu ();
        dwid = bware - 1;
        for (row = 0; row < nrows; ) {
            dwid++;
            for (i = 0; i < DISTARR; i++, row++) {
                d_tax[i] = (float) ((lrand48 () % 2001) * 0.0001);
                randstr (d_name[i], 6, 10);
                randstr (d_street_1[i], 10, 20);
                randstr (d_street_2[i], 10, 20);
                randstr (d_city[i], 10, 20);
                randstr (str2, 2, 2);
                randnum (num9, 9);
                num9[4] = num9[5] = num9[6] = num9[7] = num9[8] =
'1';
                if (gen) {
                    printf ("%d %d 3000000 %6.4f 3001 %s %s %s %s %s
%s\n",
i + 1, dwid, d_tax[i], d_name[i],
d_street_1[i],
d_street_2[i], d_city[i], str2, num9 );
                }
                else {
                    d_id[i] = i + 1;
                    d_w_id[i] = dwid;
                    strncpy (d_state[i], str2, 2);
                    strncpy (d_zip[i], num9, 9);
                }
            }
            if (gen) {
                fflush (stdout);
            }
            else {
                status = OCIStmtExecute(tpcsvc, curd, errhp, (ub4)
DISTARR, (ub4) 0,
(CONST OCISnapshot*) 0, (OCISnapshot*)
0,
(ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
                if (status != OCI_SUCCESS) {
                    fprintf (stderr, "Aborted at ware %d, dist 1\n", dwid);
                    OCIERROR(errhp, status);
                    quit ();
                    exit (1);
                }
            }
        }
        end_time = gettime ();
        end_cpu = getcpu ();
        fprintf (stderr, "Done. %d rows loaded/generated in
%10.2f sec. (%10.2f cpu)\n\n",
nrows, end_time - begin_time, end_cpu -
begin_cpu);
/*-----
+ | Load the CUSTOMER table. |
+-----*/
    if (do_A || do_c) {
        nrows = (eware - bware + 1) * CUSTFAC * DISTFAC;
        fprintf (stderr, "Loading/generating customer: w%d - w%d
(%d rows)\n",
bware, eware, nrows);

```

```

    if (getenv("tpcc_hash_overflow")) {
        fprintf(stderr, "Hash overflow is enabled\n");
        OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT,
0, (dvoid**)0);
        sprintf ((char *) stmbuf, SQLXTENHA);
        OCIStmtPrepare(curi, errhp, stmbuf, strlen((char
*)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);
        OCIERROR(errhp,OCIStmtExecute(tpcsvc, curi,
errhp,1,0,0,0,OCI_DEFAULT));
        OCIHandleFree(curi, OCI_HTYPE_STMT);
        fprintf (stderr, "Customer loaded for horizontal
partitioning\n");
    }
    else
    {
        fprintf (stderr, "Customer not loaded for horizontal
partitioning\n");
    }
    begin_time = gettime ();
    begin_cpu = getcpu ();
    s_c_id = 1;
    s_c_d_id = 1;
    s_c_w_id = bware;
    while (s_c_w_id <= eware) {
        status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4) 1,
(ub4) 0,
(CONST OCISnapshot*) 0, (OCISnapshot*) 0,
(ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
            OCIERROR(errhp, status);
            quit ();
            exit (1);
        }
        if (s_c_count == 0) {
            s_c_w_id--;
            break;
        }
        else s_c_w_id++;
    }
    if (s_c_w_id < bware ) s_c_w_id = bware;
    else {
        if (s_c_w_id > eware ) s_c_w_id = eware;
        while (s_c_d_id <= DISTFAC) {
            status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4)
1, (ub4) 0,
(CONST OCISnapshot*) 0, (OCISnapshot*) 0,
(ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                fprintf (stderr, "Select failed\n");
                OCIERROR(errhp, status);
                quit ();
                exit (1);
            }
            if (s_c_count == 0) {
                s_c_d_id--;
                break;
            }
            else s_c_d_id++;
        }
        if (s_c_d_id > DISTFAC) s_c_d_id = DISTFAC;
        while (s_c_id <= CUSTFAC) {
            status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4)
1, (ub4) 0,
(CONST OCISnapshot*) 0, (OCISnapshot*) 0,
(ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                OCIERROR(errhp, status);
                quit ();
                exit (1);
            }
            if (s_c_count == 0) break;
            else s_c_id++;
        }
    }
    if (s_c_id > CUSTFAC) {
        s_c_d_id=1;
        s_c_w_id++;
        s_c_id=1;
    }
    fprintf (stderr, "start at wid: %d, did: %d, cid: %d\n",
s_c_w_id, s_c_d_id, s_c_id);
    cid = s_c_id - 1;
    cdid = s_c_d_id;
    cwid = s_c_w_id;
    nrows = (eware - s_c_w_id + 1) * DISTFAC * CUSTFAC -
(s_c_d_id - 1) * CUSTFAC - s_c_id + 1;
    fprintf (stderr, "remaining rows: %d\n", nrows);
    loopcount = 0;
    for (row = 0; row < nrows; ) {
        for (i = 0; i < CUSTARR && row < nrows; i++, row++) {
            cid++;
            if (cid > CUSTFAC) {
                /* cycle cust id */
                cid = 1;
                /* cheap mod */
                cdid++;
                /* shift dist cycle */
            }
            if (cdid > DISTFAC) {
                cdid = 1;
                /* shift ware cycle */
                cwid++;
            }
        }
    }

```

```

c_id[i] = cid;
c_d_id[i] = cdid;
c_w_id[i] = cwid;
if (cid <= 1000)
    randlastname (c_last[i], cid - 1);
else
    randlastname (c_last[i], NURand (255, 0, 999,
CNUM1));
c_credit[i][1] = 'C';
if (lrand48 () % 10)
    c_credit[i][0] = 'G';
else
    c_credit[i][0] = 'B';
c_discount[i] = (float)((lrand48 () % 5001) *
0.0001);
randstr (c_first[i], 8, 16);
randstr (c_street_1[i], 10, 20);
randstr (c_street_2[i], 10, 20);
randstr (c_city[i], 10, 20);
randstr (str2, 2, 2);
randnum (num9, 9);
num9[4] = num9[5] = num9[6] = num9[7] = num9[8] =
'1';
randnum (num16, 16);
randstr (c_data[i], 300, 500);
if (gen) {
    printf ("%d %d %d %s OE %s %s %s %s %s %s %s %s\n",
%c 5000000 %6.4f -1000 1000 1 0 %s\n",
        cid, cdid, cwid, c_first[i], c_last[i],
str2, num9,
        num16, sdate, c_credit[i][0],
c_discount[i], c_data[i]);
    } else {
        strncpy (c_state[i], str2, 2);
        strncpy (c_zip[i], num9, 9);
        strncpy (c_phone[i], num16, 16);
    }
}
if (gen) {
    fflush (stdout);
} else {
    status = OCISmtExecute(tpcsvc, curc, errhp, (ub4)
i, (ub4) 0,
        (CONST OCISnapshot*) 0, (OCISnapshot*)
0,
        (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
    if (status != OCI_SUCCESS) {
        fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
            c_w_id[0], c_d_id[0], c_id[0]);
        OCIERROR(errhp, status);
        quit ();
        exit (1);
    }
}
if ((++loopcount) % 50)
    fprintf (stderr, ".");
else
    fprintf (stderr, " %d rows committed\n ", row);
}
end_time = gettime ();
end_cpu = getcpu ();
fprintf (stderr, "Done. %d rows loaded/generated in
%10.2f sec. (%10.2f cpu)\n\n",
    nrows < 0 ? 0 : nrows, end_time - begin_time,
end_cpu - begin_cpu);
if (getenv("tpcc_hash_overflow")) {
    fprintf(stderr, "Hash overflow is disabled\n");
    OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT,
0, (dvoid**)0);
    sprintf ((char *) stmbuf, SQLTXTHIHA);
    OCISmtPrepare(curi, errhp, stmbuf, strlen((char
*)stmbuf),
        OCI_NTV_SYNTAX, OCI_DEFAULT);
    OCIERROR(errhp, OCISmtExecute(tpcsvc, curi,
errhp, 1, 0, 0, 0, OCI_DEFAULT));
    OCIHandleFree(curi, OCI_HTYPE_STMT);
}
}
}
/*-----
+
| Load the CUSTOMER table (cluster around c_id) |
+-----
*/
if (do_C) {
    /* start each loader off on a different seed */
    srand (bcid);
#ifdef ORA_NT
    srand48 (bcid);
#endif
    nrows = (ecid - bcid + 1) * (eware - bware + 1) * DISTFAC;
    fprintf (stderr, "Loading/generating customer: c%d - c%d,
w%d - w%d (%d rows)\n ",
        bcid, ecid, bware, aware, nrows);

```

```

if (getenv("tpcc_hash_overflow")) {
    fprintf(stderr, "Hash overflow is enabled\n");
    OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT,
0, (dvoid**)0);
    sprintf ((char *) stmbuf, SQLXTENHA);
    OCISmtPrepare(curi, errhp, stmbuf, strlen((char
*)stmbuf),
        OCI_NTV_SYNTAX, OCI_DEFAULT);
    OCIERROR(errhp, OCISmtExecute(tpcsvc, curi,
errhp, 1, 0, 0, 0, OCI_DEFAULT));
    OCIHandleFree(curi, OCI_HTYPE_STMT);
    fprintf (stderr, "Customer loaded for horizontal
partitioning\n");
} else
{
    fprintf (stderr, "Customer not loaded for horizontal
partitioning\n");
}
begin_time = gettime ();
begin_cpu = getcpu ();
s_c_id = bcid;
s_c_d_id = 1;
s_c_w_id = bware;
while (s_c_id <= ecid) {
    status = OCISmtExecute(tpcsvc, curcs, errhp, (ub4) 1,
(ub4) 0,
        (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
(ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
    if (status != OCI_SUCCESS) {
        OCIERROR(errhp, status);
        quit ();
        exit (1);
    }
    if (s_c_count == 0) {
        s_c_id--;
        break;
    }
    else s_c_id++;
}
if (s_c_id < bcid) s_c_id = bcid;
else
{
    if (s_c_id > ecid) s_c_id = ecid;
    while (s_c_w_id <= aware) {
        status = OCISmtExecute(tpcsvc, curcs, errhp, (ub4)
1, (ub4) 0,
            (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
(ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
            fprintf (stderr, "Select failed\n");
            OCIERROR(errhp, status);
            quit ();
            exit (1);
        }
        if (s_c_count == 0) {
            s_c_w_id--;
            break;
        }
        else s_c_w_id++;
    }
    if (s_c_w_id > aware) s_c_w_id = aware;
    else if (s_c_w_id < bware) s_c_w_id = bware;
}
while (s_c_d_id <= DISTFAC) {
    status = OCISmtExecute(tpcsvc, curcs, errhp, (ub4)
1, (ub4) 0,
        (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
(ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
    if (status != OCI_SUCCESS) {
        OCIERROR(errhp, status);
        quit ();
        exit (1);
    }
    if (s_c_count == 0) break;
    else s_c_d_id++;
}
}
if (s_c_d_id > DISTFAC) {
    s_c_d_id = 1;
    s_c_w_id = bware;
    s_c_id++;
}
}
fprintf (stderr, "start at cid: %d, wid: %d, did: %d\n
", s_c_id, s_c_w_id, s_c_d_id);
cid = s_c_id;
cdid = s_c_d_id - 1;
cwid = s_c_w_id;
nrows = (ecid - s_c_id + 1) * (aware - bware + 1) *
DISTFAC - (s_c_w_id - 1) * DISTFAC - s_c_d_id + 1;
fprintf (stderr, "remaining rows: %d\n ", nrows);
loopcount = 0;
for (row = 0; row < nrows; ) {
    for (i = 0; i < CUSTARR && row < nrows; i++, row++) {
        cdid++;
        if (cdid > DISTFAC) {
            cdid = 1;
            cwid++;
            if (cwid > aware) {
                cwid = bware;
                /* shift ware
cycle */

```

```

        cid++;
    }
    c_id[i] = cid;
    c_d_id[i] = cdid;
    c_w_id[i] = cwid;
    if (cid <= 1000)
        randlastname (c_last[i], cid - 1);
    else
        randlastname (c_last[i], NURand (255, 0, 999,
CNUM1));
    c_credit[i][1] = 'C';
    if (lrand48 () % 10)
        c_credit[i][0] = 'G';
    else
        c_credit[i][0] = 'B';
    c_discount[i] = (float)((lrand48 () % 5001) *
0.0001);
    randstr (c_first[i], 8, 16);
    randstr (c_street_1[i], 10, 20);
    randstr (c_street_2[i], 10, 20);
    randstr (c_city[i], 10, 20);
    randstr (str2, 2, 2);
    randnum (num9, 9);
    num9[4] = num9[5] = num9[6] = num9[7] = num9[8] =
'1';
    randnum (num16, 16);
    randstr (c_data[i], 300, 500);

    if (gen) {
        printf ("%d %d %d %s OE %s %s %s %s %s %s %s %s %s\n",
%c 5000000 %6.4f -1000 1000 1 0 %s\n",
            cid, cdid, cwid, c_first[i], c_last[i],
str2, num9,
            num16, sdate, c_credit[i][0],
c_discount[i], c_data[i]);
    }
    else {
        strncpy (c_state[i], str2, 2);
        strncpy (c_zip[i], num9, 9);
        strncpy (c_phone[i], num16, 16);
    }
}

if (gen) {
    fflush (stdout);
}
else {
    status = OCISmtExecute(tpcsvc, curc, errhp, (ub4)
i, (ub4) 0,
                                (CONST OCISnapshot*) 0, (OCISnapshot*)
0,
                                (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);

    if (status != OCI_SUCCESS) {
        fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
                c_w_id[0], c_d_id[0], c_id[0]);
        OCIERROR(errhp, status);
        quit ();
        exit (1);
    }
}

if ((++loopcount) % 50)
    fprintf (stderr, ".");
else
    fprintf (stderr, " %d rows committed\n ", row);

end_time = gettime ();
end_cpu = getcpu ();
fprintf (stderr, "Done. %d rows loaded/generated in
%10.2f sec. (%10.2f cpu)\n\n",
        nrows, end_time - begin_time,
end_cpu - begin_cpu);
if (getenv("tpcc_hash_overflow")) {
    fprintf(stderr, "Hash overflow is disabled\n");
    OCIHandleAlloc(tpcenv, (dvoid **)&scuri, OCI_HTYPE_STMT,
0, (dvoid**)0);
    sprintf ((char *) stmbuf, SQLTXTHA);
    OCIStmtPrepare(curc, errhp, stmbuf, strlen((char
*)stmbuf),
                                OCI_NTV_SYNTAX, OCI_DEFAULT);
    OCIERROR(errhp, OCIStmtExecute(tpcsvc, curc,
errhp, 1, 0, 0, OCI_DEFAULT));
    OCIHandleFree(curc, OCI_HTYPE_STMT);
}
}

/*-----
+
| Load the ITEM table.
+-----
*/

if (do_A || do_i) {
    nrows = ITEMFACT;

    fprintf (stderr, "Loading/generating item: (%d rows)\n",
nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    loopcount = 0;

```

```

for (row = 0; row < nrows; ) {
    for (i = 0; i < ITEMARR; i++, row++) {
        i_im_id[i] = (lrand48 () % 10000) + 1;
        i_price[i] = ((lrand48 () % 9901) + 100);
        randstr (i_name[i], 14, 24);
        randdatastr (i_data[i], 26, 50);

        if (gen) {
            printf ("%d %d %s %d %s\n", row + 1, i_im_id[i],
i_name[i],
                                i_price[i], i_data[i]);
        }
        else {
            i_id[i] = row + 1;
        }
    }

    if (gen) {
        fflush (stdout);
    }
    else {
        status = OCISmtExecute(tpcsvc, curi, errhp, (ub4)
ITEMARR, (ub4) 0,
                                (CONST OCISnapshot*) 0, (OCISnapshot*)
0,
                                (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
            fprintf (stderr, "Aborted at i_id %d\n", i_id[0]);
            OCIERROR(errhp, status);
            quit ();
            exit (1);
        }
    }

    if ((++loopcount) % 50)
        fprintf (stderr, ".");
    else
        fprintf (stderr, " %d rows committed\n ", row);

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done. %d rows loaded/generated in
%10.2f sec. (%10.2f cpu)\n\n",
            nrows, end_time - begin_time, end_cpu -
begin_cpu);
}

/*-----
+
| Load the STOCK table.
+-----
*/

if (do_A || do_s) {
    nrows = (eware - bware + 1) * STOCFAC;

    fprintf (stderr, "Loading/generating stock: w%d - w%d (%d
rows)\n ",
            bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    s_s_i_id = 1;
    s_s_w_id = bware;

    while (s_s_w_id <= eware) {
        status = OCISmtExecute(tpcsvc, curss, errhp, (ub4) 1,
(ub4) 0,
                                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
            OCIERROR(errhp, status);
            quit ();
            exit (1);
        }
        if (s_s_count == 0) {
            s_s_w_id--;
            break;
        }
        else s_s_w_id++;
    }

    if (s_s_w_id < bware) s_s_w_id = bware;
    else {
        if (s_s_w_id > eware) s_s_w_id = eware;
        while (s_s_i_id <= STOCFAC) {
            status = OCISmtExecute(tpcsvc, curss, errhp, (ub4)
1, (ub4) 0,
                                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                                (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                OCIERROR(errhp, status);
                quit ();
                exit (1);
            }
            if (s_s_count == 0) {
                break;
            }
            else s_s_i_id++;
        }
    }
    if (s_s_i_id > STOCFAC) {

```

```

        s_s_i_id=1;
        s_s_w_id++;
    }
    fprintf(stderr,"start at s_i_id: %d, s_w_id: %d\n ",
s_s_i_id, s_s_w_id);

    sid = s_s_i_id - 1;
    swid = s_s_w_id;
    nrows = (eware - s_s_w_id + 1) * STOCFAC - ( s_s_i_id -
1);
    fprintf (stderr, "remaining rows: %d\n ", nrows);
    loopcount = 0;

    for (row = 0; row < nrows; ) {
        /* added row < nrows condition on next line - alex.ni
*/
        for (i = 0; (i < STOCARR) && (row < nrows); i++, row++) {
            if (++sid > STOCFAC) { /* cheap mod */
                sid = 1;
                swid++;
            }
            s_quantity[i] = (lrand48 () % 91) + 10;
            randstr (s_dist_01[i], 24, 24);
            randstr (s_dist_02[i], 24, 24);
            randstr (s_dist_03[i], 24, 24);
            randstr (s_dist_04[i], 24, 24);
            randstr (s_dist_05[i], 24, 24);
            randstr (s_dist_06[i], 24, 24);
            randstr (s_dist_07[i], 24, 24);
            randstr (s_dist_08[i], 24, 24);
            randstr (s_dist_09[i], 24, 24);
            randstr (s_dist_10[i], 24, 24);
            randdatastr (s_data[i], 26, 50);

            if (gen) {
                printf ("%d %d %d %s %s %s %s %s %s %s %s %s 0
0 0 %s\n",
                    sid, swid, s_quantity[i], s_dist_01[i],
                    s_dist_02[i],
                    s_dist_03[i], s_dist_04[i], s_dist_05[i],
                    s_dist_06[i],
                    s_dist_07[i], s_dist_08[i], s_dist_09[i],
                    s_dist_10[i],
                    s_data[i]);
            }
            else {
                s_i_id[i] = sid;
                s_w_id[i] = swid;
            }
        }
        if (gen) {
            fflush (stdout);
        }
        else {
            /* Changed to STOCKARR to i - alex.ni */
            status = OCISmtExecute(tpcsvc, curs, errhp, (ub4)
i, (ub4) 0,
                (CONST OCISnapshot*) 0, (OCISnapshot*)
0,
                (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n",
s_w_id[0], s_i_id[0]);
                OCIERROR(errhp, status);
                quit ();
                exit (1);
            }
        }
        if ((++loopcount) % 50)
            fprintf (stderr, ".");
        else
            fprintf (stderr, " %d rows committed\n ", row);
    }

    end_time = gettimeofday ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done. %d rows loaded/generated in
%10.2f sec. (%10.2f cpu)\n\n",
nrows < 0 ? 0 : nrows, end_time - begin_time,
end_cpu - begin_cpu);

/*-----
+
| Load the STOCK table (cluster around s_i_id). |
+-----
*/

    if (do_S) {
        nrows = (eitem - bitem + 1) * (eware - bware + 1);

        fprintf (stderr, "Loading/generating stock: i%d - i%d, w%d
- w%d (%d rows)\n ",
                bitem, eitem, bware, aware, nrows);

        begin_time = gettimeofday ();
        begin_cpu = getcpu ();

        s_s_i_id = bitem;
        s_s_w_id = bware;

        while (s_s_i_id <= eitem) {

```

```

        status = OCISmtExecute(tpcsvc, curs, errhp, (ub4) 1,
(ub4) 0,
                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
            OCIERROR(errhp, status);
            quit ();
            exit (1);
        }
        if (s_s_count == 0) {
            s_s_i_id--;
            break;
        }
        else s_s_i_id++;
    }

    if (s_s_i_id < bitem) s_s_i_id = bitem;
    else {
        if (s_s_i_id > eitem) s_s_i_id = eitem;
        while (s_s_w_id <= aware) {
            status = OCISmtExecute(tpcsvc, curss, errhp, (ub4)
1, (ub4) 0,
                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                OCIERROR(errhp, status);
                quit ();
                exit (1);
            }
            if (s_s_count == 0) {
                break;
            }
            else s_s_w_id++;
        }
    }
    if (s_s_w_id > aware) {
        s_s_w_id=bware;
        s_s_i_id++;
    }

    fprintf(stderr,"start at s_i_id: %d, s_w_id: %d\n ",
s_s_i_id, s_s_w_id);

    sid = s_s_i_id;
    swid = s_s_w_id - 1;
    nrows = (eitem - s_s_i_id + 1) * (eware - bware + 1) -
(s_s_w_id - bware);
    fprintf (stderr, "remaining rows: %d\n ", nrows);
    loopcount = 0;

    for (row = 0; row < nrows; ) {
        for (i = 0; i < STOCARR && row < nrows; i++, row++) {
            if (++swid > aware) { /* cheap mod */
                swid = bware;
                sid++;
            }
            s_quantity[i] = (lrand48 () % 91) + 10;
            randstr (s_dist_01[i], 24, 24);
            randstr (s_dist_02[i], 24, 24);
            randstr (s_dist_03[i], 24, 24);
            randstr (s_dist_04[i], 24, 24);
            randstr (s_dist_05[i], 24, 24);
            randstr (s_dist_06[i], 24, 24);
            randstr (s_dist_07[i], 24, 24);
            randstr (s_dist_08[i], 24, 24);
            randstr (s_dist_09[i], 24, 24);
            randstr (s_dist_10[i], 24, 24);
            randdatastr (s_data[i], 26, 50);

            if (gen) {
                printf ("%d %d %d %s %s %s %s %s %s %s %s %s 0
0 0 %s\n",
                    sid, swid, s_quantity[i], s_dist_01[i],
                    s_dist_02[i],
                    s_dist_03[i], s_dist_04[i], s_dist_05[i],
                    s_dist_06[i],
                    s_dist_07[i], s_dist_08[i], s_dist_09[i],
                    s_dist_10[i],
                    s_data[i]);
            }
            else {
                s_i_id[i] = sid;
                s_w_id[i] = swid;
            }
        }
        if (gen) {
            fflush (stdout);
        }
        else {
            status = OCISmtExecute(tpcsvc, curs, errhp, (ub4)
i, (ub4) 0,
                (CONST OCISnapshot*) 0, (OCISnapshot*)
0,
                (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                fprintf (stderr, "Aborted at w_id %d, s_i_id
%d\n", s_w_id[0], s_i_id[0]);
                OCIERROR(errhp, status);
                quit ();
                exit (1);
            }
        }
        if ((++loopcount) % 50)
            fprintf (stderr, ".");
        else

```

```

        fprintf (stderr, " %d rows committed\n", row);
    }

    end_time = gettimeofday ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done. %d rows loaded/generated in
%10.2f sec. (%10.2f cpu)\n\n",
nrows < 0 ? 0 : nrows, end_time - begin_time,
end_cpu - begin_cpu);
}

/*-----
+ | Load the HISTORY table. |
+-----
*/

if (do_A || do_h) {
nrows = (eware - bware + 1) * HISTFAC;

fprintf (stderr, "Loading/generating history: w%d - w%d
(%d rows)\n",
bware, aware, nrows);

begin_time = gettimeofday ();
begin_cpu = getcpu ();

cid = 0;
cddid = 1;
cwid = bware;
loopcount = 0;

for (row = 0; row < nrows; ) {
for (i = 0; i < HISTARR; i++, row++) {
cid++;
if (cid > CUSTFAC) { /* cycle cust id */
cid = 1; /* cheap mod */
cddid++; /* shift district cycle
*/
if (cddid > DISTFAC) {
cddid = 1;
cwid++; /* shift warehouse cycle
*/
}
}
h_c_id[i] = cid;
h_d_id[i] = cddid;
h_w_id[i] = cwid;
randstr (h_data[i], 12, 24);
if (gen) {
printf ("%d %d %d %d %d %s 1000 %s\n", cid, cddid,
cwid, sdate, h_data[i]);
}
}
if (gen) {
fflush (stdout);
}
else {
status = OCIStmtExecute(tpcsvc, curh, errhp, (ub4)
HISTARR, (ub4) 0, (CONST OCISnapshot*) 0, (OCISnapshot*)
0, (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
if (status != OCI_SUCCESS) {
fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id
%d\n",
h_w_id[0], h_d_id[0], h_c_id[0]);
OCIERROR(errhp, status);
quit ();
exit (1);
}
}
if ((++loopcount) % 50)
fprintf (stderr, ".");
else
fprintf (stderr, " %d rows committed\n", row);
}

end_time = gettimeofday ();
end_cpu = getcpu ();
fprintf (stderr, "Done. %d rows loaded/generated in
%10.2f sec. (%10.2f cpu)\n\n",
nrows, end_time - begin_time, end_cpu -
begin_cpu);
}

/*-----
+ | Load the ORDERS and ORDER-LINE table. |
+-----
*/

if (do_A || do_o) {
int batch_olcnt;

nrows = (eware - bware + 1) * ORDEFAC * DISTFAC;

fprintf (stderr, "Loading/generating orders and order-
line: w%d - w%d (%d ord, ~%d ordl)\n",
bware, aware, nrows, nrows * 10);

begin_time = gettimeofday ();
begin_cpu = getcpu ();

```

```

cid = 0;
cddid = 1;
cwid = bware;
loopcount = 0;

for (row = 0; row < nrows; ) {
batch_olcnt = 0;

for (i = 0; i < ORDEARR; i++, row++) {
cid++;
if (cid > ORDEFAC) { /* cycle cust id */
cid = 1; /* cheap mod */
cddid++; /* shift district cycle
*/
if (cddid > DISTFAC) {
cddid = 1;
cwid++; /* shift warehouse cycle
*/
}
}
o_carrier_id[i] = lrand48 () % 10 + 1;
o_ol_cnt[i] = olcnt = lrand48 () % 11 + 5;

if (gen) {
if (cid < 2101) {
printf ("%d %d %d %d %s %d %d %d\n", cid, cddid,
cwid,
randperm3000[cid - 1],
sdate, o_carrier_id[i],
o_ol_cnt[i]);
}
else {
/* set carrierid to 11 instead of null */
printf ("%d %d %d %d %s %d %d %d\n", cid, cddid,
cwid,
randperm3000[cid - 1], sdate,
o_ol_cnt[i]);
}
}
else {
o_id[i] = cid;
o_d_id[i] = cddid;
o_w_id[i] = cwid;
o_c_id[i] = randperm3000[cid - 1];
if (cid >= 2101) {
o_carrier_id[i] = 11;
}
}
}
for (j = 0; j < o_ol_cnt[i]; j++, batch_olcnt++) {
+ 1;
if (cid < 2101)
ol_amount[batch_olcnt] = 0;
else
ol_amount[batch_olcnt] = (lrand48 () % 999999
+ 1);
randstr (str24[j], 24, 24);

if (gen) {
if (cid < 2101) {
fprintf (olfp, "%d %d %d %d %s %d %d %d %d
%5\n", cid,
cddid, cwid, j + 1, sdate,
ol_i_id[batch_olcnt], cwid,
ol_amount[batch_olcnt], str24[j]);
}
else {
/* Insert a default date instead of null
date */
fprintf (olfp, "%d %d %d %d 01-Jan-1811 %d
%5 %d %s\n", cid,
cddid, cwid, j + 1,
ol_i_id[batch_olcnt], cwid,
ol_amount[batch_olcnt], str24[j]);
}
}
else {
ol_o_id[batch_olcnt] = cid;
ol_d_id[batch_olcnt] = cddid;
ol_w_id[batch_olcnt] = cwid;
ol_number[batch_olcnt] = j + 1;
ol_supply_w_id[batch_olcnt] = cwid;
strncpy (ol_dist_info[batch_olcnt], str24[j],
24);
}
}
if (gen) {
fflush (olfp);
}
}
o_cnt = ORDEARR;
ol_cnt = batch_olcnt;

for (j = 0; j < batch_olcnt; j++) {
ol_o_id_len[j] = sizeof(int);
ol_d_id_len[j] = sizeof(int);
ol_w_id_len[j] = sizeof(int);
ol_number_len[j] = sizeof(int);
ol_i_id_len[j] = sizeof(int);
ol_supply_w_id_len[j] = sizeof(int);
ol_dist_info_len[j] = 24;
ol_amount_len[j] = sizeof(int);
}
for (j = batch_olcnt; j < 15*ORDEARR; j++) {

```

```

        ol_o_id_len[j] = 0;
        ol_d_id_len[j] = 0;
        ol_w_id_len[j] = 0;
        ol_number_len[j] = 0;
        ol_i_id_len[j] = 0;
        ol_supply_w_id_len[j] = 0;
        ol_dist_info_len[j] = 0;
        ol_amount_len[j] = 0;
    }

    o_id_clen = ORDEARR;
    o_d_id_clen = ORDEARR;
    o_w_id_clen = ORDEARR;
    o_c_id_clen = ORDEARR;
    o_carrier_id_clen = ORDEARR;
    o_ol_cnt_clen = ORDEARR;

    ol_o_id_clen = batch_olcnt;
    ol_d_id_clen = batch_olcnt;
    ol_w_id_clen = batch_olcnt;
    ol_number_clen = batch_olcnt;
    ol_i_id_clen = batch_olcnt;
    ol_supply_w_id_clen = batch_olcnt;
    ol_dist_info_clen = batch_olcnt;
    ol_amount_clen = batch_olcnt;

(ub4) 1, (ub4) 0,
        (CONST OCISnapshot*) 0,
(OCISnapshot*) 0,
        (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS );

        if ((++loopcount) % 50) {
            fprintf (stderr, ".");
        } else {
            fprintf (stderr, " %d orders committed\n",
row);
        }
    }

    end_time = gettimeofday ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done. %d orders loaded/generated in
%10.2f sec. (%10.2f cpu)\n\n",
nrows, end_time - begin_time, end_cpu -
begin_cpu);
}

/*-----
+
| Load the NEW-ORDER table.
+-----
*/

if (do_A || do_n) {
    nrows = (eware - bware + 1) * NEWOFAC * DISTFAC;

    fprintf (stderr, "Loading/generating new-order: w%d - w%d
(%d rows)\n",
bware, aware, nrows);

    begin_time = gettimeofday ();
    begin_cpu = getcpu ();

    cid = 0;
    cdid = 1;
    cwid = bware;
    loopcount = 0;

    for (row = 0; row < nrows; ) {
        for (i = 0; i < NEWOARR; i++, row++) {
            cid++;
            if (cid > NEWOFAC) {
                cid = 1;
                cdid++;
                if (cdid > DISTFAC) {
                    cdid = 1;
                    cwid++;
                }
            }

            if (gen) {
                printf ("%d %d %d\n", cid + 2100, cdid, cwid);
            }
            else {
                no_o_id[i] = cid + 2100;
                no_d_id[i] = cdid;
                no_w_id[i] = cwid;
            }
        }

        if (gen) {
            fflush (stdout);
        }
        else {
            status = OCISmtExecute(tpcsvc, curno, errhp, (ub4)
NEWOARR, (ub4) 0,
                (CONST OCISnapshot*) 0,
(OCISnapshot*) 0,
                (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n",
cwid, cdid, cid + 2100);
                OCIERROR(errhp, status);
                quit ();
                exit (1);
            }
        }
    }
}

```

```

    }
}

if ((++loopcount) % 45)
    fprintf (stderr, ".");
else
    fprintf (stderr, " %d rows committed\n", row);
}

    end_time = gettimeofday ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done. %d rows loaded/generated in
%10.2f sec. (%10.2f cpu)\n\n",
nrows, end_time - begin_time, end_cpu -
begin_cpu);
}

/*-----
+
| clean up and exit.
+-----
*/

if (olfp)
    fclose (olfp);
if (!gen)
    quit ();
exit (0);
}

void initperm ()
{
    int i;
    int pos;
    int temp;

    /* init randperm3000 */

    for (i = 0; i < 3000; i++)
        randperm3000[i] = i + 1;
    for (i = 3000; i > 0; i--) {
        pos = lrand48 () % i;
        temp = randperm3000[i - 1];
        randperm3000[i - 1] = randperm3000[pos];
        randperm3000[pos] = temp;
    }
}

void randstr (str, x, y)
char *str;
int x;
int y;
{
    int i, j;
    int len;

    len = (lrand48 () % (y - x + 1)) + x;
    for (i = 0; i < len; i++) {
        j = lrand48 () % 62;
        if (j < 26)
            str[i] = (char) (j + 'a');
        else if (j < 52)
            str[i] = (char) (j - 26 + 'A');
        else
            str[i] = (char) (j - 52 + '0');
    }
    str[len] = '\0';
}

void randdatastr (str, x, y)
char *str;
int x;
int y;
{
    int i, j;
    int len;
    int pos;

    len = (lrand48 () % (y - x + 1)) + x;
    for (i = 0; i < len; i++) {
        j = lrand48 () % 62;
        if (j < 26)
            str[i] = (char) (j + 'a');
        else if (j < 52)
            str[i] = (char) (j - 26 + 'A');
        else
            str[i] = (char) (j - 52 + '0');
    }
    str[len] = '\0';
    if ((lrand48 () % 10) == 0) {
        pos = (lrand48 () % (len - 8));
        str[pos] = 'O';
        str[pos + 1] = 'R';
        str[pos + 2] = 'I';
        str[pos + 3] = 'G';
        str[pos + 4] = 'I';
        str[pos + 5] = 'N';
        str[pos + 6] = 'A';
        str[pos + 7] = 'L';
    }
}

void randnum (str, len)
char *str;
int len;
{

```

```

int i;
for (i = 0; i < len; i++)
    str[i] = (char) (lrand48 () % 10 + '0');
str[len] = '\0';
}

void randlastname (str, id)
char *str;
int id;
{
    id = id % 1000;
    strcpy (str, lastname[id / 100]);
    strcat (str, lastname[(id / 10) % 10]);
    strcat (str, lastname[id % 10]);
}

int NURand (A, x, y, cnum)
int A, x, y, cnum;
{
    int a, b;

    a = lrand48 () % (A + 1);
    b = (lrand48 () % (y - x + 1)) + x;
    return (((a | b) + cnum) % (y - x + 1)) + x;
}

void sysdate (sdate)
char *sdate;
{
    time_t tp;
    struct tm *tmptr;

    time (&tp);
    tmptr = localtime (&tp);
    strftime (sdate, 29, "%d-%b-%Y", tmptr);
}

int ocierror(fname, lineno, errhp, status)
char *fname;
int lineno;
OCIError *errhp;
sword status;
{
    text errbuf[512];
    sb4 errcode;
    sb4 lstat;
    ub4 recno=2;

    switch (status) {
    case OCI_SUCCESS:
        break;
    case OCI_SUCCESS_WITH_INFO:
        fprintf(stderr, "Module %s Line %d\n", fname, lineno);
        fprintf(stderr, "Error - OCI_SUCCESS_WITH_INFO\n");
        lstat = OCIErrorGet (errhp, recno++, (text *) NULL,
        &errcode, errbuf,
        (ub4) sizeof(errbuf),
OCI_HTYPE_ERROR);
        fprintf(stderr, "Error - %s\n", errbuf);
        break;
    case OCI_NEED_DATA:
        fprintf(stderr, "Module %s Line %d\n", fname, lineno);
        fprintf(stderr, "Error - OCI_NEED_DATA\n");
        return (IRRECERR);
    case OCI_NO_DATA:
        fprintf(stderr, "Module %s Line %d\n", fname, lineno);
        fprintf(stderr, "Error - OCI_NO_DATA\n");
        return (IRRECERR);
    case OCI_ERROR:
        lstat = OCIErrorGet (errhp, (ub4) 1,
        (text *) NULL, &errcode, errbuf,
        (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
        if (errcode == NOT_SERIALIZABLE) return (errcode);
        if (errcode == SNAPSHOT_TOO_OLD) return (errcode);
        while (lstat != OCI_NO_DATA)
        {
            fprintf(stderr, "Module %s Line %d\n", fname, lineno);
            fprintf(stderr, "Error - %s\n", errbuf);
            lstat = OCIErrorGet (errhp, recno++, (text *) NULL,
        &errcode, errbuf,

```

```

        (ub4) sizeof(errbuf),
OCI_HTYPE_ERROR);
        }
        return (errcode);
    case OCI_INVALID_HANDLE:
        fprintf(stderr, "Module %s Line %d\n", fname, lineno);
        fprintf(stderr, "Error - OCI_INVALID_HANDLE\n");
        exit(-1);
    case OCI_STILL_EXECUTING:
        fprintf(stderr, "Module %s Line %d\n", fname, lineno);
        fprintf(stderr, "Error - OCI_STILL_EXECUTE\n");
        return (IRRECERR);
    case OCI_CONTINUE:
        fprintf(stderr, "Module %s Line %d\n", fname, lineno);
        fprintf(stderr, "Error - OCI_CONTINUE\n");
        return (IRRECERR);
    default:
        fprintf(stderr, "Module %s Line %d\n", fname, lineno);
        fprintf(stderr, "Status - %s\n", status);
        return (IRRECERR);
    }
    return (RECOVERERR);
}

```

```

-----
views.sql
-----
connect tpcc/tpcc;
set echo on;

create or replace view wh_cust
(w_id, w_tax, c_id, c_d_id, c_w_id, c_discount, c_last,
c_credit)
as select w.w_id, w.w_tax,
        c.c_id, c.c_d_id, c.c_w_id, c.c_discount, c.c_last,
c.c_credit
    from cust c, ware w
    where w.w_id = c.c_w_id;

create or replace view wh_dist
(w_id, d_id, d_tax, d_next_o_id, w_tax)
as select w.w_id, d.d_id, d.d_tax, d.d_next_o_id, w.w_tax
    from dist d, ware w
    where w.w_id = d.d_w_id;

create or replace view stock_item
(i_id, s_w_id, i_price, i_name, i_data, s_data, s_quantity,
s_order_cnt, s_ytd, s_remote_cnt,
s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10)
as
select /*+ leading(s) use_nl(i) */
    i.i_id, s.w_id, i.i_price, i.i_name, i.i_data, s_data,
s_quantity,
s_order_cnt, s_ytd, s_remote_cnt,
s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10
    from stok s, item i
    where i.i_id = s.s_i_id;

set echo off;

```

```

-----
create_raw_nodes.sh
-----
{
#count=110 #for backup
count=1
while read a b
do
    name=`basename ${a}`
    mknod /home/oracle/dev/${name} c 162 ${count}
    let count+=1
    echo $count
done < /home/oracle/tpcc_96000/files.dat
}

```


Appendix C: Tunable Parameters

SEQUENCE OF EVENTS FOR PERFORMANCE RUN

1. Boot up systems clients, servers, & RTEs.
2. Change interrupt delay on cpqarray running cfgcciss 3000.
3. Startup the database on the server using p_run.ora.
4. Start the RTE.
5. Adjust RTE throttle.

```
-----
p_run.ora
-----
compatible = 10.1.0.0.0
db_name = tpcc
control_files = /home/oracle/dev/control_001
recovery_parallelism = 20
parallel_max_servers = 0
db_cache_size = 1700M
db_keep_cache_size = 18040M
db_recycle_cache_size = 150M
db_8k_cache_size = 92M
db_16k_cache_size = 2318M
db_files = 300
dml_locks = 100
log_buffer = 1048576
processes = 150
sessions=200
shared_pool_size = 900M
cursor_space_for_time = TRUE
db_block_size = 2048
undo_management = auto
undo_retention = 2
undo_tablespace = undo_1
_in_memory_undo=true
_imu_pools=80
_collect_undo_stats = false
_undo_autotune=false
plsql_optimize_level=2
query_rewrite_enabled = false
aq_tm_processes = 0
log_checkpoint_interval = 0
log_checkpoint_timeout = 1200
log_checkpoints_to_alert = true
fast_start_mttr_target = 0
replication_dependency_tracking = false
_db_cache_pre_warm = false
db_block_checking = false
db_block_checksum = false
java_pool_size = 0
timed_statistics = false
statistics_level = basic
_enable_NUMA_optimization = false
_db_block_numa = 1
_db_writer_flush_imu = false
_cursor_cache_frame_bind_memory = true
_check_block_after_checksum=false
lgwr_async_io = true
trace_enabled=false
_kghdsidx_count=1
_two_pass=false
_lightweight_hdrs=true
remote_os_authent=true
```

```
-----
cfgcciss.c
-----
#include <stdio.h>
#include <fcntl.h>
#include <linux/cciss_ioctl.h>

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

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

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

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

for (i=0; i<9; i++) {
    sprintf(ctrlname, "/dev/cciss/c%dd", i);

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

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

    cfg_coalint_new.delay = delay;
    cfg_coalint_new.count = 1;

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

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

    close(fd);
}
}
```

```
-----
chkconfig --list (server)
-----
keytable      0:off 1:on 2:on 3:off 4:on 5:on 6:off
atd            0:off 1:off 2:off 3:off 4:on 5:on 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:on 5:on 6:off
sendmail      0:off 1:off 2:off 3:off 4:off 5:off 6:off
kudzu         0:off 1:off 2:off 3:on 4:on 5:on 6:off
netdump-server 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
acpid         0:off 1:off 2:off 3:off 4:on 5:on 6:off
ipchains      0:off 1:off 2:on 3:off 4:on 5:on 6:off
iptables     0:off 1:off 2:off 3:off 4:on 5:on 6:off
cron          0:off 1:off 2:off 3:off 4:off 5:off 6:off
anacron       0:off 1:off 2:off 3:off 4:off 5:off 6:off
lpd           0:off 1:off 2:off 3:off 4:off 5:off 6:off
xfs           0:off 1:off 2:on 3:off 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:off 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:on 5:on 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
smartd       0:off 1:off 2:off 3:off 4:off 5:off 6:off
rhnstd       0:off 1:off 2:off 3:off 4:off 5:off 6:off
isdns        0:off 1:off 2:on 3:off 4:on 5:on 6:off
sshd         0:off 1:off 2:on 3:on 4:on 5:on 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
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
innd         0:off 1:off 2:off 3:off 4:off 5:off 6:off
winbind      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
postgresql  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
ip6tables    0:off 1:off 2:on 3:off 4:on 5:on 6:off
rarpd        0:off 1:off 2:off 3:off 4:off 5:off 6:off
named        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:on 5:on 6:off
ipvsadm      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
reconfig     0:off 1:off 2:off 3:on 4:on 5:on 6:off
amd          0:off 1:off 2:off 3:off 4:off 5:off 6:off
bootparamd  0:off 1:off 2:off 3:off 4:off 5:off 6:off
dhcpd        0:off 1:off 2:off 3:off 4:off 5:off 6:off
gated        0:off 1:off 2:off 3:off 4:off 5:off 6:off
irda         0:off 1:off 2:off 3:off 4:off 5:off 6:off
iscsi        0:off 1:off 2:off 3:off 4:off 5:off 6:off
junkbuster   0:off 1:off 2:off 3:off 4:off 5:off 6:off
kadmin       0:off 1:off 2:off 3:off 4:off 5:off 6:off
kprop        0:off 1:off 2:off 3:off 4:off 5:off 6:off
krb524       0:off 1:off 2:off 3:off 4:off 5:off 6:off
krb5kdc      0:off 1:off 2:off 3:off 4:off 5:off 6:off
mars-nwe     0:off 1:off 2:off 3:off 4:off 5:off 6:off
mcserv       0:off 1:off 2:off 3:off 4:off 5:off 6:off
mysqld       0:off 1:off 2:off 3:off 4:off 5:off 6:off
ups          0:off 1:off 2:off 3:off 4:off 5:off 6:off
ldap         0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

```

routed      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
bgpd       0:off 1:off 2:off 3:off 4:off 5:off 6:off
ospf6d     0:off 1:off 2:off 3:off 4:off 5:off 6:off
ospfd      0:off 1:off 2:off 3:off 4:off 5:off 6:off
ripd       0:off 1:off 2:off 3:off 4:off 5:off 6:off
ripngd     0:off 1:off 2:off 3:off 4:off 5:off 6:off
zebra      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
services: off
servers: off
time-udp: off
time: off
dbskkd-cdb: off
sgi_fam: on
finger: off
rexec: on
rlogin: on
rsh: on
ntalk: off
talk: off
telnet: off
wu-ftpd: off
rsync: on
amanda: off
comsat: off
amandaidx: off
amidxtape: off
imap: off
imaps: off
ipop2: off
ipop3: off
pop3s: off
eklogin: off
gssftp: off
klogin: off
krb5-telnet: off
kshell: off
swat: off
tftp: off

```

setrrpri.sh

```

sleep $1

# Run oracle system processes at sched_rr priority
/home/oracle/config/rr -p 95 $(ps aux | grep ora_ | grep -v grep
| awk '{print $2}')

# Run oracle client processes at sched_rr priority
/home/oracle/config/rr -p 95 $(ps aux | grep oracletp | grep -v
grep | awk '{print $2}')

# Run lgwr at a higher priority

/home/oracle/config/rr -p 96 $(ps aux | grep ora_lgwr | grep -v
grep | awk '{print $2}')

```

elilo.conf

```

prompt
timeout=50
default=linux

image=vmlinuz-2.4.21-15.0.3.EL
label=2.4.21-15.0.3.EL
initrd=initrd-2.4.21-15.0.3.EL.img
read-only
append="root=LABEL"

image=vmlinuz-2.4.21-15.ELcustom
label=linux
initrd=initrd-2.4.21-15.ELcustom.img
read-only
append="root=LABEL/ hugetlbpool=23296"

```

```

-----
rc.local
-----
#!/bin/sh
#
# This script will be executed *after* all the other init
scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

```

```

touch /var/lock/subsys/local
echo kiobuf 60 10 > /proc/slabinfo

sh /root/new-partitions/new-run_links.sh
mount none -t hugetlbfs /mnt/htlb
chown oracle:dba /mnt/htlb

rdate -s dbgate

```

sysctl.conf

```

# Kernel sysctl configuration file for Red Hat Linux
#
# For binary values, 0 is disabled, 1 is enabled. See sysctl(8)
and
# sysctl.conf(5) for more details.

# Controls IP packet forwarding
net.ipv4.ip_forward = 0

# Controls source route verification
net.ipv4.conf.default.rp_filter = 1

# Controls the System Request debugging functionality of the
kernel
kernel.sysrq = 0

# Controls whether core dumps will append the PID to the core
filename.
# Useful for debugging multi-threaded applications.

kernel.core_uses_pid = 1
kernel.shmall = 2097512
kernel.shmmax = 34359738368
kernel.sem = 250 32000 100 128
fs.file-max = 65536
net.ipv4.ip_local_port_range = 1024 65000
vm.hugetlbpool = 30720
fs.aio-max-nr = 104857
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_window_scaling = 0
net.ipv4.tcp_sack = 0

```

DBInit

```

[TPCC]
DBConnections=40
StartTerm=20401
KMaxterms=210
DeliveryQueues=200
DeliveryThreads=40

```

Appendix D:
Third Party Letters

Phone Orders
800.287.2323

ORDER TRACKING CLEARANCE SHOPPING

Home About Us Product Return Customer Service Contact Us Phone Orders

IMPORTANT:
Closed for Holiday
Sept 29-Oct 8
[Click Here](#)

Netgear WGR614NA Wireless-G 2.4ghz 4 Port Cable/DSL Router...
\$39.99
New! Sandisk 256MB Ultra II Compact Flash Card... \$44
SAVE! Creative Labs Nomad Jukebox Zen XTRA 40GB \$229.99
WOW! RIO DNNA Digital Networks Rio Karma USB 2.0 20GB Hard Drive... \$233

shop by product shop by brand

SEARCH STORE

Bottom of Form

PRODUCTS

- CDR/CDRW
- CAMCORDERS
- CASES
- CLOSEOUTS
- CONTROLLER CARDS
- CPU's
- DIGITAL CAMERAS
- Digital Accessories
- DVD's
- FLASH CARDS
- FLOPPY DRIVES
- GPS
- HANDHELDS/PDA's
- HARD DRIVES
- HUBS
- KEYBOARD/MOUSE
- MEMORY
- MEMORY STICK
- MICROSOFT
- MODEMS
- MONITORS
- MOTHERBOARDS
- MP3 PLAYERS

NETGEAR



NETGEAR GS105NA 5-PORT GIGABIT ETHERNET 10/100/1000MBPS SWITCH
• Price: \$64.00
• Item is brand new and in Stock!

tell a friend

DESCRIPTION:

The NETGEAR GS105, 5-Port Gigabit Ethernet Switch lets you build a system that provides dedicated, auto-switching 1000, 100, or 10Mbps connections so you can move even very large files across your network instantly--up to 2000Mbps on each port. [More Info & Product Specification](#)

More Info & Product Specification

- Five 10/100/1000 Mbps Ethernet ports
- High-speed, auto-sensing, full-duplex connections
- Supports PC or Macintosh® O/S
- Sturdy, space-saving form factor
- Silent operation for a peaceful work environment
- Package Contents: 5-port 10/100/1000 Mbps Gigabit Ethernet Switch GS105, Wall-mount kit, Power adapter, Installation guide, Warranty and Support information card
- Status LEDs: Power, Collision, Activity, and Duplex indicators for each port, Link and Speed indicators built into each RJ-45 port
- AC Power: 14.4W 12VAC, 1.2A
- Physical Specifications: Dimensions: 158 x 105 x 25 mm (6.22 x 4.13 x .98 in.)
- Warranty: Switch: 5 years, Power adapter: 2 years
- Pictures are for illustration purposes only.

FEAT

- Pa Tungst Connec 64MB I 802.11 \$339.9
- R Netwo 64MB I Retail I \$49.95
- H H5555 2003 P \$624.0
- N 3700 3 Digital \$249.5

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

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

Microsoft

September 15, 2004

Hewlett-Packard Company
paul cao
22555 SH 249
Houston, TX 77070

Mr. cao:

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

All pricing shown is in US Dollars (\$).

Part Number	Description	Unit Price	Quantity	Price
C11-00821	Windows 2000 Server <i>Server License Only - No CALs</i> <i>Discount Schedule: No Level</i> <i>Unit Price reflects a 8% discount from the</i> <i>retail unit price of \$799.</i>	\$738	2	\$1,476

All products are currently orderable through Microsoft's normal distribution channels.

This quote is valid for the next 90 days.

If we can be of any further assistance, please contact Jamie Reding at (425) 703-0510 or jamiere@microsoft.com.

Reference ID: PCpaca0415092282

Please include this Reference ID in any correspondence regarding this price quote.

Appendix E:
Database Pricing

----- Original Message -----

Subject: Pricing for 2 processor system

Date: Thu, 01 Jul 2004 17:11:49 -0700

From: MaryBeth Pierantoni

[<mary.beth.pierantoni@oracle.com>](mailto:mary.beth.pierantoni@oracle.com)

Reply-To: mary.beth.pierantoni@oracle.com

Organization: Server Technologies

To: Othayoth, Raghunath

[<raghunath.othayoth@hp.com>](mailto:raghunath.othayoth@hp.com)

CC: Harald Eri [<harald.eri@oracle.com>](mailto:harald.eri@oracle.com)

Hello Raghu,

Per your request. Please send me the final executive summary when ready for review and approval.

Thanks,
MaryBeth

Product	Quantity	Price	Extended Price
Oracle Database 10g Standard Edition One, for 3 years, Per Processor, Unlimited Users	2	\$2,497.50	\$4,995
Oracle Database Server Support Package for 3 years	1		\$6,000
Total			10,995

Contact: MaryBeth Pierantoni, mary.beth.pierantoni@oracle.com, 916-315-5081