NEC

TPC Benchmark™ C
Full Disclosure Report

NEC Express5800/1160Xe (8 SMP)

with Oracle® Database 10g Enterprise Edition
and
Red Hat Enterprise Linux AS 4
for Itanium Processor Family

Third Edition
28-Mar-2006

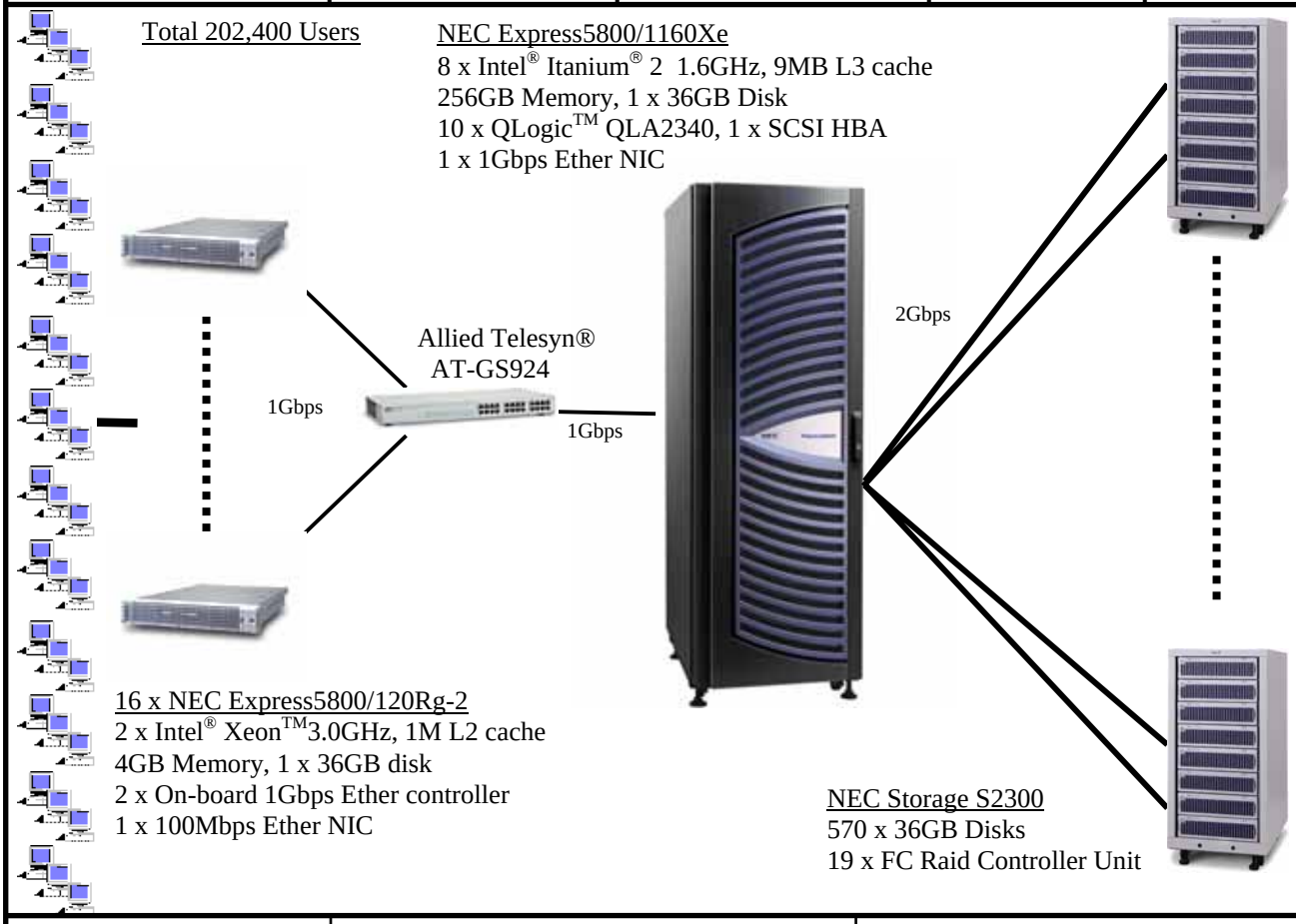
NEC, the Sponsor of this benchmark test, believes that the information in this document is accurate as of the publication date. The information in this document is subject to change without notice. The Sponsor assumes no responsibility for any errors that may appear in this document. The pricing information in this document is believed to accurately reflect the current prices as of the publication date. However, the Sponsor 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. NEC does not warrant or represent that a user can or will achieve similar performance expressed in transactions per minute (tpmC) or normalized price/performance ($/tpmC). No warranty of system performance or price/performance is expressed or implied in this report.

Printed in USA, 2006

NEC	NEC Express5800/1160Xe C/S with Express5800/120Rg-2		TPC-C Rev. 5.5 Report Date 18-Jan-2006 Revised Date 28-Mar-2006
Total System Cost	**TPC-C Throughput**	**Price/Performance**	**Availability Date**
$1,352,709 (USD)	**254,471 tpmC**	**$5.32 per tpmC**	**15-Mar-2006**

Database Server Processors/Cores/Threads	Database Manager	Operating System	Other Software	Number of Users
8/8/8 Intel® Itanium® 2 1.6GHz for Server	Oracle®Database 10g Enterprise Edition	Red Hat Enterprise Linux AS 4	BEA Tuxedo 8.1	202,400



System Component	Server		Each Client	
Processors/Cores/Threads **Cache**	8/8/8	Intel® Itanium® 2 1.6GHz 9MB L3 Cache	2/2/4	Intel® Xeon™ 3.0GHz 1MB L2 Cache
Memory		256GB		4GB
Disk Controllers	10 1	QLogic™ QLA2340 SCSI HBA	1	On-board SCSI controller
Disk Drives	1 570	36GB 36GB	1	36GB
Total Storage		20,556GB		36GB
Others	1 1	DVD-ROM Drive 1Gbps Ether NIC	1 2 1	CD-ROM Drive On-board Ether controller 100Mbps Ether NIC

NEC

NEC Express5800/1160Xe
C/S with Express5800/120Rg-2

TPC-C Rev. 5.5

Report Date: 18-Jan-2006

Description	Part Number	Third Party Brand	Pricing	Unit Price	Qty	Extended Price	3-yr Mnt. Price
Server Hardware							
Express5800/1160Xe system (8way model)	850200719	NEC	1	449,130	1	449,130	107,791
Base Original 16(Express5800/1160Xe base system)					1		
Itanium2 1.6GHz/9M cache					8		
Cell Card Original					4		
8GB Memory Option (4 x 2GB DIMMs)					32		
Core PCI box					1		
Base PCI box					1		
PCI Cable					4		
36GB Ultra 320/m SCSI HDD (1") 10000RPM					1		
10/100/1000Mb Ether card					1		
SCSI card					1		
NEC AccuSync500 (15" monitor, +2 spares)	573757	NEC	5	120	3	360	-
FC HBA QLA2340 (+2 spares)	408081	QLogic	5	952	12	11,424	-
Fast Ether Cable 25' RJ45-RJ45 (+10% spares)	324516	Tripp Lite	5	7	21	157	
					Subtotal	461,071	107,791
Disk Subsystem							
NEC Storage S2300 Basic Unit	850171001	NEC	1	33,999	19	645,981	-
2 RAID_Cntrl, 1GB Cache, 1DE, w/o drives							
1GB(2 x 512MB) PC133 SDRAM memory	062-02301-000	NEC	1	1,999	57	113,943	-
S2300 DE (+10% spares)	062-02402-000	NEC	1	7,999	21	167,979	-
36GB 15K rpm FC HDD (+10% spares)	062-02431-000	NEC	1	579	627	363,033	-
3 yr Telephone Upgrade Hdw & Storage	IS-THSU-1095-0724	NEC	1	1499	19		28,481
Manager SW Support; 7days/ 24hours per day							
42U Rackframe	050-01790-000	NEC	1	1,799	5	8,995	-
FC Cable 10M LC-LC (+10% spares)	062-02304-000	NEC	1	200	21	4,200	-
					Subtotal	1,304,131	28,481
Server Software							
Red Hat Enterprise Linux AS (Standard)		Red Hat	3	1,499	3	4,497	
Oracle Database 10g Enterprise Edition, Processor License for 3 years		Oracle	2	20,000	8	160,000	
Oracle Database Server Support Package for 3 years		Oracle	2	6,000	1		6,000
					Subtotal	164,497	6,000
Client Hardware							
NEC Express5800/120Rg-2							
120Rg-2 base rack svr, one Xeon Processor 3.0GHz/1MB cache	850174001	NEC	1	3,799	16	60,784	-
Xeon 3.0GHz/1MB ECC Processor Kit	062-01777-000	NEC	1	799	16	12,784	-
1GB LP DDR SDRAM (2 x 512MB)	062-02264-000	NEC	1	599	32	19,168	
2GB LP DDR SDRAM (2 x 1GB)	062-02265-000	NEC	1	1,599	16	25,584	-
36.4GB SCSI HDD (1")	062-02398-000	NEC	1	549	16	8,784	-
CD-ROM, 2 x On-board LAN, KB/MS	Included	NEC	1	-	16	-	-
3 years of warranty service to 4-hour response, 7x24	DE-0000-1095-7244	NEC	1	1,599	16	-	25,584
NEC AccuSync500 (15" monitor, +2 spares)	573757	NEC	5	120	3	360	-
42U Rackframe	050-01790-000	NEC	1	1,799	1	1,799	-
Intel PRO/100+ Adapter, 20-pack (+10% spares)	191126	Intel	5	644	1	644	-
Fast Ether Cable 25' RJ45-RJ45 (+10% spares)	324527	Tripp Lite	5	6	36	233	-
					Subtotal	130,140	25,584
Client Software							
Red Hat Enterprise Linux ES (Standard)		Red Hat	3	799	48	38,352	
BEA Tuxedo Core Functionality Services (CFS-R)		BEA	4	1,140	16	18,240	12,096
					Subtotal	56,592	12,096
Connectivity							
Gigabit Ethernet Switch 24port (+2 spares)	806699	Allied Telesyn	5	345	3	1,035	
					Subtotal	1,035	0
					TOTAL	2,117,466	179,952
Oracle Mandatory E-Business Discount						-24,900	
NEC brand total(Pricing 1-NEC)						1,882,164	161,856
NEC brand Large Purchase Cash Prepay Discount	-45%					-846,974	-72,835

Notes:

Pricing: 1-NEC 2-Oracle 3-RedHat 4-BEA 5-CDW

Oracle pricing contact: MaryBeth Pierantoni, mary.beth.pierantoni@oracle.com, 916-315-5081.

Results and methodology audited by Francois Raab of InfoSizing, Inc. (www.sizing.com)

3-Yr. Cost of Ownership: **$1,352,709**

tpmC Rating: **254471**

$ / tpmC: $5.32

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 reflects standard pricing policies for the listed components. For complete details, see the pricing sections of the TPC benchmark specifications.If you find that the stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org. Thank you.

Numerical Quantities Summary

MQTh, Computed Maximum Qualified Throughput					254,471 tpmC	
Response Times(in seconds)	**Average**		**Maximum**		**90%**	
New-Order	0.23		117.21		0.25	
Payment	0.23		117.25		0.25	
Delivery(interactive portion)	0.11		0.19		0.11	
Stock-Level	0.21		97.14		0.24	
Order-status	0.22		93.50		0.25	
Delivery(deferred portion)	0.02		0.48		0.05	
Menu	0.11		0.24		0.11	
Response time delay added for emulated components					0.1	
Transaction Mix , in percent of total transaction						
New-Order					44.81%	
Payment					43.07%	
Delivery					4.04%	
Stock-Level					4.04%	
Order-Status					4.04%	
Keying/Think Times (in seconds)	**Average**		**Min**		**Max**	
New-Order	18.01	12.08	18.01	0.00	18.01	120.7
Payment	3.01	12.07	3.01	0.00	3.01	120.7
Delivery	2.01	5.07	2.01	0.00	2.01	50.7
Stock-Level	2.01	5.08	2.01	0.00	2.01	50.7
Order-Status	2.01	10.08	2.01	0.00	2.01	100.7
Test Duration						
Ramp-up time					9557 seconds	
Measurement interval					7230 seconds	
Number of checkpoints					4	
Checkpoint interval					1753 seconds	
Number of transactions (all types) completed in measurement interval					68443399	

ABSTRACT 8
TPC BENCHMARK™ C METRICS 8
STANDARD AND EXECUTIVE SUMMARY STATEMENTS 8
AUDITOR 8
PREFACE 9
TPC BENCHMARK™ C OVERVIEW 9
DOCUMENT STRUCTURE 10
GENERAL ITEMS 11
ORDER AND TITLES 11
SUMMARY STATEMENT 11
NUMERICAL QUANTITIES SUMMARY 11
APPLICATION PROGRAM 11
SPONSOR 11
PARAMETERS AND OPTIONS 12
CONFIGURATION DIAGRAMS 12
MEASURED CONFIGURATION 13
PRICED SYSTEM CONFIGURATION 14
CLAUSE 1 : LOGICAL DATABASE DESIGN AND RELATED ITEMS 15
TABLE DEFINITIONS 15
TABLE ORGANIZATION 15
INSERT AND DELETE OPERATIONS 15
DISCLOSURE OF PARTITIONING 15
REPLICATION OF TABLES 15
ADDITIONAL AND/OR DUPLICATED ATTRIBUTES IN ANY TABLE 15
CLAUSE 2 : TRANSACTION AND TERMINAL PROFILES RELATED ITEMS 16
RANDOM NUMBER GENERATION 16
TERMINAL INPUT/OUTPUT SCREEN LAYOUT 16
TERMINAL FEATURE VERIFICATION 16
PRESENTATION MANAGER OR INTELLIGENT TERMINAL 16
TRANSACTION PROFILES 16
TRANSACTION MIX 16
QUEUING MECHANISM 17
CLAUSE 3 : TRANSACTION AND SYSTEM PROPERTIES RELATED ITEMS 18
TRANSACTION SYSTEM PROPERTIES (ACID) 18
ATOMICITY TESTS 18
Completed Transactions *18*
Aborted Transactions *18*
CONSISTENCY TESTS 18
ISOLATION TESTS 18
DURABILITY TESTS 19
Instantaneous Interruption and Loss of Memory *19*
Loss of Log and Data Disk *19*
Loss of mirrored write-back cache *19*
CLAUSE 4 : SCALING AND DATABASE POPULATION RELATED ITEMS 20
INITIAL CARDINALITY OF TABLES 20
CONSTANT VALUE FOR THE NURAND FUNCTION 20
DISTRIBUTION OF TABLES AND LOGS 21
TYPE OF DATABASE 22
DATABASE MAPPING 22
60-DAYS SPACE 22
CLAUSE 5 : PERFORMANCE METRICS AND RESPONSE TIME RELATED ITEMS 23

THROUGHPUT 23
RESPONSE TIMES 23
KEYING AND THINK TIMES 23
RESPONSE TIME FREQUENCY DISTRIBUTION CURVES 24
RESPONSE TIME VERSUS THROUGHPUT CURVE 26
NEW-ORDER THINK TIME FREQUENCY DISTRIBUTION 27
NEW-ORDER THROUGHPUT VS. ELAPSED TIME 27
STEADY STATE 28
WORK PERFORMED DURING STEADY STATE 28
MEASUREMENT PERIOD DURATION AND CHECKPOINT DURATION 28
REGULATION OF TRANSACTION MIX 28
TRANSACTION STATISTICS 28
CHECKPOINT COUNT AND LOCATION 29

CLAUSE 6 : SUT,DRIVER,AND COMMUNICATION DEFINITION RELATED ITEMS 30

DESCRIPTIONS OF RTE 30
LOSS OF TERMINAL CONNECTIONS 30
EMULATED COMPONENTS 30
FUNCTIONAL DIAGRAMS AND DETAIL OF DRIVER SYSTEM 30
NETWORK CONFIGURATIONS AND DRIVER SYSTEM 30
NETWORK BANDWIDTH 30
OPERATOR INTERVENTION 30

CLAUSE 7 : PRICING RELATED ITEMS 31

HARDWARE AND SOFTWARE COMPONENTS 31
AVAILABILITY 31
THROUGHPUT, AND PRICE PERFORMANCE 31
COUNTRY SPECIFIC PRICING 31
USAGE PRICING 31
SYSTEM PRICING 32

CLAUSE 8 : AUDIT RELATED ITEMS 33

AUDITOR'S REPORT 33
AVAILABILITY OF THE FULL DISCLOSURE REPORT 33
AUDITOR'S LETTER 34

APPENDIX A : APPLICATION SOURCE CODE 36

APPENDIX B : DATABASE DESIGN 197

APPENDIX C : TUNABLE PARAMETERS 341

APPENDIX D : SPACE CALCULATION 355

APPENDIX E : PRICE QUOTATION 356

Abstract

This report documents the compliance of NEC Corporation's TPC Benchmark™ C tests on the NEC Express5800/1160Xe client/server system with version 5.5 of the TPC Benchmark C Standard Specification. 16 Clients (NEC Express5800/120Rg-2) were used as the front-end clients.

The operating system and the DBMS used on the server were Red Hat Enterprise Linux AS 4 for Itanium Processor Family and Oracle® Database 10g Enterprise Edition. The operating system on the clients was Red Hat Enterprise Linux ES 4 for x86. Those clients ran Apache HTTP Server and BEA Tuxedo® 8.1.
Two standard metrics, transaction-per-minute-C(tpmC) and price per tpmC($/tpmC) are reported, in accordance with the TPC Benchmark™ C Standard. The independent auditor's report by Francois Raab appears at the end of this report.

TPC Benchmark™ C Metrics

The standard TPC Benchmark™ C metrics, tpmC (transactions per minute), price per tpmC (three year capital cost per measured tpmC) are reported.

System	SW	Total System Cost	TpmC	$ per tpmC	Availability Date
NEC Express5800 /1160Xe	Oracle® Database 10g Enterprise Edition Red Hat Enterprise Linux AS 4 for Itanium Processor Family	$1,352,709 (USD)	254,471	$5.32	15-Mar-2006

Standard and Executive Summary Statements

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

Auditor

The benchmark configuration, environment and methodology were audited by Francois Raab of Info Sizing Inc. to verify compliance with the relevant TPC specifications.

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 5.5

TPC Benchmark™ C Overview

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

TPC Benchmark™ C (TPC-C) is an 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 on 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 TPC-C results must include the tpmC rate, the associated price-per-tpmC, and the availability date of the priced configuration.

Although these specifications express implementation in terms of a relational data model with conventional locking scheme, the database may be implemented using any commercially available database management system (DBMS), database server, file system, or other data repository that provides a functionally equivalent implementation. The terms "table", "row", and "column" are used in this document only as examples of logical data structures.

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 any other environment 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 benchmarking when critical capacity planning and/or product evaluation decisions are contemplated.

Benchmark sponsors are permitted several possible system designs, insofar as they adhere to the model described and pictorially illustrated in Clause 6. A Full Disclosure Report of the implementation details, as specified in Clause 8, must be made available along with the reported results.

Document Structure

This TPC Benchmark™ C Full Disclosure Report is organized as follows:

- The main body of the document lists each item in Clause 8 of the TPC-C Standard and explains how each requirement is satisfied.
- Appendix A contains the source code of the TPC-C application code used to implement the TPC-C transactions.
- Appendix B contains the database definition and population code used in the tests.
- Appendix C contains the tunable parameters used in the TPC-C tests.
- Appendix D contains space calculation table.
- Appendix E contains third-party price quotations.

TPC Benchmark™ C Full Disclosure

The TPC Benchmark™ C Standard Specification requires test sponsors to publish, and make available to the public, a full disclosure report for the results to be considered compliant with the Standard. The required contents of the full disclosure report are specified in Clause 8. This report is intended to satisfy the Standard's requirement for full disclosure. It documents the compliance of the benchmark tests with each item listed in Clause 8 of the TPC Benchmark™ C Standard Specification.

In the Standard Specification, the main headings in Clause 8 are keyed to the other clauses. The headings in this report use the same sequence, so that they correspond to the titles or subjects referred to in Clause 8.

Each section in this report begins with the text of the corresponding item from Clause 8 of the Standard Specification, printed in italic type. The plain text that follows explains how the tests comply with the TPC Benchmark™ C requirement. In sections where Clause 8 requires extensive listings, the section refers to the appropriate appendix at the end of this report.

General Items

Order and titles

The order and titles of sections in the Test Sponsor's Full Disclosure report must correspond with the order and titles of sections from the TPC-C standard specification (i.e., this document). The intent is to make it as easy as possible for readers to compare and contrast material in different Full Disclosure reports.

The order and titles of sections in this report correspond with that of the TPC-C standard specification.

Summary Statement

The TPC Executive Summary Statement must be included near the beginning of the Full Disclosure report.

The TPC Executive Summary Statement is included at the beginning of this report.

Numerical Quantities Summary

The numerical quantities listed below must be summarized near the beginning of the Full Disclosure report :

- *measurement interval in minutes,*
- *number of checkpoints in the measurement interval,*
- *longest checkpoint interval in minutes,*
- *number of transactions (all types) completed within the measurement interval,*
- *computed Maximum Qualified Throughput in tpmC,*
- *ninetieth percentile, average and maximum response times for the New-Order, Payment, Order-Status, Stock-Level, Delivery (deferred and interactive) and Menu transactions,*
- *time in seconds added to response time to compensate for delays associated with emulated components,*
- *percentage of transaction mix for each transaction type.*

These numerical quantities are summarized at the beginning of this report.

Application Program

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

Appendix A contains the application source codes used in the TPC-C benchmark.

Sponsor

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

This benchmark test was sponsored by NEC Corporation. NEC has authorized NEC Corp. to publish TPC-C performance and price/performance results for the NEC Express5800/1160Xe. Price quotations contained in Appendix E correspond to the NEC Express5800/1160Xe server.

Parameters and Options

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

- *Database tuning options.*
- *Recovery/commit options.*
- *Consistency/locking options.*
- *Operating system and application configuration parameters.*
- *Compilation and linkage options and run-time optimizations used to create/install applications, OS, and/or databases.*

Appendix C contains the tunable parameters used in the TPC-C tests.

Configuration Diagrams

Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:

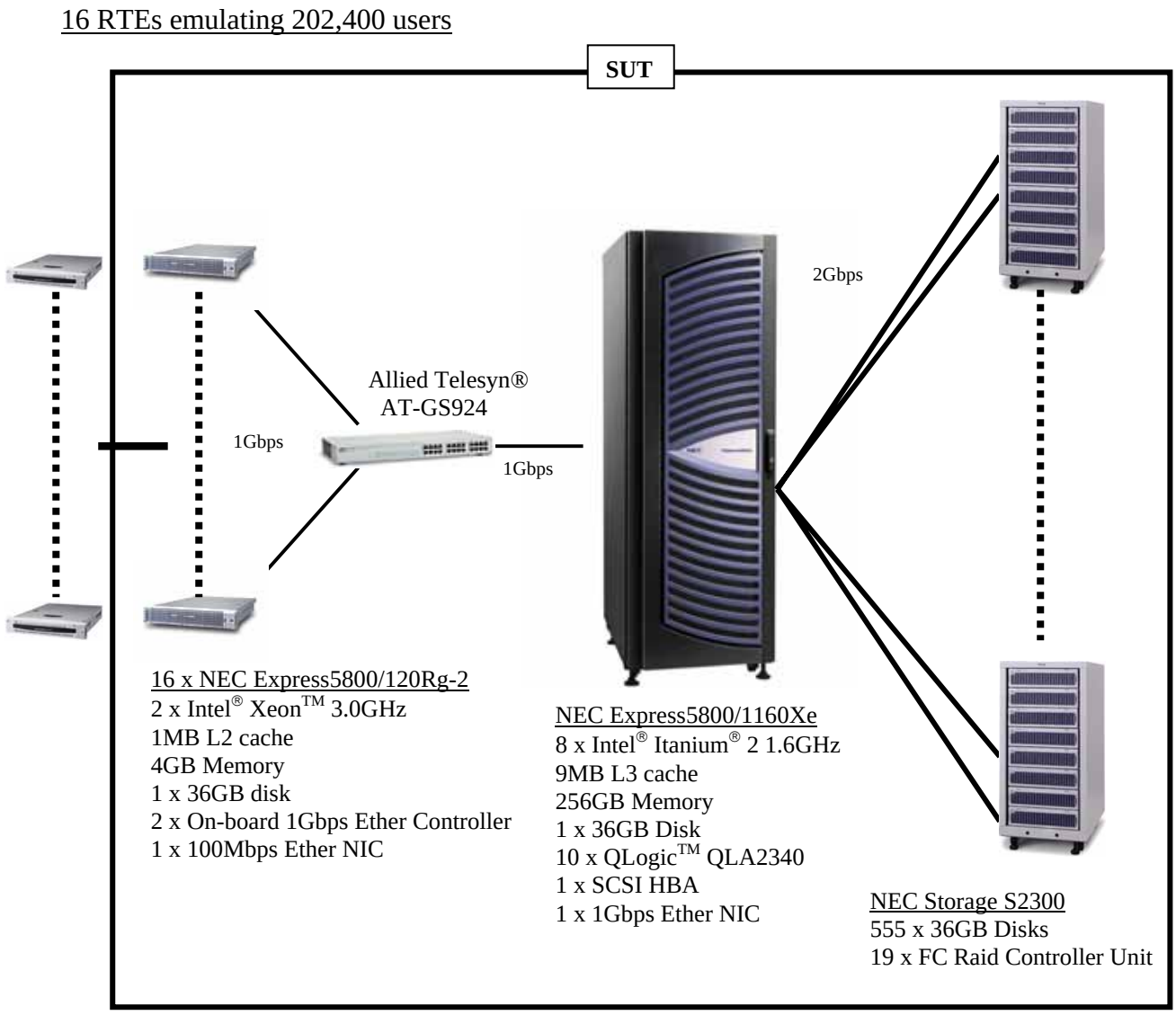
- *Number and type of processors/cores/threads.*
- *Size of allocated memory, and any specific mapping/partitioning of memory unique to the test.*
- *Number and type of disk units (and controllers, if applicable).*
- *Number of channels or bus connections to disk units, including their protocol type.*
- *Number of LAN (e.g., Ethernet) connections, including routers, workstations, terminals, etc., that were physically used in the test or are incorporated into the pricing structure (see Clause 8.1.8).*
- *Type and the run-time execution location of software components (e.g., DBMS, client processes, transaction monitors, software drivers, etc.).*

Figure 1.1 shows the measured configuration diagram.
Figure 1.2 shows the priced configuration diagram.

Measured Configuration

The following figure represents the measured configuration. The benchmark system used a remote terminal emulator (RTE) to initiate transactions and measure response times of transactions, as well as record various data for each transaction.

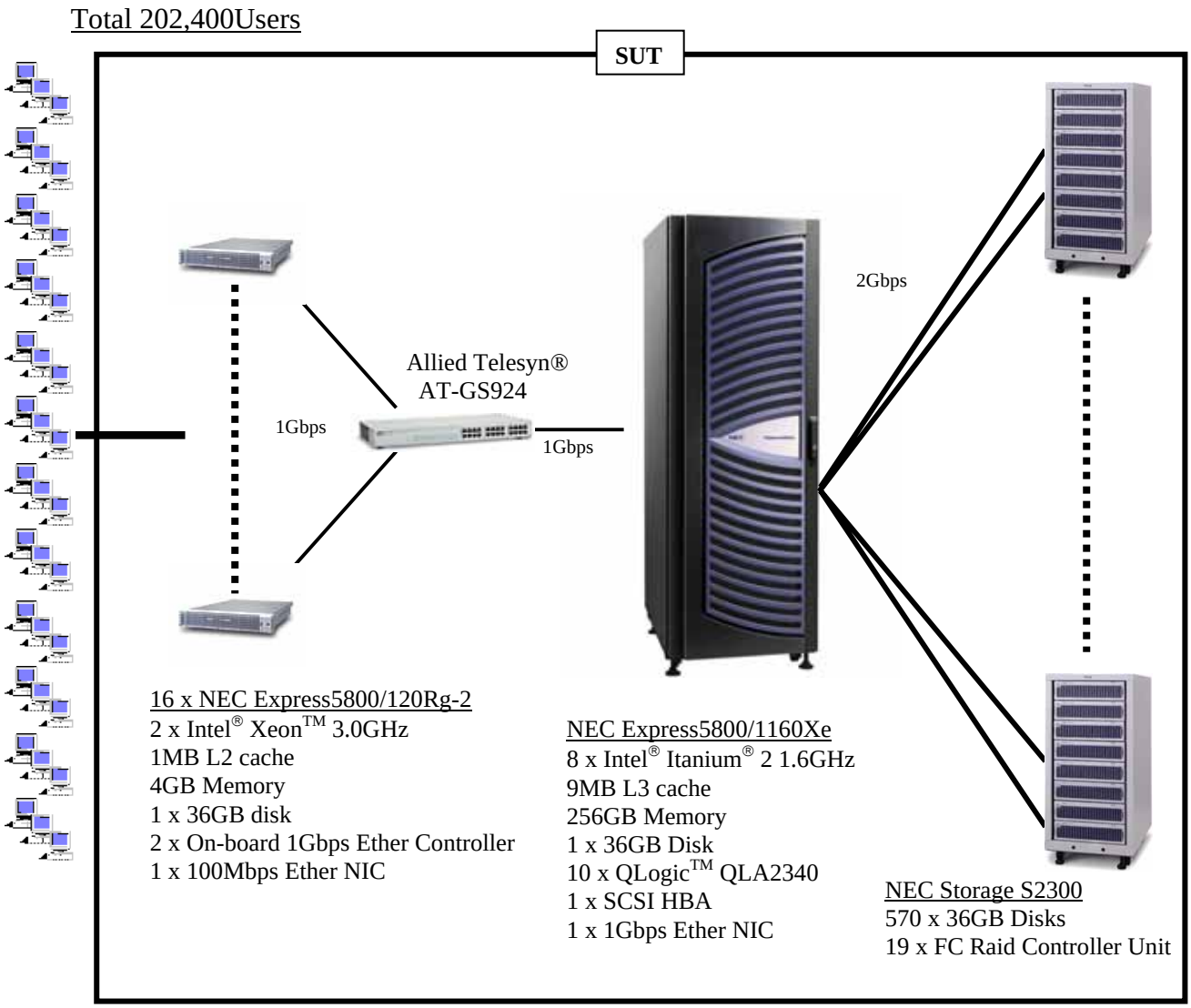
Figure1.1: Express5800/1160Xe, Measured Configuration Diagram



Priced System Configuration

The following figure depicts the priced system, whose cost determines the normalized price per tpmC reported for the test.

Figure1.2: Express5800/1160Xe, Priced Configuration Diagram



Clause 1 : Logical Database Design and Related Items

Table Definitions

Listings 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.

Table Organization

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

Appendix B contains the code used to define the physical organization of tables and indices

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 restriction 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 maximum key value for these new rows

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

Disclosure of Partitioning

While there are a few restrictions placed upon horizontal or vertical partitioning of tables and rows in the TPC-C benchmark (see Clause 1.6), any such partitioning must be disclosed.

Partitioning was not used on any table in this benchmark.

Replication of Tables

Replication of tables, if used, must be disclosed (see Clause 1.4.6).

No tables were replicated in this benchmark test.

Additional and/or Duplicated Attributes in any Table

Additional and/or duplicated attributes in any table must be disclosed along with a statement on the impact on performance (see Clause 1.4.7).

No duplications or additional attributes were used in this benchmark.

Clause 2 : Transaction and Terminal profiles Related Items

Random Number Generation

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

Ramdom numbers generation in RTE and DB generator were verified by auditor independently.

Terminal Input/Output Screen Layout

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

All screen layouts were verified to be exactly followed the specification by auditor.

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).

Each of five transaction types was tested by the auditor. The auditor verified that all the features specified in Clause 2.2.2.4 were provided.

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 applications is listed in Appendix A.

Transaction Profiles

. *The percentage of home and remote order-lines in the New-Order transactions must be disclosed.*

. *The percentage of New-Order transactions that were rolled back as a result of an unused item number must be disclosed.*

. *The number of items per orders entered by New-Order transactions must be disclosed.*

. *The percentage of home and remote Payment transactions must be disclosed.*

. *The percentage of Payment and Order-Status transactions that used non-primary key (C_LAST) access to the database must be disclosed.*

. *The percentage of Delivery transactions that were skipped as a result of an insufficient number of rows in the NEW-ORDER table must be disclosed.*

Table 1 shows the numerical quantities required by Clause 8.1.3.5 through 8.1.3.10.

Transaction Mix

The Mix (i.e. , percentages) of transaction types seen by the SUT must be disclosed.

Table 1 shows the mix of transaction types seen by the SUT during the reported measurement interval.

Following table summarizes the data required for disclosure in Clause 8.1.3.5 through 8.1.3.11

Table 1 Transaction Statistics

Statistic		Value
New Order	Home warehouse order lines	**98.99%**
	Remote warehouse order lines	**1.01%**
	Rolled back transactions	**0.99%**
	Average items per order	**9.9**
Payment	Home warehouse payments	**85.02%**
	Remote warehouse payments	**14.98%**
	Accessed by last name	**59.98%**
Order Status	Accessed by last name	**59.98%**
Delivery	Skipped deliveries	**0**
Transaction Mix	New Order	**44.81%**
	Payment	**43.07%**
	Delivery	**4.04%**
	Stock Level	**4.04%**
	Order Status	**4.04%**

Queuing Mechanism

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

The client application processes submitted delivery transactions to server software running on the client machines. These delivery servers were responsible for processing queued deliveries to submit to the database server.

The source code is listed in Appendix A.

Clause 3 : Transaction and System Properties 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.

The TPC Benchmark™ C Standard Specification defines a set of transaction processing system properties that a system under test (SUT) must support during the execution of the benchmark. Those properties are Atomicity, Consistency, Isolation and Durability (ACID). This section quotes the specification definition of each of those properties and describes the tests done as specified and monitored by the auditor, to demonstrate compliance.

Atomicity Tests

The system under test must guarantee that 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

Perform the Payment transaction for a randomly selected warehouse, district and customer(by customer number as specified in Clause 2.5.1.2) and verify that the records in the CUSTOMER,DISTRICT and WAREHOUSE tables have been changed appropriately.

The value of w_ytd, d_ytd, c_balance, c_ytd_payment and c_payment_cnt of a randomly selected warehouse, district, and customer were retrieved. The Payment transaction was executed on the same warehouse, district, and customer. The transaction was committed. The values w_ytd, d_ytd, c_balance, c_ytd_payment, and c_payment_cnt were retrieved again. It was verified that all values had been changed appropriately.

Aborted Transactions

Perform the Payment transaction for a randomly selected warehouse, district and customer (by customer number as specified in Clause 2.5.1.2) and substitute a ROLLBACK of the transaction for the COMMIT of the transaction. Verify that the records in CUSTOMER, DISTRICT and WAREHOUSE tables have Not been changed.

The value of w_ytd, d_ytd, c_balance, c_ytd_payment and c_payment_cnt of randomly selected warehouse, district, and customer were retrieved. The Payment transaction was executed on the same warehouse, district, and customer. The transaction was rolled back. The values of w_ytd, d_ytd, c_balance, c_ytd_payment, c_payment_cnt were retrieved again. It was verified that none of the values had changed.

Consistency Tests

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 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 over 5 minutes under 202,400 users condition on a 21,600 warehouse database. The transaction rate was within 10% of the reported tpmC rate. A checkpoint generated in the test. The shell script of consistency was executed before and after the run. The result of the same queries verified that the database remained consistent after the run.

Isolation Tests

Sufficient conditions must be enabled at either the system or application level to ensure the required isolation level is obtained.

Isolation tests one through nine were executed using shell scripts to issue queries to the database. Each script included timestamps to demonstrate the concurrency of operations. The results of the queries were captured to files. The captured files were verified to demonstrate the required isolation had been met.
Case D was followed for Isolation Test 7.

Durability Tests

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

- *Permanent irrecoverable failure of any single durable medium during the Measurement Interval containing TPC-C database tables or recovery log data.*
- *Instantaneous interruption (system or subsystem crash/system hang) in processing which causes all or part of the processing of atomic transactions to halt.*
- *Failure of all or part of memory(loss of contents)*

Instantaneous Interruption and Loss of Memory

Because the loss of power erases the contents of memory, both of instantaneous interruption and loss of memory were combined into a single test. The following steps were performed on 21,600 warehouse database.

1. A sum of D_NEXT_O_ID of all rows in the district table was taken.
2. 202,400 users were logged in to the database and kept running transactions about 5 minutes in steady state.
3. The system was powered off.
4. The RTE was shut down.
5. The system was powered up. Oracle® Database 10g was stared up and database was recovered.
6. A new count of D_NEXT_O_ID was taken.
7. This number was compared with the number of new orders reported by the RTE.

Loss of Log and Data Disk

Loss of log and loss of data disk were combined to a one test and were demonstrated on a 21,600 warehouse database. The standard driving mechanism was used to generate the transaction load of 21,600 users for the test. To demonstrate recovery from a permanent failure of durable media containing TPC-C tables and log, the following steps were performed.

1. A sum of D_NEXT_O_ID of all rows in the district table was taken.
2. 21,600 users were logged in to the database and kept running transactions about 5 minutes in steady state.
3. Removed one of RAID5 log disk. The running couninued without any interruptions.
4. One disk drive for data part in the array was removed causing Oracle® Database 10g error. Shut down Oracle® Database 10g.
5. The 21,600 warehouse database was restored from backup.
6. The database was started and recovered by using recover command of Oracle® Database 10g.
7. A new count of D_NEXT_O_ID was taken.
8. This number was compared with the number of new orders reported by the RTE.

Loss of mirrored write-back cache

The Fibre Array system used for this benchmark has integrated feature of mirrored write-back cache. When a LUN is configured to enable write-back caching, the data on the cache is automatically mirrored on the RAMs in two controller modules, which are powered and protected from loss of power by the controller BBU. Loss of write-back cache was demonstrated on 21,600 warehouse database, by pulling off one of the controller module.

1. A sum of D_NEXT_O_ID of all rows in the district table was taken.
2. 202,400 users were logged in to the database and kept running transactions about 5 minutes in steady state.
3. A controller module, which manages write-back cache of mirrored drives, was pulled off.
4. Fibre system reported system error resulted in IO error.
5. Contents on mirrored cache on another controller were automatically saved to temporal disk area and the Fibre Array house was eventually stopped. System was shut down.
6. The Fibre array was re-powered.
7. Saved data was automatically restored to mirrored cache and flushed to drives.
8. Oracle® Database 10g was started up and database was recovered.
9. A new count of D_NEXT_O_ID was taken.
10. This number was compared with the number of new orders reported by the RTE.

Clause 4 : Scaling and Database Population Related Items

Initial Cardinality of Tables

The cardinality (e.g., the number of rows) of each table, as it existed at the start of the benchmark run (see Clause 4.2), must be disclosed. If the database was over-scaled and inactive rows of the WAREHOUSE table were deleted (see Clause 4.2.2), the cardinality of the WAREHOUSE table as initially configured and the number of rows deleted must be disclosed.

The TPC-C database was originally built with 21,600 warehouses.

Table 2 Number of Rows for Server

Table	Cardinality as benchmarked
Warehouse	20,240
District	216,000
Customer	648,000,000
History	648,000,000
Orders	648,000,000
New Order	194,400,000
Order Line	6,480,188,608
Stock	2,160,000,000
Item	100,000
Deleted Warehouse Rows	1,360

1,360 warehouses ware deleted before measurement. 20,240 warehouses and their associated tables ware accessed during the measurement.

Constant Value for the NURand function

The following values were used as constant value inputs to the NURand function for this benchmark.

C_LAST (Build)	**1**
C_LAST (RUN)	**66**

Distribution of Tables and Logs

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

Table 3 depicts the distribution of the database over the disks of the tested system.
Figure 1.1, 1.2 shows the disk configuration for measured and priced system.

Table 3 : Data Distribution

HBA#	storage system#	SP#	DEU#	# of Disks	RAID Level	Capacity (GB)	Partition1
Partitions for DB Log							
0	0	0	0	15	5	465.64	log_1 (460GB)
			1	15	5	465.64	log_2 (460GB)

HBA#	storage system#	SP#	DEU#	# of Disks	RAID Level	Capacity (GB)	Partition1-11	Partition12 (10GB)
Partitions for DB Data								
1	1	0	0	15	0	498.91	*1	control_001
			1	15	0	498.91		
	2	0	0	15	0	498.91		WARE_0
			1	15	0	498.91		
2	3	0	0	15	0	498.91		control_002
			1	15	0	498.91		
	4	0	0	15	0	498.91		DIST_0
			1	15	0	498.91		
3	5	0	0	15	0	498.91		SYSTEM
			1	15	0	498.91		
	6	0	0	15	0	498.91		ITEM_0
			1	15	0	498.91		
4	7	0	0	15	0	498.91		SYSAUX
			1	15	0	498.91		
	8	0	0	15	0	498.91		IWARE_0
			1	15	0	498.91		
5	9	0	0	15	0	498.91		UNDO_TS
			1	15	0	498.91		
	10	0	0	15	0	498.91		IDIST_0
			1	15	0	498.91		
6	11	0	0	15	0	498.91		SP_0
			1	15	0	498.91		
	12	0	0	15	0	498.91		IITEM_0
			1	15	0	498.91		
7	13	0	0	15	0	498.91		
			1	15	0	498.91		
	14	0	0	15	0	498.91		
			1	15	0	498.91		
8	15	0	0	15	0	498.91		
			1	15	0	498.91		
	16	0	0	15	0	498.91		
			1	15	0	498.91		
9	17	0	0	15	0	498.91		
			1	15	0	498.91		

HBA#	storage system#	SP#	DEU#	# of Disks	RAID Level	Capacity (GB)	Partition1 (9G)	Partition2 (18G)	Partition3 (53G)
Partitions for DB Data									
9	18	0	0	15	0	498.91	NORD_0	ICUST1_0	ISTOK_0

*1. Composition from Partition1 to Partition11 is shown in the following table

Partition1-11

partiton#	tablespace name
Partition1	STOK_0 (8GB)
Partition2	STOK_0 (8GB)
Partition3	STOK_0 (8GB)
Partition4	CUST_0 (7GB)
Partition5	CUST_0 (7GB)
Partition6	CUST_0 (7GB)
Partition7	HIST_0 (3GB)
Partition8	ORDR_0 (35GB)
Partition9	ICUST2_0 (2GB)
Partition10	IORDR2_0 (2GB)
Partition11	TEMP_0(4G)

Type of Database

A statement must be provided that describes:

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

No partitioning or replication was used.

60-Days 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 (see Clause 4.2.3).

The detail of 60-day space calculation is shown in Appendix D.
To calculate the space required to sustain the database log for 8 hours of growth at steady state, the following steps were followed:

1. Logfile usage was obtained from system view.
2. The space used was calculated as the difference between the first and second query.
3. The number of NEW-ORDERS was verified from an RTE report covering the entire run.
4. The space used was divided by the number of NEW-ORDERS giving a space used per NEW-ORDER transaction.
5. The space used per transaction was multiplied by the measured tpmC rate times 480 minutes.

The results of the above steps yielded a requirement of 578.59 GB of logspace (i.e. 619.92 GB of RAID5 transaction log volume) to be available to sustain 8 hours. Total space of priced disks for the transaction log volume was 1496.70 GB. It indicates that enough storage was configured to sustain 8 hours of growth.

The same methodology was used to compute growth requirements for dynamic tables Order,Order-Line and History.

Clause 5 : Performance Metrics and Response Time Related Items

Throughput

Measured tpmC must be reported

Table 4 : Measured tpmC

254,471 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 : Response Times (in seconds)

Type	Average	Maximum	90th %
New-Order	0.23	117.21	0.25
Payment	0.23	117.25	0.25
Interactive Delivery	0.11	0.19	0.11
Stock Level	0.21	97.14	0.24
Order Status	0.22	93.50	0.25
Deferred Delivery	0.02	0.48	0.05
Menu	0.11	0.24	0.11

Keying and Think Times

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

.

Table 6 : Keying Times

Type	Average	Minimum	Maximum
New-Order	18.01	18.01	18.01
Payment	3.01	3.01	3.01
Interactive Delivery	2.01	2.01	2.01
Stock Level	2.01	2.01	2.01
Order Status	2.01	2.01	2.01

Table 7 : Think Times

Type	Average	Minimum	Maximum
New-Order	12.08	0.00	120.7
Payment	12.07	0.00	120.7
Interactive Delivery	5.07	0.00	50.7
Stock Level	5.08	0.00	50.7
Order Status	10.08	0.00	100.7

Response Time Frequency Distribution Curves

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

Figure 2.1 : New-Order Response Time Distribution

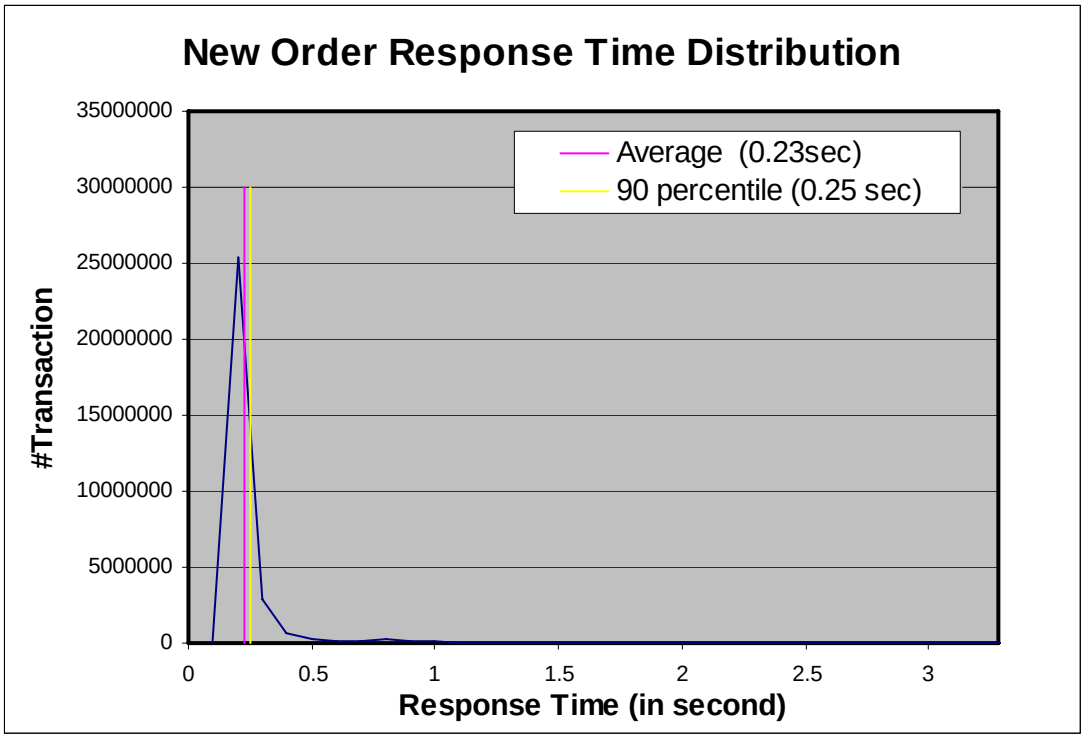


Figure 2.2 : Payment Response Time Distribution

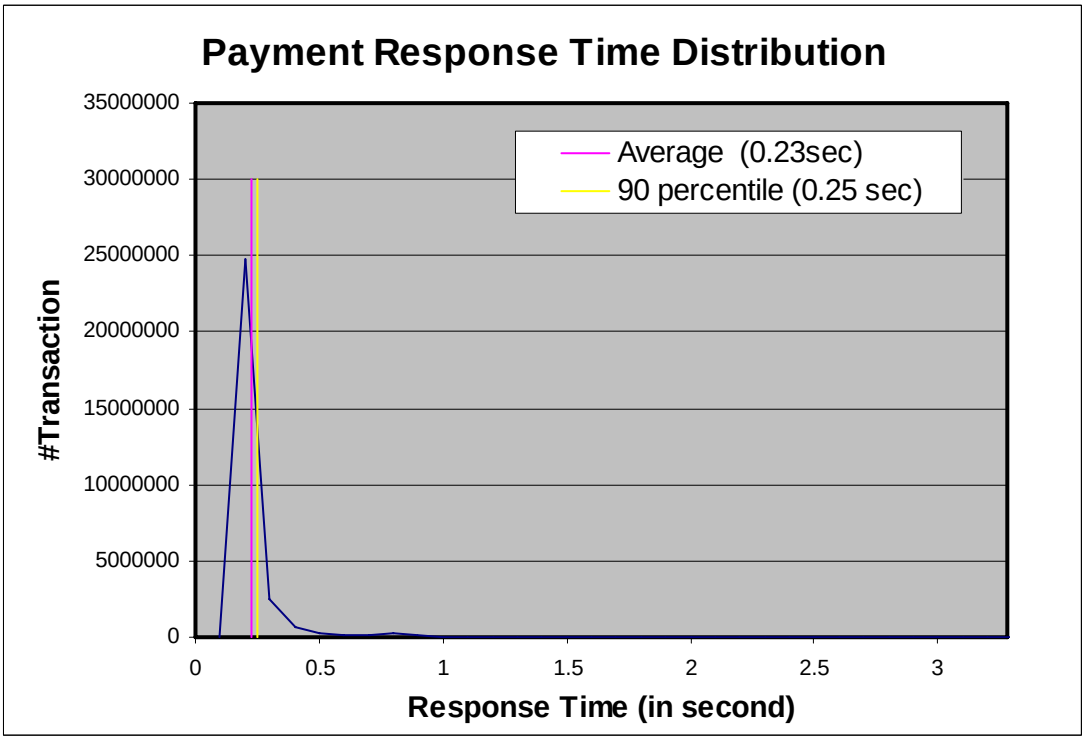


Figure 2.3 : Order-Status Response Time Distribution

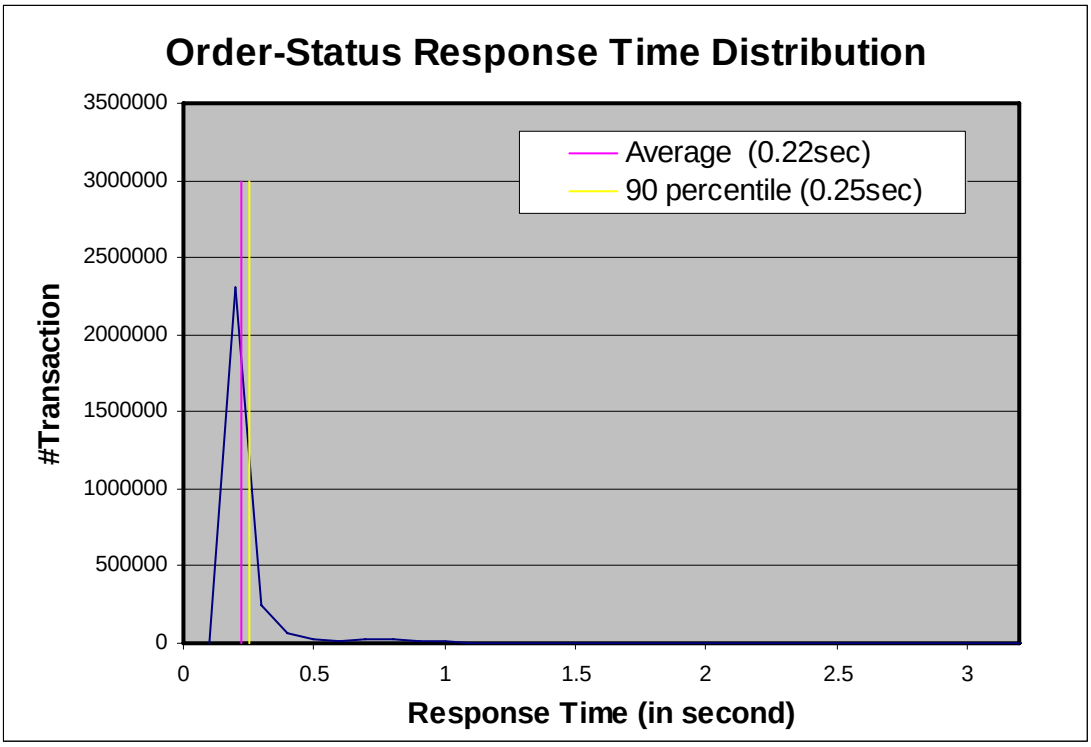


Figure 2.4 : Delivery Response Time Distribution

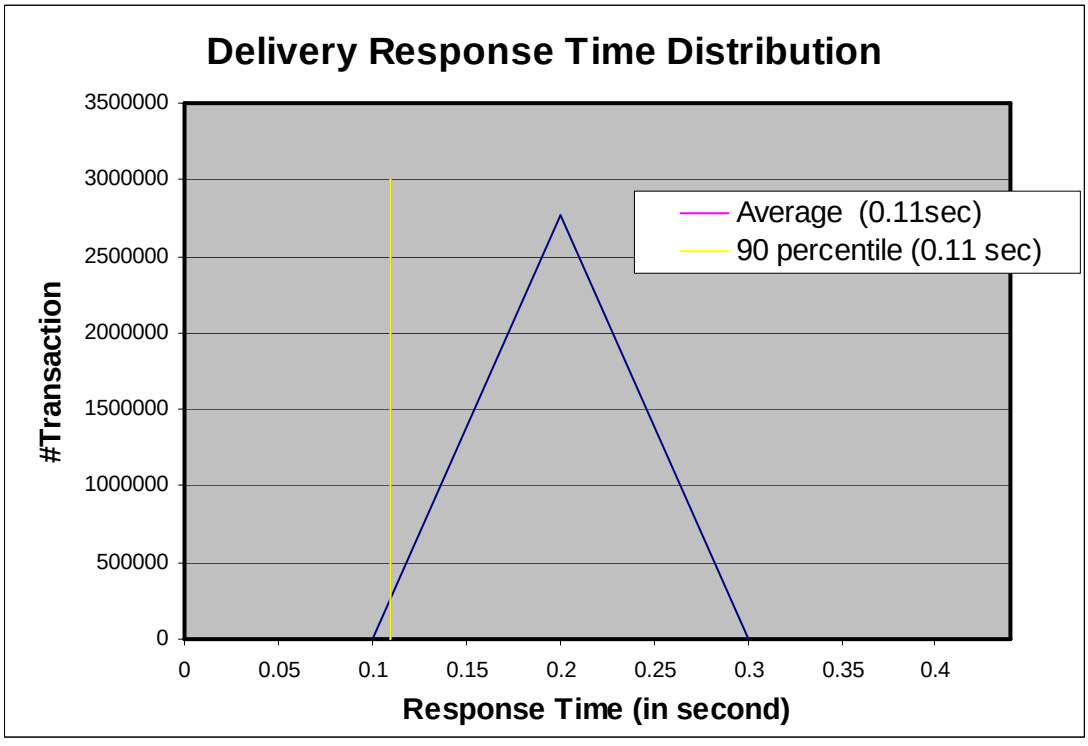
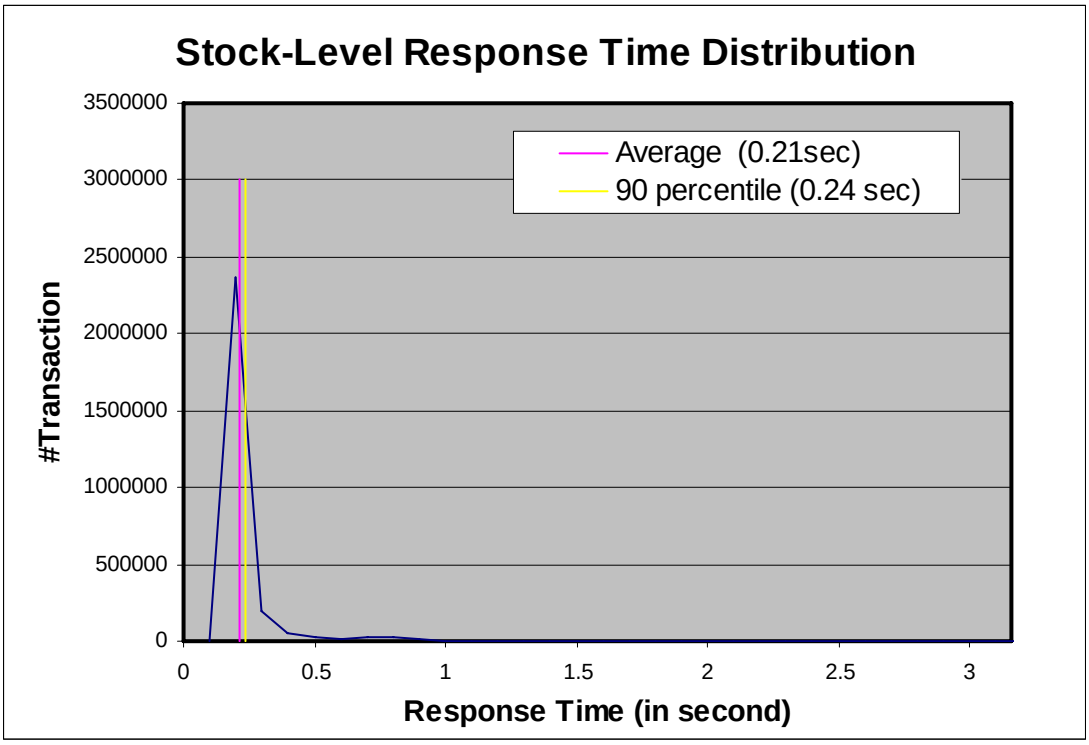


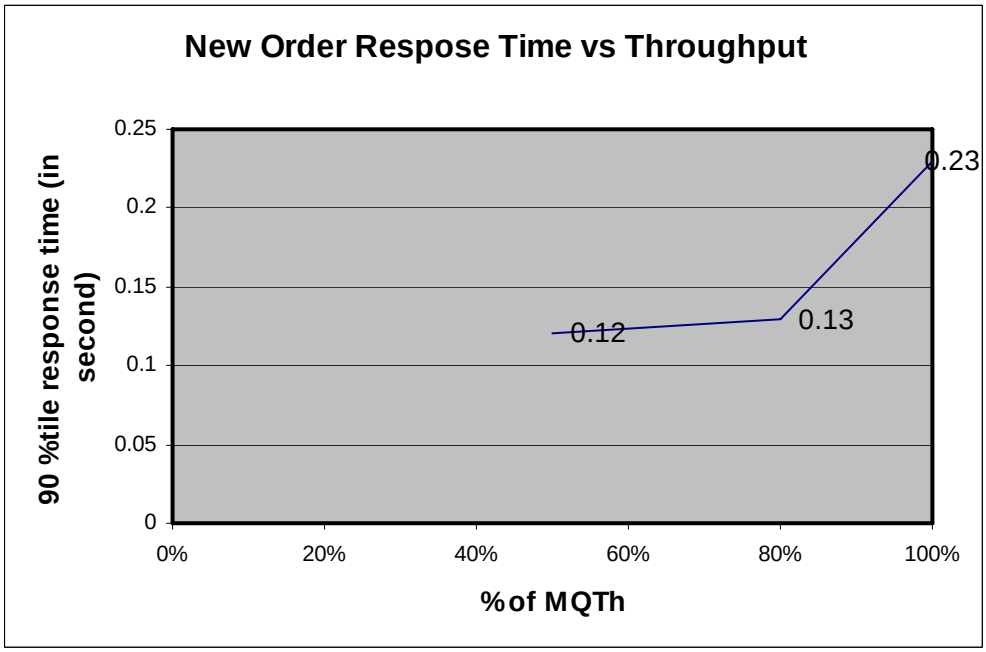
Figure 2.5 : Stock-Level Response Time Distribution



Response time versus Throughput Curve

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

Figure 2.6 Response Time vs. Throughput Curve



New-Order Think Time Frequency Distribution

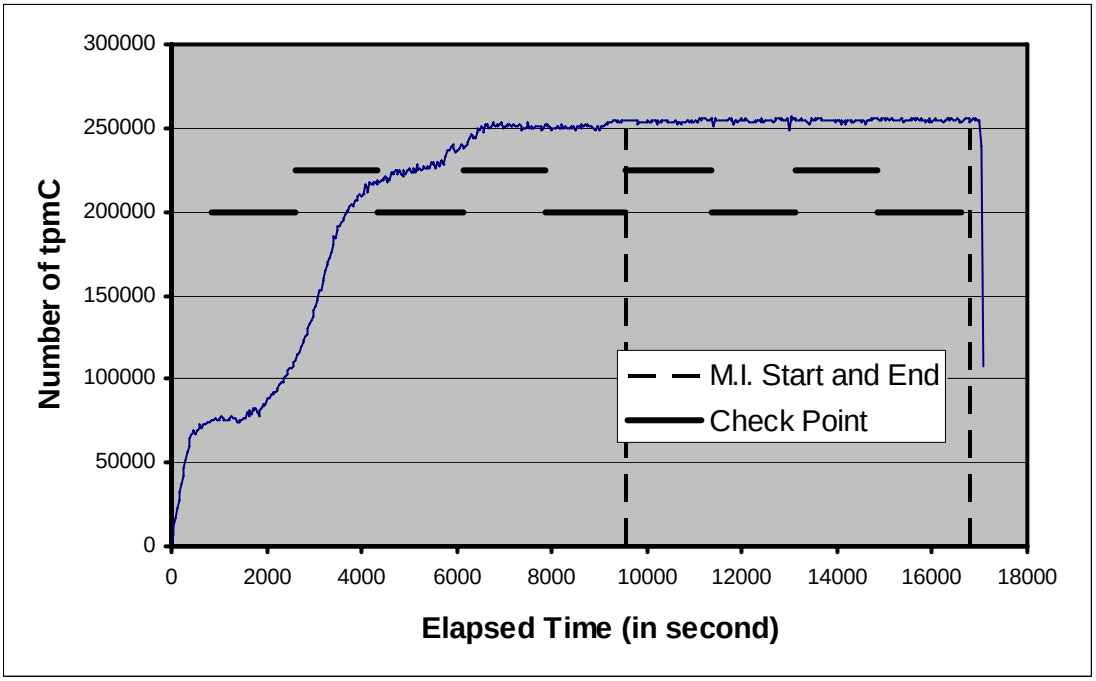
Think Time frequency distribution curves (see Clause 5.6.3) must be reported for the New-Order transaction.

Figure 2.7 New-Order Think Time

New-Order Throughput vs. Elapsed Time

A graph of throughput versus elapsed time (see Clause 5.6.4) must be reported for the New-Order transaction.

Figure 2.8 New Order Throughput vs. Elapsed Time



Steady State

The method used to determine that the SUT had reached a steady state prior to commencing the measurement interval (see Clause 5.5) must be described.

Steady state was confirmed by the throughput data collected during the run and graphed in Figure 2.8.

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.

A checkpoint in Oracle® Database 10g writes to disk all updated memory pages that have not been yet actually written to disk. The TPC-C benchmark was setup to automatically checkpoint every 30 minutes. 4 checkpoints occurs during the measurement interval. The incremental checkpoint duration during the measurement is 1753 second.

Measurement Period Duration and Checkpoint Duration

. *The start time and duration in seconds of at least the four (4) longest checkpoints during the Measurement Interval must be disclosed (see Clause 5.5.2.2 (2)).*

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

	Start	End	Duration (in second)
M.I.	15:43:00	17:43:30	7230
1st Checkpoint	15:43:47	16:12:59	1752
2nd Checkpoint	16:12:59	16:42:11	1752
3rd Checkpoint	16:42:11	17:11:26	1755
4th Checkpoint	17:11:26	17:40:40	1754

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 entered 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.*

The above statistics are disclosed in Table 1.

Checkpoint Count and Location

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

There was five checkpoints before measurement and four checkpoints during measurement.
The time of the first checkpoint during the measurement interval is 7 seconds after the start of the measurement, and the average checkpoint interval is 1753 seconds.

Clause 6 : SUT,Driver,and Communication Definition Related Items

Descriptions of RTE

The RTE input parameters, code fragments, functions, etc. used to generate each transaction input field must be disclosed.

The RTE used was NEC proprieta1ly. The RTE input parameters are listed in Appendix C.

Loss of Terminal Connections

The number of terminal connections lost during the Measurement Interval must be disclosed (see Clause 6.6.2).

No terminal connections were lost.

Emulated Components

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

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. One tenth of a second (100 milli seconds) was added to each transaction time to compensate for the overhead of the Web browser.

Functional Diagrams and Detail of Driver System

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 software and hardware functionality being performed on the Driver System, and its interface to the SUT must be disclosed (see Clause 6.6.3.6).

The diagrams in figure 1.1 and 1.2 show the tested and priced benchmark configurations.

Network configurations and Driver system

The network configurations of both the tested services and the 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 (see Clause 6.6.4).

Figure 1.1 and 1.2 in this report has the network configurations of both the tested system and the priced system.

Network Bandwidth

The bandwidth of the network(s) used in the tested/priced configuration must be disclosed.

The database server contains one Gigabit Ethernet controller. This adapter is connected to one Gigabit Ethernet switch. 16 front-end clients contain two on-board Gigabit Ethernet controllers and one 100Mbps Ethernet controller. One on-board Gigabit Ethernet controller is connected to the Gigabit Ethernet switch. The other on-board Giagbit Ethernet controller and one 100Mbps Ethernet controller are connected to RTE system. The network bandwidth between RTE system and the front-end clients is 100Mbps. 32 segments are used for the connection of this tested configuration.

Operator Intervention

If the configuration requires operator intervention (see Clause 6.6.6), the mechanism and the frequency of this intervention must be disclosed.111

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

Clause 7 : Pricing Related Items

Hardware and Software Components

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(s) 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 justification of any discounts applied must be disclosed in the price sheet. Sufficient detail of what items are being discounted and by how much they are being discounted must be provided so that the discount amount used in the computation of the total system cost can be independently reproduced.

The detailed list of all hardware and software for the priced configuration is listed in the system pricing summary.

Availability

The committed delivery date for general availability (availability date) of products used in the price calculations must be reported. When the priced system includes products with different availability dates, the reported availability date for the priced system must be the date at which all components are committed to be available. This single date must be reported on the first page of the Executive Summary. All availability dates, whether for individual components or for the SUT as a whole, must be disclosed to a precision of one day.

The total system as priced will be available 15-Mar-2006.

- "Red Hat Enterprise Linux AS (Standard)" will be available by 15-Mar-2006.
- The other products are already available.

Throughput, and Price Performance

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

- Maximum Qualified Throughput : 254,471tpmC
- Price per tpmC : $5.32 per tpmC
- Total 3-year cost of ownership : $1,352,709 USD

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 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:

- 8 Oracle® Database 10g Enterprise Edition, Processor License for 3 years
- 3 Red Hat Enterprise Linux AS (Standard)
- 48 Red Hat Enterprise Linux ES (Standard)
- 16 BEA® Tuxedo Core Functionality Services (CFS-R)

System Pricing

System pricing should include subtotals for the following components: Server Hardware, Server Software, Client Hardware, Client Software, and Network Components. Clause 6.1 describes the Server and Client components.

System pricing must include line item indication where non-sponsoring companies' part numbers are used. System pricing must also include line item indication of third party pricing.

A detailed list of all hardware and software, including the 3-year price, is provided in the Executive Summary at the front of this report. All third-party quotations are included in Appendix E at the end of this document.

Clause 8 : Audit Related Items

Auditor's Report

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

Next page contains the complete independent auditor's report by Francois Raab of Info Sizing Inc. for the test described in this report.

Availability of the Full Disclosure Report

The Full Disclosure Report must be readily available to the public at a reasonable charge, similar to charges for similar documents by that 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 Ruger St. (surface)
P.O. Box 29920 (mail)
San Francisco, CA 94129-0920
Voice: 415-561-6272
Fax: 415-561-6120
Email: info@tpc.org

Auditor's letter

INFO SIZING

CERTIFIED AUDITOR

Benchmark Sponsor: Jun Suzuki
NEC Corporation
1-10 Nisshincho
Fuchu-City, Tokyo 183-8501, Japan

January 12, 2006

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

Platform: NEC Express5800/1160Xe c/s
Operating system: Red Hat Enterprise Linux AS 4
Database Manager: Oracle Database 10g Enterprise Edition
Transaction Manager: BEA Tuxedo 8.1

The results were:

CPU's Speed	Memory	Disks	NewOrder 90% Response Time	tpmC
Server: NEC Express5800/1160Xe				
8 x Itanium 2 (1.6 GHz)	256 GB (9 MB L3)	570 x 36 GB FC 1 x 36 GB SCSI	0.25 Seconds	254,471.41
Sixteen Client: Express5800/120Rg-2 (each with)				
2 x Xeon (3.0 GHz)	4 GB (1 MB cache)	1 x 36 GB SCSI	n/a	n/a

In my opinion, these performance results were produced in compliance with the TPC requirements for the benchmark.

The following verification items were given special attention:

- The transactions were correctly implemented
- The database records were the proper size
- The database was properly scaled and populated

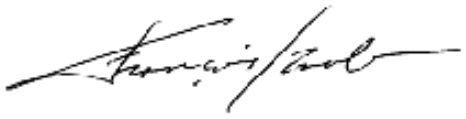
125 West Monroe Street • Colorado Springs, CO 80907 • 719-473-7555 • www.sizing.com

- The ACID properties were met
- Input data was generated according to the specified percentages
- The transaction cycle times included the required keying and think times
- The reported response times were correctly measured.
- At least 90% of all delivery transactions met the 80 Second completion time limit
- All 90% response times were under the specified maximums
- The measurement interval was representative of steady state conditions
- The reported measurement interval was 120 minutes
- Four checkpoints were taken during the measurement interval
- The 60 day storage requirement was correctly computed
- The system pricing was verified for major components and maintenance

Additional Audit Notes:

None.

Respectfully Yours,



François Raab, President

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

Appendix A : Application Source Code

A1. Front-End Source Code, Scripts

tpcc.c

```
/*============================================================
======+
|       Copyright (c) 1997  Oracle Corp, Redwood Shores, CA      |
|                    All Rights Reserved                    |

+============================================================
======+
| FILE:  TPCC.C                                          |
| DESCRIPTION: Main module for TPCC.DLL                        |
| Master created: 15 Apr 97                              |

*============================================================
===== */
#ifdef TOPEND
#define TP_MT_SOURCE
#define STRICT
#endif



#define LOCAL_ALLOC 1   /* force local alloc to be true */

#define APACHE                  /* convert this source to apache module */

//#include <windows.h>
//#include <process.h>
#include <stdio.h>
#include <stdarg.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/timeb.h>
//#include <io.h>
#include <assert.h>
//#include <tchar.h>

#include "tpccerr.h"
#include "tpcc_info.h"
//#include "httpext.h"


//#include "getval.h"

#ifdef TOPEND
#include <tp_csi.h>
BOOL TP_InitDone = FALSE;
#define DllExport _declspec(dllexport)
DllExport LONG tp_ChangeToGroup(LPTSTR,DWORD,DWORD);

tp_dif_structs_t *client_dif;

/*
 * Topend structure offset.
 */
#define TOPEND_STRUCT_OFFSET 4124

#endif

#ifdef TUX
#include <tmenv.h>
#include <xa.h>
#include <atmi.h>
#include <Uunix.h>
#endif


//static void UtilStrCpy(char * pDest, char * pSrc, int n);

#include "mod_tpcc.h"
#include "tpccapi.h"

static TPINIT   *tpinf;
//static int  *    TLSIsTpInitedKey;

static DWORD    TLSNewOrderKey;
```

```
static DWORD    TLSPaymentKey;
static DWORD    TLSOrderStatusKey;
static DWORD    TLSDeliveryKey;
static DWORD    TLSStockLevelKey;
static int          ThrTpInit();

char    szServer[32]    = { 0 };        //global variables used with this DLL
char    szUser[32]      = { 0 };
char    szPassword[32]  = { 0 };
char    szDatabase[32]  = "tpcc";
char    szIID[16][8]    =
{"IID00*","IID01","IID02","IID03","IID04","IID05","IID06","IID07","IID08","IID0
9","IID10","IID11","IID12","IID13","IID14","IID15"};
char    szSP[16][8]     =
{"SP00*","SP01","SP02","SP03","SP04","SP05","SP06","SP07","SP08","SP09","SP
10","SP11","SP12","SP13","SP14","SP15"};
char    szQty[16][8]    =
{"Qty00*","Qty01","Qty02","Qty03","Qty04","Qty05","Qty06","Qty07","Qty08","Qt
y09","Qty10","Qty11","Qty12","Qty13","Qty14","Qty15"};

BOOL    bLog            = FALSE;
BOOL    dLog            = FALSE;



#ifdef TOPEND
int         wait_time;
int     inactivity_time = 1500;
#endif

BOOL    bGeneric        = FALSE;
/* add structure to get delivery time stamp */
#ifdef WIN32
struct _timeb  timebuffer;
#else
struct timeb  timebuffer;
#endif

int         iThreads            = 5;
int         iMaxWareHouses  = 625;
int         iDelayMs            = 100;
short   iDeadlockRetry  = (short)3;
short   iMaxConnections = (short)625;
int         iErrVal             = 0;

//allowable client command strings i.e. CMD=command
char *szCmds[] =
{
        "..NewOrder..", "..Payment..", "..Delivery..", "..Order-Status..", "..Stock-
Level..", "..Exit..",
        "Submit", "Begin", "Process", "Menu", "Clear", ""
};

//defined command string functions, called via CMD=command http string from html
client.

void (*DoCmd[])(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId) =
{
        NewOrderForm,
        PaymentForm,
        DeliveryForm,
        OrderStatusForm,
        StockLevelForm,
        Exitcmd,
        SubmitCmd,
        BeginCmd,
        ProcessCmd,
        MenuCmd,
        ClearCmd
};


//Terminal client id structure and interface defination
//TERM  Term = { 0, 0, 0, FALSE, NULL, TermInit, TermAllocate, TermRestore,
TermAdd, TermDelete };

TERM *Term=NULL;

apr_status_t tpcc_shm_create(tpcc_shm_t* m, apr_pool_t* p)
{
        apr_status_t s= apr_shm_create(&(m->shm),m->size, m->name, p);
//      if (s != APR_SUCCESS)
        if (dLog)
                {
                        FILE *fp;
```

```
                        fp = fopen("/tmp/errlog", "ab");
                        fprintf(fp, "create_shm *init Moudle pid=%d,
thread_id=%d, name=%s\n", getpid(),GetCurrentThreadId(),m->name);
                        fclose(fp);
                }
        return s;
}


/* change shm segment size and preserve contents
 * the old shm segment must already attached before this call
 */
void* realloc_shm(tpcc_shm_t*m,apr_size_t new_size, apr_pool_t* p)
{
        apr_size_t old_size=m->size;
        apr_shm_t* new_shm;
        void* old, *new,*temp;
        if (dLog)
                {FILE *fp;      fp = fopen("/tmp/errlog", "ab");
                fprintf(fp, "* In realloc before malloc, size=%d\n",m->size);
                fclose(fp);}
        temp = malloc(old_size);
        if (temp)
                if (dLog)
                {FILE *fp;      fp = fopen("/tmp/errlog", "ab");
                fprintf(fp, "* In realloc malloc done\n");
                fclose(fp);}
        old=apr_shm_baseaddr_get(m->shm);
                if (dLog)
                {FILE *fp;      fp = fopen("/tmp/errlog", "ab");
                fprintf(fp, "* In realloc old_base_addr, shm=%x\n",m->shm);
                fclose(fp);}
        memcpy(temp, old, m->size);

//      apr_shm_detach(m->shm);
        apr_shm_destroy(m->shm);
        unlink(m->name);

        m->size=new_size;

        apr_shm_create(&(m->shm),m->size, m->name,p);

        new= apr_shm_baseaddr_get(m->shm);
        if (old_size>new_size)
                memcpy(new,temp,m->size);
        else
                memcpy(new,temp,old_size);
        free(temp);
                if (dLog)
                {FILE *fp;      fp = fopen("/tmp/errlog", "ab");
                fprintf(fp, "* In realloc new size=%d, shm=%x\n",m->size,m-
>shm);
                fclose(fp);}
        m->addr=new;
        return new;
}

char* _strupr(char* str)
{
        while(*str)
        {
                *str=toupper(*str);
                str++;
        }
        return str;
}

char *_strtime(char* str)
{
        time_t now;
        struct tm *local;
        time(&now);
        local=localtime(&now);
        strftime(str, 9, "%T",local);
}


//for debug
void debugdll(char *filename, char *format, ...)
{
        FILE *fp;
        va_list args;
        char buf[4096];
        int len;
        va_start( args, format );
        _strtime( buf );
```

```
        strcat( buf," ");
        len = strlen( buf );
#ifdef WIN32
        (void)_vsnprintf( buf+ len, sizeof( buf) - len - 1, format, args);
#else
        (void)vsnprintf( buf+ len, sizeof( buf) - len - 1, format, args);
#endif
        buf[sizeof( buf )- 1]= '\0';
        va_end( args );

        if (fp = fopen(filename, "ab"))
          {
                fprintf(fp,"%s.\n", buf);
                fflush(fp);
                fclose(fp);
        }
}

#ifdef WIN32
void WriteMessageToEventLog(LPTSTR lpszMsg)
                //XYZ//
{
  TCHAR   szMsg[256];
  HANDLE  hEventSource;
  LPTSTR  lpszStrings[2];

  // Use event logging to log the error.
  //
  hEventSource = RegisterEventSource(NULL, TEXT("TPCC.DLL"));

  _stprintf(szMsg, TEXT("Error in TPCC.DLL: "));
  lpszStrings[0] = szMsg;
  lpszStrings[1] = lpszMsg;

  if (hEventSource != NULL)
        {
    ReportEvent(hEventSource, // handle of event source
      EVENTLOG_ERROR_TYPE,  // event type
      0,              // event category
      0,              // event ID
      NULL,               // current user's SID
      2,              // strings in lpszStrings
      0,              // no bytes of raw data
      (LPCTSTR *)lpszStrings,    // array of error strings
      NULL);              // no raw data

    (VOID) DeregisterEventSource(hEventSource);
  }
}
#else
void write_err_log(request_rec *r, const char * szMsg)
{
  ap_log_rerror(APLOG_MARK, APLOG_ERR, 0, r, "%s",szMsg);
}
#endif


//welcome to tpc-c html form buffer, this is first form client sees.
static char     *szWelcomeForm =        "<HTML>"

          "<HEAD><TITLE>Welcome To TPC-C</TITLE></HEAD><BODY>"
                                                                  "Please
Identify your Warehouse and District for this session.<BR>"
                                                                  "<FORM
ACTION=\"tpcc.dll\" METHOD=\"GET\">"
                                                                  "<INPUT
TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"0\">"
                                                                  "<INPUT
TYPE=\"hidden\" NAME=\"ERROR\" VALUE=\"0\">"
                                                                  "<INPUT
TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"1\">"
                                                                  "<INPUT
TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"-2\">"
                                                                  "<INPUT
TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"0\">"
                                                                  "Warehouse
ID <INPUT NAME=\"w_id\" SIZE=6><BR>"
                                                                  "District ID
<INPUT NAME=\"d_id\" SIZE=2><BR>"
                                                                  "<HR>"
                                                                  "<INPUT
TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Submit\">"
                                                                  "
"
                                                                  "
"
```

```
                                                                "
"                                                                   static  CRITICAL_SECTION        CriticalSection;
                                                                "   static  CRITICAL_SECTION       ErrorLogCriticalSection;
"
                                                                "   int     bTpccExit;      //exit delivery disconnect loop as dll exiting.
"
                                                                "   #ifdef TOPEND
"                                                                   typedef struct DTEntry {
                                                                "     HANDLE      hSync;
"                                                                     struct DTEntry *next;
                                                                "     tp_dif_structs_t *dif_struct;
"                                                                     char        *message_buffer;
                                                                "     long        *usr_dlg_ptr;
"                                                                     long        usr_dlg_storage;
                                                                "     long        buffer_length;
"                                                                     int         rc;
                                                                "   } DialogTableEntry_t;
"
                                                                "   //a place to put free DTEntry cells
"                                                                   static DialogTableEntry_t DTFreeList={NULL,NULL,NULL,NULL,NULL,0,0,0};
                                                                "
"                                                                   //a place to put active dialogs
                                                                "   static DialogTableEntry_t DTActiveList={NULL,NULL,NULL,NULL,NULL,0,0,0};
"
                                                                "   //can use for tuning, keep high
"                                                                   #define RECEIVE_THREAD_PRIORITY
                                                                "   THREAD_PRIORITY_TIME_CRITICAL
"                                                                   #define MT_LOG_ERROR {\
                                                                "   char buf[160];\
"                                                                   sprintf(buf,"Problem at %d in %s\n",__LINE__,__FILE__);\
                                                                "   tp_system_log_text("TEthreads",buf);\
"                                                                   }
                                                                "
"                                                                   static void TEReceiveThread(void *ptr);
                                                                "   DialogTableEntry_t *GetDTE();
"                                                                   DialogTableEntry_t *FindOnList(long tp_user_dialog_id,DialogTableEntry_t *head);
                                                                    void RemoveFromList (DialogTableEntry_t *DTE, DialogTableEntry_t *head);
"</FORM><BODY>"                                                     void AddToList (DialogTableEntry_t *DTE, DialogTableEntry_t *head);
                                                        "</HTML>"; //prototypes for calls used in the application

char szTpccLogPath[256]; //path to html log file if logging turned on in registry.
char szErrorLogPath[256];       //path to error log file.

                                                                    int tp_mt_initialize(tp_application_info_t *application_info,
                                                                                         tp_funct_struct_t   function_array[],
                                                                                         long member_functions);
```

```
int tp_mt_client_signon (tp_dialogue_info_t *info,
                         tp_dialogue_user_t *client,
                         long inactivity_time,
                         tp_service_name_t *service,
                         tp_input_format_t *input_format,
                         long   message_length,
                         char   *message_text);

int tp_mt_client_send (tp_dialogue_info_t *info,
                       tp_service_name_t *service,
                       tp_input_format_t *input_format,
                       long    message_length,
                       char    *message_text);

int tp_mt_client_receive (tp_dialogue_info_t *info,
                          long         wait_time,
                          tp_output_format_t *output_format,
                          tp_service_name_t *service,
                          tp_location_t *location,
                          long         *buffer_length,
                          char         *message_buffer);

#endif


/* FUNCTION: BOOL APIENTRY DllMain(HANDLE hModule, DWORD
ul_reason_for_call,
 *          LPVOID lpReserved)
 *
 * PURPOSE: This function is the entry point for the DLL. This implementation
 *         is based on the fact that DLL_PROCESS_ATTACH is only called from
 *         the inet service once.
 *         Connections are sent to this function as thread attachments.
 *
 * ARGUMENTS:   HANDLE  hModule              module handle
 *            DWORD   ul_reason_for_call     reason for call
 *            LPVOID  lpReserved             reserved for future use
 *
 * RETURNS:     BOOL    FALSE                 errors occured in initialization
 *                  TRUE                  DLL successfully initialized
 *
 * COMMENTS: None
 *
 */
/*
BOOL APIENTRY DllMain(HANDLE hModule, DWORD ul_reason_for_call,
LPVOID lpReserved)
{
    int      i;
    static SECURITY_ATTRIBUTES       sa;
    static PSECURITY_DESCRIPTOR pSD;

#ifdef DEBUG_ENTRY
    DebugBreak( );
#endif


    switch( ul_reason_for_call )
    {
        case DLL_PROCESS_ATTACH:
        if ( ReadRegistrySettings() )
        {
          MessageBox(NULL, "Cannot Find TPCC Key in registry (run
install.exe).",
                                "Init", MB_OK | MB_ICONSTOP);
          return FALSE;
        }

        InitializeCriticalSection(&CriticalSection);
        InitializeCriticalSection(&ErrorLogCriticalSection);


        TermInit(EXTENSION_CONTROL_BLOCK* pECB);
        if ( !TermAllocate(EXTENSION_CONTROL_BLOCK* pECB) )
        {
          MessageBox(NULL, "Error Trm.Allocate().", "Init", MB_OK |
          MB_ICONSTOP);
          return FALSE;
        }

        for(i=Term->iNext; i<Term->iAvailable; i++)
          Term->pClientData[i].inUse = 0;
        Term->pClientData[0].inUse = 1;

        TLSIsTpInitedKey = TlsAlloc(); // check for failure later
```

```
        TLSNewOrderKey = TlsAlloc();
        TLSPaymentKey = TlsAlloc();
        TLSOrderStatusKey = TlsAlloc();
        TLSDeliveryKey = TlsAlloc();
        TLSStockLevelKey = TlsAlloc();
        // assumption:value inited to 0


        break;

        case DLL_THREAD_ATTACH:
        break;

        case DLL_THREAD_DETACH:
        break;

        case DLL_PROCESS_DETACH:


        if ( pSD )
                free( pSD );

        bTpccExit = TRUE;

        TermRestore(EXTENSION_CONTROL_BLOCK* pECB);

        DeleteCriticalSection(&CriticalSection);
        DeleteCriticalSection(&ErrorLogCriticalSection);

        TlsFree(TLSIsTpInitedKey);

        TlsFree(TLSNewOrderKey);
        TlsFree(TLSPaymentKey);
        TlsFree(TLSOrderStatusKey);
        TlsFree(TLSDeliveryKey);
        TlsFree(TLSStockLevelKey);


                break;
    }
    return TRUE;
}
*/

/* FUNCTION: BOOL WINAPI GetExtensionVersion(HSE_VERSION_INFO
*pVer)
 *
 * PURPOSE: This function is called by the inet service when the DLL is first
 *          loaded.
 * ARGUMENTS:   HSE_VERSION_INFO        *pVer   passed in structure in which
to
 *                                      place expected version number.
 *
 * RETURNS:     TRUE                    inet service expected return value.
 *
 * COMMENTS:    None
 *
 */
#ifdef WIN32
_declspec (dllexport)

BOOL WINAPI GetExtensionVersion(HSE_VERSION_INFO *pVer)
{
    pVer->dwExtensionVersion = MAKELONG(HSE_VERSION_MINOR,
HSE_VERSION_MAJOR);
    lstrcpyn(pVer->lpszExtensionDesc, "TPC-C Server.",
HSE_MAX_EXT_DLL_NAME_LEN);
    return TRUE;
}
#endif

/* FUNCTION: DWORD WINAPI
HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB)
 *
 * PURPOSE: This function is the main entry point for the TPCC DLL. The
 *          internet service calls this function passing in the http string.
 *
 * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB    structure pointer
to passed in
 *                                      internet service information.
 *
 * RETURNS:     DWORD   HSE_STATUS_SUCCESS    connection can be dropped
if error
 *                  HSE_STATUS_SUCCESS_AND_KEEP_CONN keep connect valid
```

```
comment sent
 *
 * COMMENTS:   None
 *
 */
/*
_declspec (dllexport)
DWORD WINAPI HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB)

                    ////////////////////////////
*/
DWORD           HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB)
{
    int iCmd, FormId, TermId, iSyncId;
    FILE  *fp;

    static BOOL bReadRegistry = FALSE;


    request_rec *r = pECB->r;
//    SET_TERM_AND_CDATA

    if ( iMaxConnections == -1 )
    {
        ErrorMessage(pECB, ERR_CAN_NOT_SET_MAX_CONNECTIONS,
                    ERR_TYPE_WEBDLL, NULL, -1, -1);
        return HSE_STATUS_SUCCESS;
    }


    //if registry setting is for html logging then show http string passed in.
        if ( bLog )
        {

#ifdef WIN32
                    SYSTEMTIME      systemTime;

                    fp = fopen(szTpccLogPath, "ab");

                    GetLocalTime(&systemTime);

                    fprintf(fp, "* QUERY
* %2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d\r\n\r\n%s\r\n\r\n",
                                systemTime.tm_year, systemTime.tm_mon,
systemTime.tm_mday,
                                systemTime.tm_hour, systemTime.tm_min,
systemTime.tm_sec,
                                pECB->lpszQueryString);
                    fclose(fp);
#else
                    struct tm* systemTime;
                    time_t    now;

                    fp = fopen(szTpccLogPath, "ab");
                    time(&now);
                    systemTime = localtime(&now );

                    fprintf(fp, "* QUERY
* %2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d\r\n\r\n%s\r\n\r\n",
                                systemTime->tm_year, systemTime->tm_mon,
systemTime->tm_mday,
                                systemTime->tm_hour, systemTime->tm_min,
systemTime->tm_sec,
                                pECB->lpszQueryString);
                    fclose(fp);
#endif
            }

      //process http query
    if ( !ProcessQueryString(pECB, &iCmd, &FormId, &TermId, &iSyncId) )
    {
                        if ( TermId < 0 ){


                                WriteMessageToEventLog(TEXT("ProcessQueryString is
failed. Invalid TermID"));

                            ErrorMessage(pECB, ERR_INVALID_TERMID,
ERR_TYPE_WEBDLL,
                                    NULL, TermId, iSyncId);
                        }
            else

                    WriteMessageToEventLog(TEXT("Cannot get TermID and
SyncID"));
```

```
                ErrorMessage(pECB, ERR_COMMAND_UNDEFINED,
                        ERR_TYPE_WEBDLL, NULL, TermId, iSyncId);
            return 0;//HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }
     if ( TermId != 0 )
     {

                if ( !IsValidTermId(TermId) )
                {

#ifdef FURLOG
                                                        {       FILE
*fur;
                                                        fur =
fopen("C:\\temp\\fur.txt", "ab");
                                                        fprintf(fur,
"non0 Invalid termid = %d \n", TermId);
                                                        fclose(fur);
                                                                }
#endif // FURLOG


                WriteMessageToEventLog(TEXT("Invalid TermID"));

                ErrorMessage(pECB, ERR_INVALID_TERMID,
ERR_TYPE_WEBDLL,
                NULL, TermId, iSyncId);
                return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
                }

        //must have a valid syncid here since termid is valid
                if ( iSyncId < 1 || iSyncId != Term->pClientData[TermId].iSyncId )
                {


                        WriteMessageToEventLog(TEXT("Invalid SyncID"));

                        ErrorMessage(pECB,
ERR_INVALID_SYNC_CONNECTION,
                                ERR_TYPE_WEBDLL, NULL, TermId,
iSyncId);
                        return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
                }
    }

                //set use time
                Term->pClientData[TermId].iTickCount = GetTickCount();

                //go execute http: command

                (*DoCmd[iCmd])(pECB, FormId, TermId, iSyncId);
                //finish up and keep connection
                return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
}


/* FUNCTION: static BOOL IsValidTermId(int TermId)
 *
 * PURPOSE:  This function checks to see of the passed in terminal id is valid.
 *
 * ARGUMENTS:   int     TermId          client terminal id
 *
 * RETURNS:     BOOL    FALSE           Terminal ID Invalid
 *                      TRUE            Terminal ID valid
 *
 * COMMENTS:    None
 *
 */

static BOOL IsValidTermId(int TermId)
{
    return (BOOL) ( TermId > 0 && TermId <= Term->iAvailable && Term-
>pClientData[TermId].inUse );
}


/* FUNCTION: BOOL ProcessQueryString(EXTENSION_CONTROL_BLOCK
*pECB, int *pCmd,
 *                              int *pFormId, int *pTermId, int *pSyncId)
 *
 * PURPOSE: This function extracts the relevent information out of the http
 *          command passed in from the browser.
 *
 * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB    structure pointer
```

```
to passed in
 *                                  internet service information.
 *          int             *pCmd    returned command id
 *          int             *pFormId returned active form client
                                                        browser is on
 *          int             *pTermId returned client terminal id
 *
 * RETURNS:    BOOL    FALSE   success
 *                 TRUE    command passed in is invalid
 *
 * COMMENTS: If this is the initial connection i.e. client is at welcome screen
 *         then there will not be a terminal id or current form id.
 *         If this is the case then the pTermid and pFormid return values are
 *          undefined.
 */

BOOL ProcessQueryString(EXTENSION_CONTROL_BLOCK *pECB, int *pCmd,
int *pFormId,
                        int *pTermId, int *pSyncId)
{

    char *ptr;
    char szBuffer[25];
    char szTmp[25];
    char *dest = szBuffer;
    int i;

//          WR_LOG("enter pqs");
//          WR_LOG(pECB->lpszQueryString);
    if ( (ptr = strstr(pECB->lpszQueryString, "FORMID=")) )
          *pFormId = *(ptr+7) & 0x0F;

    if ( (ptr = strstr(pECB->lpszQueryString, "TERMID=")) )
    {
                *pTermId = atoi((ptr+7));

                if ( *pTermId == 0 )          //terminal id 0 used internally
                        *pTermId = -1;
                if ( *pTermId == -2 )         //login screen
                        *pTermId = 0;
    }
    else
                *pTermId = 0;

    if ( (ptr = strstr(pECB->lpszQueryString, "SYNCID=")) )
                *pSyncId = atoi((ptr+7));
    else
                *pSyncId = 0;

    if ( !(ptr = strstr(pECB->lpszQueryString, "CMD=")) )
    {
                ptr = szBuffer;

                if ( !stricmp(szBuffer, "Default") )
                        strcpy(szBuffer, "CMD=Begin");

                switch( *pFormId )
                {
                        case WELCOME_FORM:
                                strcpy(szBuffer, "CMD=Submit");
                                break;
                        case MAIN_MENU_FORM:
                                strcpy(szBuffer, "CMD=NewOrder");
                                break;
                        case NEW_ORDER_FORM:
                        case PAYMENT_FORM:
                        case DELIVERY_FORM:
                        case ORDER_STATUS_FORM:
                        case STOCK_LEVEL_FORM:
                                if ( !(*pTermId) )
                                        return FALSE;
                                if ( GetKeyValue(pECB->lpszQueryString,"PI*",
szTmp, sizeof(szTmp)) )
                                        strcpy(szBuffer, "CMD=Process");
                                else
                                {
                                        strcpy(szBuffer, "CMD=");
                                        strcat(szBuffer, szCmds[*pFormId -
NEW_ORDER_FORM]);
                                }
                                break;
                        default:
                                return FALSE;
```

```
                }
        }
        ptr += 4;

        while( *ptr && *ptr != '&' )
                *dest++ = *ptr++;
        *dest = 0;
//      WR_LOG(szBuffer);

        for(i=0; szCmds[i][0]; i++)
        {
                if ( !strcmp(szCmds[i], szBuffer) )
                {
                        *pCmd = i;

                return TRUE;
                }
        }
        return FALSE;
}


/* FUNCTION: void NewOrderForm(EXTENSION_CONTROL_BLOCK *pECB,
int iFormId,
 *                      int iTermId, intiSyncId)
 *
 * PURPOSE:  This function wraps the functionality needed for the TPC-C New
 *          Order Form.
 *
 * ARGUMENTS:   int     iFormId         unused
 *          int     iTermId         id of calling browser, i.e. TERMID=from http
 *                                                  command line
 *          EXTENSION_CONTROL_BLOCK *pECB           structure pointer to
passed in
 *                                  internet service information
 *
 * RETURNS:     None
 *
 * COMMENTS:    None
 *
 */

void NewOrderForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{
    WriteZString(pECB, MakeNewOrderForm(iTermId, iSyncId, TRUE,
FALSE,""));
    UNUSEDPARAM(iFormId);
    return;
}


/* FUNCTION: void PaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int
 *                      iTermId, int iSyncId)
 *
 * PURPOSE: This function wraps the functionality for the TPC-C Payment Form.
 *
 * ARGUMENTS:   int     iFormId         unused
 *          int     iTermId         id of calling browser, i.e. TERMID=
 *                                  from http command line
 *          int     iSyncId         sync id of calling browser
 *
 *          EXTENSION_CONTROL_BLOCK *pECB   structure pointer to passed in
 *                                  internet service information.
 * RETURNS:     None
 *
 * COMMENTS:    None
 *
 */

void PaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{
    WriteZString(pECB, MakePaymentForm(iTermId, iSyncId, TRUE) );
    UNUSEDPARAM(iFormId);
    return;

}


/* FUNCTION: void DeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int
 *                      iTermId, int iSyncId)
 *
 * PURPOSE: This function wraps the functionality for the TPC-C Delivery Form.
```

```
 *
 * ARGUMENTS:   int     iFormId        unused
 *              int     iTermId        id of calling browser, i.e. TERMID=
 *                                     from http command line
 *              int     iSyncId        sync id of calling browser
 *              EXTENSION_CONTROL_BLOCK *pECB   structure pointer to passed in
 *                                     internet service information.
 * RETURNS:     None
 *
 * COMMENTS:    None
 *
 */

void DeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{

    WriteZString(pECB, MakeDeliveryForm(iTermId, iSyncId, TRUE, "") );
    UNUSEDPARAM(iFormId);

    return;

}


/* FUNCTION: void OrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB,
int iFormId,
 *                          int iTermId, int iSyncId)
 *
 * PURPOSE: This function wraps the function of the TPC-C Order Status Form.
 *
 * ARGUMENTS:   int     iFormId        unused
 *              int     iTermId        id of calling browser, i.e. TERMID=
 *                                     from http command line
 *              int     iSyncId        sync id of calling browser
 *              EXTENSION_CONTROL_BLOCK *pECB   structure pointer to passed in
 *                                     internet service information.
 * RETURNS:     None
 *
 * COMMENTS:    None
 *
 */
```

```
void OrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{

    WriteZString(pECB, MakeOrderStatusForm(iTermId, iSyncId, TRUE) );
    UNUSEDPARAM(iFormId);

    return;

}


/* FUNCTION: void StockLevelForm(EXTENSION_CONTROL_BLOCK *pECB,
int iFormId,
c *                          int iTermId, int iSyncId)
 *
 * PURPOSE:  This function wraps the functions of the TPC-C Stock Level Form.
 *
 * ARGUMENTS:   int     iFormId        unused
 *              int     iTermId        id of calling browser, i.e. TERMID=
 *                                     from http command line
 *              int     iSyncId        sync id of calling browser
 *              EXTENSION_CONTROL_BLOCK *pECB   structure pointer to passed in
 *                                     internet service information.
 * RETURNS:     None
 *
 * COMMENTS:    None
 *
 */

void StockLevelForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{

    WriteZString(pECB, MakeStockLevelForm(iTermId, iSyncId, TRUE) );

    return;
}


/* FUNCTION: void Exitcmd(EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId, int iSyncId)
 *
```

```
 * PURPOSE: This function removes a terminal id from use, the allocated
 *          structure however remains valid so the next request for a new client
 *          will not require a new memory allocation.
 *
 * ARGUMENTS:   int     iFormId         unused
 *              int     iTermId         id of calling browser, i.e. TERMID=
 *                                      from http command line
 *              int     iSyncId         sync id of calling browser
 *              EXTENSION_CONTROL_BLOCK *pECB   structure pointer to passed in
 *                                      internet service information.
 * RETURNS:     None
 *
 * COMMENTS:    None
 *
 */

void Exitcmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId,
int iSyncId)
{
    TermDelete(pECB, iTermId);
    WriteZString(pECB, MakeWelcomeForm() );
    UNUSEDPARAM(iFormId);
    UNUSEDPARAM(iSyncId);

    return;
}


/* FUNCTION: void SubmitCmd(EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int
 *                      iTermId, int iSyncId)
 *
 * PURPOSE: This function allocates a new terminal id in the Term struct array.
 *
 * ARGUMENTS:   int     iFormId         unused
 *              int     iTermId         id of calling browser, i.e. TERMID=
 *                                      from http command line
 *              int     iSyncId         sync id of calling browser
 *      EXTENSION_CONTROL_BLOCK         *pECB   structure pointer to passed in
 *                                      internet service information.
 * RETURNS:     None
 *
 * COMMENTS:  A terminal id can be allocated but still be invalid if the
 *          requested warehouse number is outside the range specified in the
 *          registry. This then will force the client id to be invalid and an
 *          error message sent to the users browser.
 */

void SubmitCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{
    int iCurrent;
    if ( (iCurrent = TermAdd(pECB, pECB->lpszQueryString)) < 0 )
    {
        ErrorMessage(pECB, ERR_CANNOT_INIT_TERMINAL,
ERR_TYPE_WEBDLL,NULL, iCurrent, iSyncId);
        return;
    }

    if ( Term->pClientData[iCurrent].w_id > iMaxWareHouses || Term-
>pClientData[iCurrent].w_id < 1 )
    {
        ErrorMessage(pECB, ERR_W_ID_INVALID, ERR_TYPE_WEBDLL,
NULL, iCurrent, iSyncId);
        TermDelete(pECB, iCurrent);
        return;
    }
    if ( Term->pClientData[iCurrent].d_id < 1 || Term->pClientData[iCurrent].d_id >
10 )
    {
        ErrorMessage(pECB, ERR_D_ID_INVALID, ERR_TYPE_WEBDLL,
NULL, iCurrent, iSyncId);
        TermDelete(pECB, iCurrent);
        return;
    }
    WriteZString(pECB, MakeMainMenuForm(iCurrent, Term-
>pClientData[iCurrent].iSyncId) );
    return;
}


/* FUNCTION: void BeginCmd(EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId, int iSyncId)
 *
 * PURPOSE:  This function is the first command executed. It is executed with
 *          the command:   CMD=Begin?Server=xxx from the http command line.
```

```
 *
 * ARGUMENTS:   int     iFormId         unused
 *              int     iTermId         id of calling browser, i.e. TERMID=
 *                                      from http command line
 *              int     iSyncId         sync id of calling browser
 *      EXTENSION_CONTROL_BLOCK         *pECB   structure pointer to passed in
 *                                      internet service information.
 *
 * RETURNS:     None
 *
 * COMMENTS:    SQL server must be specified, however the user and password
 *              parameters are optional.  The complete command line is
 *              CMD=Begin&Server=server&User=sa&Psw=&. The & are used
 *              to separate parameters which is internet browser standard.
 */

void BeginCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{
    LPSTR pQueryString;
    pQueryString = pECB->lpszQueryString;

    WriteZString(pECB, MakeWelcomeForm() );

    UNUSEDPARAM(iFormId);
    return;
}


/* FUNCTION: void ProcessCmd(EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int
 *                      iTermId, int iSyncId)
 *
 * PURPOSE:     This function process the passed in http command
 *
 * ARGUMENTS:   int     iFormId         unused
 *              int     iTermId         id of calling browser, i.e. TERMID=
 *                                      from http command line
 *              int     iSyncId         sync id of calling browser
 *      EXTENSION_CONTROL_BLOCK         *pECB   structure pointer to passed in
 *                                      internet service information.
 * RETURNS:     None
 *
 * COMMENTS:    None
 *
 */
void ProcessCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{
    switch( iFormId )
    {
        case WELCOME_FORM:
                return;
        case MAIN_MENU_FORM:
                return;
          }

    switch( iFormId )
    {
    case NEW_ORDER_FORM:
                ProcessNewOrderForm(pECB, iTermId, iSyncId);
                break;
    case PAYMENT_FORM:
                ProcessPaymentForm(pECB, iTermId, iSyncId);
                break;
    case DELIVERY_FORM:
                ProcessDeliveryForm(pECB, iTermId, iSyncId);
                break;
    case ORDER_STATUS_FORM:
                ProcessOrderStatusForm(pECB, iTermId, iSyncId);
                break;
    case STOCK_LEVEL_FORM:
                ProcessStockLevelForm(pECB, iTermId, iSyncId);
                break;
    }
}


/* FUNCTION: void ClearCmd(EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int
 *                      iTermId, int iSyncId)
 *
 * PURPOSE:     This function frees all currently logged in terminal ids.
 *
 * ARGUMENTS:   int     iFormId         unused
 *              int     iTermId         id of calling browser, i.e. TERMID=
```

```
 *                              from http command line
 *          int     iSyncId         sync id of calling browser
 *     EXTENSION_CONTROL_BLOCK         *pECB   structure pointer to passed in
 *                                  internet service information.
 * RETURNS:     None
 *
 * COMMENTS:    Use this function with caution, it may cause unpredictable
 *          results if existing browsers attempt to use the web client
 *          without beginning at the login screen for each client.
 */

void ClearCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId,
int iSyncId)
{
    int i;
    EnterCriticalSection(&CriticalSection);
    for(i=0; i<Term->iAvailable; i++)
    {
        if ( Term->pClientData[i].inUse )
            TermDelete(pECB, i);
    }
    Term->iNext             = 0;
    Term->iAvailable        = 0;
    Term->iMasterSyncId     = 1;

    if ( Term->pClientData )
        free(Term->pClientData);

    Term->pClientData       = NULL;
    Term->bInit             = FALSE;

    TermInit(pECB->r->server);

    if ( !TermAllocate(pECB->r->server) )
    {
        ErrorMessage(pECB, ERR_MAX_CONNECT_PARAM,
ERR_TYPE_WEBDLL,NULL, iTermId, iSyncId);
        return;
    }

    for(i=Term->iNext; i<Term->iAvailable; i++)
        Term->pClientData[i].inUse = 0;
    Term->pClientData[0].inUse = 1;
    LeaveCriticalSection(&CriticalSection);

    WriteZString(pECB, MakeWelcomeForm() );
    return;
}


/* FUNCTION: void MenuCmd(EXTENSION_CONTROL_BLOCK *pECB, int
iFormId,
 *                  int iTermId, int iSyncId)
 *
 * PURPOSE: This function causes an exit to the main menu
 *
 * ARGUMENTS:   int     iFormId         unused
 *          int     iTermId         id of calling browser, i.e. TERMID=
 *                              from http command line
 *          int     iSyncId         sync id of calling browser
 *     EXTENSION_CONTROL_BLOCK         *pECB   structure pointer to passed in
 *                                  internet service information.
 * RETURNS:     None
 *
 * COMMENTS:    None
 *
 */

void MenuCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{
    WriteZString(pECB, MakeMainMenuForm(iTermId, iSyncId) );
    return;
}


/* FUNCTION: void WriteZString(EXTENSION_CONTROL_BLOCK *pECB, char
*szStr)
 *
 * PURPOSE:  This function is the low level output function. It writes a string
 * of text back to the client browser.
 *
 * ARGUMENTS: EXTENSION_CONTROL_BLOCK   *pECB   passed in structure
pointer from
 *                                  inetsrv.
```

```
 *          char                *szStr  string to display in the client
 *                                  browser.
 *
 * RETURNS:     None
 *
 * COMMENTS:    This function assumes that the string to written to the client
 *          browser has been formatted in an HTML manner.
 */

static void WriteZString(EXTENSION_CONTROL_BLOCK *pECB, char *szStr)

{
    FILE     *fp;
    int      lpbSize;
    int      iSize;
    char     szHeader[128];
    char     szHeader1[128];

    lpbSize = strlen(szStr)+1;

    if ( bLog )
    {
#ifdef WIN32
                SYSTEMTIME      systemTime;

                fp = fopen(szTpccLogPath, "ab");

                GetLocalTime(&systemTime);

                fprintf(fp, "* HTML PAGE
* %2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d\r\n\r\n%s\r\n\r\n",
                        systemTime.wYear, systemTime.wMonth,
systemTime.wDay,
                        systemTime.wHour, systemTime.wMinute,
systemTime.wSecond,
                        szStr);
#else
                struct tm* systemTime;
                time_t    now;
                fp = fopen(szTpccLogPath, "ab");
                time(&now);
                systemTime = localtime(&now );
                fprintf(fp, "* QUERY
* %2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d\r\n\r\n%s\r\n\r\n%s\r\n",
                        systemTime->tm_year, systemTime->tm_mon,
systemTime->tm_mday,
                        systemTime->tm_hour, systemTime->tm_min,
systemTime->tm_sec,
                        pECB->lpszQueryString,szStr);
#endif
                fclose(fp);
    }

#ifdef PERFORMIX

        (*pECB->ServerSupportFunction)(pECB->ConnID,
HSE_REQ_DONE_WITH_SESSION, NULL, 0, 0);
        (*pECB->WriteClient)(pECB->ConnID, szStr, &lpbSize, 0);

#else
#ifdef WIN32
        iSize = sprintf(szHeader, "200 Ok");

        sprintf(szHeader1, "Connection: keep-alive\r\nContent-type:
text/html\r\nContent-Length: %d\r\n\r\n", lpbSize);//change 2003/7/11
        (*pECB->ServerSupportFunction)(pECB->ConnID,
HSE_REQ_SEND_RESPONSE_HEADER, szHeader, &iSize,
(LPDWORD)szHeader1);
        (*pECB->WriteClient)(pECB->ConnID, szStr, &lpbSize, 0);
#else
        //ap_set_keepalive(pECB->r);
        //ap_set_content_length(pECB->r,lpbSize);
        ap_set_content_type(pECB->r,"text/html");
      ap_rputs(szStr, pECB->r);
#endif
#endif

    return;
}


/* FUNCTION: void h_printf(EXTENSION_CONTROL_BLOCK *pECB, char
*format, ...)
 *
 * PURPOSE:  This function forms a high level printf for an HTML browser
```

```
 *                                                                
 * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB   passed in structure
pointer from
 *                                  inetsrv.
 *          char                *format printf style format string
 *          ...                 other arguments as required by printf
 *                              style format string.
 *
 * RETURNS:     None
 *
 * COMMENTS: This function is mainly used for developmental support.
 */

static void h_printf(EXTENSION_CONTROL_BLOCK *pECB, char *format, ...)
{
    int lpbSize;
    char szBuff[512];
    char szTmp[512];

    va_list marker;
    va_start( marker, format );
    vsprintf(szTmp, format, marker);
    va_end( marker );

    lpbSize = wsprintf(szBuff, "<HTML>%s</HTML>", szTmp) + 1;
#ifdef WIN32
    (*pECB->WriteClient)(pECB->ConnID, szBuff, &lpbSize, 0);
#else
      ap_rputs(szBuff, pECB->r);
#endif

    return;
}



/* FUNCTION: void ErrorMessage(EXTENSION_CONTROL_BLOCK *pECB, int
iError, int
 *                          iErrorType, char *szMsg)
 *
 * PURPOSE:     This function displays an error message in the client browser.
 *
 * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB   passed in structure
pointer from
 *                                  inetsrv.
 *          int             iError  id of error message
 *          int         iErrorType      error type, ERR_TYPE_SQL,
 *                              ERR_TYPE_OCI, or
 *                              ERR_TYPE_WEBDLL
 *          int         iTermId         terminal id from browser
 *          int         iSyncid         sync id from browser
 *          char          *szMsg          optional error message string
 *                          used with ERR_TYPE_SQL and ERR_TYPE_OCI
 *
 * RETURNS:     None
 *
 * COMMENTS:    If the error type is ERR_TYPE_WEBDLL the szmsg parameter
may be
 *          NULL because it is ignored. If the error type is ERR_TYPE_SQL or
 *          ERR_TYPE_OCI then the szMsg parameter contains the text of the
 *          error message, so the szMsg parameter cannot be NULL.
 *
 */

void ErrorMessage(EXTENSION_CONTROL_BLOCK *pECB, int iError, int
iErrorType, char *szMsg, int iTermId, int iSyncId)
{
    int i;
    static SERRORMSG errorMsgs[] =
    {
        {   ERR_SUCCESS,   "Success, no error." },
        {   ERR_COMMAND_UNDEFINED, "Command undefined." },
        {   ERR_NOT_IMPLEMENTED_YET, "Not Implemented Yet." },
        {   ERR_CANNOT_INIT_TERMINAL, "Cannot initialize client
connection." },
        {   ERR_OUT_OF_MEMORY, "insufficient memory." },
        {   ERR_NEW_ORDER_NOT_PROCESSED, "Cannot process new
Order form." },
        {   ERR_PAYMENT_NOT_PROCESSED, "Cannot process payment
form." },
        {   ERR_NO_SERVER_SPECIFIED, "No Server name specified." },
        {   ERR_ORDER_STATUS_NOT_PROCESSED, "Cannot process order
status form." },
        {   ERR_W_ID_INVALID, "Invalid Warehouse ID." },
        {   ERR_CAN_NOT_SET_MAX_CONNECTIONS, "Insufficient
memory to allocate # connections." },
        {   ERR_NOSUCH_CUSTOMER, "No such customer." },
```

```
        {       ERR_D_ID_INVALID, "Invalid District ID Must be 1 to 10." },
        {       ERR_MAX_CONNECT_PARAM, "Max client connections exceeded,
run install to increase." },
        {       ERR_INVALID_SYNC_CONNECTION, "Invalid Terminal Sync
ID." },
        {       ERR_INVALID_TERMID, "Invalid Terminal ID." },
        {       ERR_PAYMENT_INVALID_CUSTOMER, "Payment Form, No such
Customer." },
        {       ERR_SQL_OPEN_CONNECTION, "SQLOpenConnection API
Failed." },
        {       ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY, "Stock Level
missing Threshold key \"TT*\"." },
        {       ERR_STOCKLEVEL_THRESHOLD_INVALID, "Stock Level
Threshold invalid data type range = 1 - 99." },
        {       ERR_STOCKLEVEL_THRESHOLD_RANGE, "Stock Level
Threshold out of range, range must be 1 - 99." },
        {       ERR_STOCKLEVEL_NOT_PROCESSED,  "Stock Level not
processed." },
        {       ERR_NEWORDER_FORM_MISSING_DID,  "New Order missing
District key \"DID*\"." },
        {       ERR_NEWORDER_DISTRICT_INVALID, "New Order District ID
Invalid range 1 - 10." },
        {       ERR_NEWORDER_DISTRICT_RANGE, "New Order District ID out
of Range. Range = 1 - 10." },
        {       ERR_NEWORDER_CUSTOMER_KEY,"New Order missing
Customer key \"CID*\"." },
        {       ERR_NEWORDER_CUSTOMER_INVALID, "New Order customer
id invalid data type, range = 1 to 3000." },
        {       ERR_NEWORDER_CUSTOMER_RANGE, "New Order customer id
out of range, range = 1 to 3000." },
        {       ERR_NEWORDER_MISSING_IID_KEY, "New Order missing Item
Id key \"IID*\"." },
        {       ERR_NEWORDER_ITEM_BLANK_LINES, "New Order blank order
lines all orders must be continuous." },
        {       ERR_NEWORDER_ITEMID_INVALID, "New Order Item Id is
wrong data type, must be numeric." },
        {       ERR_NEWORDER_MISSING_SUPPW_KEY, "New Order missing
Supp_W key \"SP##*\"." },
        {       ERR_NEWORDER_SUPPW_INVALID, "New Order Supp_W
invalid data type must be numeric." },
        {       ERR_NEWORDER_MISSING_QTY_KEY, "New Order Missing Qty
key \"Qty##*\"." },
        {       ERR_NEWORDER_QTY_INVALID, "New Order Qty invalid must
be numeric range 1 - 99." },
        {       ERR_NEWORDER_SUPPW_RANGE, "New Order Supp_W value
out of range - 1 to MaxWarehouses." },
        {       ERR_NEWORDER_ITEMID_RANGE, "New Order Item Id is out of
range. Range = 1 to 999999." },
        {       ERR_NEWORDER_QTY_RANGE, "New Order Qty is out of range.
Range = 1 to 99." },
        {       ERR_PAYMENT_DISTRICT_INVALID, "Payment District ID is
invalid must be 1 - 10." },
        {       ERR_NEWORDER_SUPPW_WITHOUT_ITEMID, "New Order
Supp_W field entered without a Item_Id." },
        {       ERR_NEWORDER_QTY_WITHOUT_ITEMID, "New Order Qty
entered without a corresponding Item_Id." },
        {       ERR_NEWORDER_NOITEMS_ENTERED, "New Order Blank Items
between items, items must be continuous." },
        {       ERR_PAYMENT_MISSING_DID_KEY,"Payment missing district
Key \"DID*\"." },
        {       ERR_PAYMENT_DISTRICT_RANGE,"Payment District Out of
range, range = 1 - 10." },
        {       ERR_PAYMENT_MISSING_CID_KEY,"Payment missing Customer
Key \"CID*\"." },
        {       ERR_PAYMENT_CUSTOMER_INVALID,"Payment Customer data
type invalid, must be numeric." },
        {       ERR_PAYMENT_MISSING_CLT,"Payment missing Customer Last
Name Key \"CLT*\"." },
        {       ERR_PAYMENT_LAST_NAME_TO_LONG,"Payment Customer
last name longer than 16 characters." },
        {       ERR_PAYMENT_CUSTOMER_RANGE,"Payment Customer ID out
of range, must be 1 to 3000." },
        {       ERR_PAYMENT_CID_AND_CLT, "Payment Customer ID and Last
Name entered must be one or other." },
        {       ERR_PAYMENT_MISSING_CDI_KEY,"Payment missing Customer
district key \"CDI*\"." },
        {       ERR_PAYMENT_CDI_INVALID,"Payment Customer district invalid
must be numeric." },
        {       ERR_PAYMENT_CDI_RANGE,"Payment Customer district out of
range must be 1 - 10." },
        {       ERR_PAYMENT_MISSING_CWI_KEY,"Payment missing Customer
Warehouse key \"CWI*\"." },
        {       ERR_PAYMENT_CWI_INVALID,"Payment Customer Warehouse
invalid must be numeric." },
        {       ERR_PAYMENT_CWI_RANGE,"Payment Customer Warehouse out
of range, 1 to Max Warehouses." },
```

```
        {       ERR_PAYMENT_MISSING_HAM_KEY,"Payment missing Amount
key \"HAM*\"." },
        {       ERR_PAYMENT_HAM_INVALID,"Payment Amount invalid data
type must be numeric." },
        {       ERR_PAYMENT_HAM_RANGE,"Payment Amount out of range, 0 -
9999.99." },
        {       ERR_ORDERSTATUS_MISSING_DID_KEY,"Order Status missing
District key \"DID*\"." },
        {       ERR_ORDERSTATUS_DID_INVALID,"Order Status District invalid,
value must be numeric 1 - 10." },
        {       ERR_ORDERSTATUS_DID_RANGE,"Order Status District out of
range must be 1 - 10." },
        {       ERR_ORDERSTATUS_MISSING_CID_KEY,"Order Status missing
Customer key \"CID*\"." },
        {       ERR_ORDERSTATUS_MISSING_CLT_KEY, "Order Status missing
Customer Last Name key \"CLT*\"." },
        {       ERR_ORDERSTATUS_CLT_RANGE, "Order Status Customer last
name longer than 16 characters." },
        {       ERR_ORDERSTATUS_CID_INVALID, "Order Status Customer ID
invalid, range must be numeric 1 - 3000." },
        {       ERR_ORDERSTATUS_CID_RANGE, "Order Status Customer ID out
of range must be 1 - 3000." },
        {       ERR_ORDERSTATUS_CID_AND_CLT, "Order Status Customer ID
and LastName entered must be only one." },
        {       ERR_DELIVERY_MISSING_OCD_KEY, "Delivery missing Carrier
ID key \"OCD*\"." },
        {       ERR_DELIVERY_CARRIER_INVALID, "Delivery Carrier ID
invalid must be numeric 1 - 10." },
        {       ERR_DELIVERY_CARRIER_ID_RANGE, "Delivery Carrier ID out
of range must be 1 - 10." },
        {       ERR_PAYMENT_MISSING_CLT_KEY, "Payment missing Customer
Last Name key \"CLT*\"." },
        {       ERR_TPINIT_BAD,                 "Tuxedo tpinit
Failed."                                        },
        {       ERR_TPALLOC_BAD,                "Tuxedo tpalloc
Failed."                                        },
        {       ERR_TPCALL_BAD,                 "Tuxedo tpcall
Failed."                                        },
        {       TOPEND_SEND_ERROR,      "TOPEND client send error."},
        {       TOPEND_RECEIVE_ERROR,   "TOPEND client receive error."},
        {       0,              "" }
    };
```

```
    static char szNoMsg[] = "";
    char        *szForm;

    if ( !szMsg  )
        szMsg = szNoMsg;

    if ( iTermId > 0 && IsValidTermId(iTermId) )
        szForm = Term->pClientData[iTermId].szBuffer; //if termid valid use
common terminal   static buffer.
    else
        szForm = Term->pClientData[0].szBuffer;  //else term id invalid so use
common terminal static buffer.


    switch(iErrorType)
    {
        case ERR_TYPE_WEBDLL:
          for(i=0; errorMsgs[i].szMsg[0]; i++)
          {
                if ( iError == errorMsgs[i].iError )
                break;
          }

          if ( !errorMsgs[i].szMsg[0] )
                i = 1;

          strcpy(szForm,  "<HTML><HEAD><TITLE>Welcome To TPC-
C</TITLE></HEAD><BODY><FORM ACTION=\"tpcc.dll\"
METHOD=\"GET\">");
          wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"STATUSID\" VALUE=\"%d\">", iError);
          wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"ERROR\" VALUE=\"%d\">", iErrVal);
          wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
          wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
          wsprintf(szForm+strlen(szForm), "Error: TPCCWEB(%d): %s",
iError,errorMsgs[i].szMsg);
          strcat(szForm, "</FORM><BODY></HTML>");
          WriteZString(pECB, szForm);
          break;
```

```
        case ERR_TYPE_SQL:
          strcpy(szForm,"<HTML><HEAD><TITLE>Welcome To TPC-
C</TITLE></HEAD><BODY><FORM ACTION=\"tpcc.dll\"
METHOD=\"GET\">");
          wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"STATUSID\" VALUE=\"%d\">", iError);
          wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"ERROR\" VALUE=\"%d\">", iErrVal);
          wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
          wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
          wsprintf(szForm+strlen(szForm), "Error: Oracle(%d): %s", iError, szMsg);
          strcat(szForm, "</FORM><BODY></HTML>");
          WriteZString(pECB, szForm);
          break;

        case ERR_TYPE_OCI:
          strcpy(szForm,"<HTML><HEAD><TITLE>Welcome To TPC-
C</TITLE></HEAD><BODY><FORM ACTION=\"tpcc.dll\"
METHOD=\"GET\">");
          wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"STATUSID\" VALUE=\"%d\">", iError);
          wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"ERROR\" VALUE=\"%d\">", iErrVal);
          wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
          wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
          wsprintf(szForm+strlen(szForm), "Error: OCI(%d): %s", iError, szMsg);
          strcat(szForm, "</FORM><BODY></HTML>");
          WriteZString(pECB, szForm);
          break;

        case ERR_TYPE_TUXEDO:
                    strcpy(szForm,  "<HTML><HEAD><TITLE>Welcome
To TPC-C</TITLE></HEAD><BODY><FORM ACTION=\"tpcc.dll\"
METHOD=\"GET\">");
                    wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"%d\">", iError);
                    wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\" NAME=\"ERROR\" VALUE=\"%d\">", iErrVal);
                    wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
                    wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
                    wsprintf(szForm+strlen(szForm), "Error: Tuxedo: %d %s",
iError, szMsg);
                    strcat(szForm, "</FORM><BODY></HTML>");
                    WriteZString(pECB, szForm);
                    break;
    }
    return;
}


/* FUNCTION: BOOL GetKeyValue(char *pQueryString, char *pKey, char *pValue,
 *                    int iMax, char **pBeyondKey)
 *
 * PURPOSE: This function parses http formatted string for specific key values.
 *
 * ARGUMENTS:   char    *pQueryString   http string from client browser
 *          char    *pKey           key value to look for
 *          char    *pValue  character array into which to place key's value
 *          int     iMax            maximum length of key value array.
 *          char    **pBeyondKey    Pointer to location beyond key value
 *
 * RETURNS:     BOOL    FALSE           key value not found
 *                  TRUE            key value found
 *
 * COMMENTS:    http keys are formatted either KEY=value& or KEY=value\0.
This
 * DLL formats TPC-C input fields in such a manner that the keys can be
 * extracted in the above manner.
 */

static BOOL GetKeyValue(char *pQueryString, char *pKey, char *pValue, int iMax)

{
    char *ptr;

    if ( !(ptr=strstr(pQueryString, pKey)) )
        return FALSE;
    if ( !(ptr=strchr(ptr, '=')) )
        return FALSE;
    ptr++;
```

```
    iMax--;
    while( *ptr && *ptr != '&' && iMax)
    {
        *pValue++ = *ptr++;
        iMax--;
    }
    *pValue = 0;
    return TRUE;
}


/* FUNCTION: void TermInit(void)
 *
 * PURPOSE: This function initializes the client ternimal structure it is
 *        called when the TPCC.DLL is first loaded by the inet service.
 *
 * ARGUMENTS:   none
 *
 * RETURNS:     None
 *
 * COMMENTS:    None
 *
 */

void TermInit(server_rec *s)
{
    tpcc_mod_conf* mc = myModConfig(s);
    TERM* Term= apr_shm_baseaddr_get(mc->shmTerm.shm);

    if ( Term->bInit )
        return;
    Term->iNext = 0;
    Term->iMasterSyncId = 1;
    Term->iAvailable = 0;
    Term->pClientData = NULL;
    Term->bInit = TRUE;
    return;
}


/* FUNCTION: void TermRestore(void)
 *
 * PURPOSE: This function frees allocated resources associated with the
 * terminal structure.
 *
 * ARGUMENTS:   none
 *
 * RETURNS:     None
 *
 * COMMENTS:    This function is called only with the inet service unloads the
 *            TPCC.DLL
 *
 */

static void TermRestore(EXTENSION_CONTROL_BLOCK* pECB)
{
    Term->iNext = 0;
    Term->iAvailable = 0;
    Term->iMasterSyncId = 0;
    if ( Term->pClientData )
        free(Term->pClientData);
    Term->pClientData = NULL;
    Term->bInit = FALSE;
    return;
}

/* FUNCTION: int TermAllocate(void)
 *
 * PURPOSE: This funct allocates terminal array entries in the Term structure.
 *
 * ARGUMENTS:   None
 *
 * RETURNS:     int TRUE or 1 if sucessfull
 *            int FALSE or 0 if terminal id cannot be allocated.
 *
 * COMMENTS:    None
 *
 */

int TermAllocate(server_rec *s)
{
/*
    Term->iAvailable += 32;

    if ( !Term->pClientData )
        Term->pClientData = (PCLIENTDATA)malloc(Term->iAvailable *
```

```
sizeof(CLIENTDATA));
    else
        term->pClientData = (PCLIENTDATA)realloc(Term->pClientData, Term-
>iAvailable * sizeof(CLIENTDATA));
*/

        TERM* Term;
        apr_size_t new_size;
        tpcc_mod_conf* mc = (tpcc_mod_conf*)myModConfig(s);

        Term = apr_shm_baseaddr_get(mc->shmTerm.shm);
     if (dLog)
        {FILE *fp;          fp = fopen("/tmp/errlog", "ab");
                fprintf(fp, "* In realloc before iAvail=%d\n",Term->iAvailable);

                fclose(fp);
        }

        Term->iAvailable +=iMaxConnections;

        new_size=Term->iAvailable* sizeof (CLIENTDATA);
        Term->pClientData = (PCLIENTDATA)realloc_shm(&(mc-
>shmCData),new_size, mc->pool);
        if (dLog)
                {FILE *fp;          fp = fopen("/tmp/errlog", "ab");
                fprintf(fp, "* In realloc after iAvail=%d, shmCData=%x\n",Term-
>iAvailable,mc->shmCData.shm);
                fclose(fp);
        }

    return ( Term->pClientData ) ? 1 : 0;

        return 1;
}

/* FUNCTION: int TermAdd(EXTENSION_CONTROL_BLOCK *pECB, char
*pQueryString, int iTermId)
 *
 * PURPOSE: This function assigns a terminal id which is used to identify a
 *          client browser.
 *
 * ARGUMENTS: EXTENSION_CONTROL_BLOCK *pECB     passed instructure
pointer
 *                                      from inetsrv.
 *          char            *pQueryString   http query string passed to
 *                                      this DLL.
 *
 *          int             iTermId         terminal id from browser
 * RETURNS:     int             assigned terminal id
 *                      -1      cannot assign id error occured.
 *
 *
 * COMMENTS: if the terminal id cannot be assigned it is because of
 *          insufficient memory or the SQL connection cannot be allocated.
 *
 */

static int TermAdd(EXTENSION_CONTROL_BLOCK *pECB, char *pQueryString)
{
    char      szTmp[32];

    int       i, iCurrent, iTotalConnections, iTickCount;
    EnterCriticalSection(&CriticalSection);
    for(i=0, iTotalConnections = 0; i<Term->iAvailable; i++)
    {
        if ( Term->pClientData[i].inUse )
        iTotalConnections++;
    }

    if ( iTotalConnections >= iMaxConnections )
    {
        for(iCurrent = 1, i=1, iTickCount = 0x7FFFFFFF; i<iMaxConnections; i++)
        {
          if ( iTickCount > Term->pClientData[i].iTickCount )
          {
                iTickCount = Term->pClientData[i].iTickCount;
                iCurrent = i;
          }
        }
    }
    else
    {
        for(i=0; i<Term->iAvailable; i++)
        {
          if ( !Term->pClientData[i].inUse )
          break;
```

```
            }

            iCurrent = i;
        }
        if ( i == Term->iAvailable )
        {
            Term->iNext = Term->iAvailable;

            if ( !TermAllocate(pECB->r->server) )
              goto TermAddErr1;

            for(i=Term->iNext; i<Term->iAvailable; i++)
              Term->pClientData[i].inUse = 0;

            iCurrent = Term->iNext;
        }

        Term->pClientData[iCurrent].inUse = 1;
        if ( !GetKeyValue(pQueryString, "w_id", szTmp, sizeof(szTmp)) )
            goto TermAddErr1;

        Term->pClientData[iCurrent].w_id = (int)atoi(szTmp); //change 2003/07/19
        if ( !GetKeyValue(pQueryString, "d_id", szTmp, sizeof(szTmp)) )
            goto TermAddErr1;

        Term->pClientData[iCurrent].d_id = atoi(szTmp);
        Term->pClientData[iCurrent].iTickCount = GetTickCount();
        Term->pClientData[iCurrent].iSyncId = Term->iMasterSyncId++;
        if ( Init(pECB, iCurrent, Term->pClientData[iCurrent].iSyncId, szServer, szUser,
szPassword, szDatabase) )
        {
              //   WR_LOG("term init failed");
            TermDelete(pECB, iCurrent);
            goto TermAddErr1;
        }

        LeaveCriticalSection(&CriticalSection);

        return iCurrent;

TermAddErr1:
        LeaveCriticalSection(&CriticalSection);
        return -1;   //terminal unsuccessfully added
}


/* FUNCTION: void TermDelete(EXTENSION_CONTROL_BLOCK *pECB, int
id)
 *
 * PURPOSE:  This function makes a terminal entry in the Term array available
 *           for reuse.
 *
 * ARGUMENTS: EXTENSION_CONTROL_BLOCK   *pECB passed in structure
pointer from
 *                                inetsrv.
 *      int             id              Terminal id of client exiting
 *
 * RETURNS:     None
 *
 * COMMENTS:    None
 *
 */

static void TermDelete(EXTENSION_CONTROL_BLOCK *pECB, int id)
{
    if ( id >= 0 && id < Term->iAvailable )
    {
            //TPCC_LOG("Term Delete");
            Close(pECB, id, -1);
            Term->pClientData[id].inUse = 0;

#ifndef LOCAL_ALLOC
        tpfree((char *)Term->pClientData[id].newOrderDataPtr);
        tpfree((char *)Term->pClientData[id].paymentDataPtr);
        tpfree((char *)Term->pClientData[id].orderStatusDataPtr);
        tpfree((char *)Term->pClientData[id].deliveryDataPtr);
        tpfree((char *)Term->pClientData[id].stockLevelDataPtr);
#endif

    }


    return;
}
```

```
/* FUNCTION: BOOL Init(EXTENSION_CONTROL_BLOCK *pECB, int iTermId,
int iSyncId,
 *                  char *szServer, char *szUser, char *szPassword,
 *                  char *szDatabase)
 *
 * PURPOSE: This function initializes the sql connection for use.
 *
 * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB   passed in structure
pointer
 *                                  from inetsrv.
 *      int     iTermId        id of browser client that this connection is for.
 *      int     iSyncId         sync id for this client session
 *      char    *szServer       sql server name
 *      char    *szUser         user name
 *      char    *szPassword     user password
 *      char    *szDatabase     database to use
 *
 * RETURNS:     BOOL    FALSE   if successfull
 *              TRUE    if an error occurs and connection cannot be established.
 *
 * COMMENTS:    None
 *
 */
BOOL Init(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId,
char *szServer, char *szUser, char*szPassword, char *szDatabase)
{
    char      szApp[32];
#ifndef LOCAL_ALLOC
    char      buf[40];
    int       iRc;
#endif

#ifdef TOPEND

    int       i;
    void      space_fill();
    tp_dif_structs_t  *tclient_dif;
    char      ascii_pid[10];
    int       msglen;
    FILE      *fp1;
#else
    sprintf(szApp, "TPCC:%ld", (int)iTermId);
  /*   Term->pClientData[iTermId].dbContext = NULL;     - do not keep context */
#endif


#ifdef TOPEND
    EnterCriticalSection(&CriticalSection);
    if (TP_InitDone == FALSE)
    {
        TP_InitDone = TRUE;
        LeaveCriticalSection (&CriticalSection);

        if((i=(int)tp_ChangeToGroup((LPSTR)getenv("TP_SYSTEM"),
            LOGON32_LOGON_INTERACTIVE,
            LOGON32_PROVIDER_DEFAULT))!=0)
        {
          if (bLog)
          {
                fp1 = fopen(szTpccLogPath,"ab");
                fprintf(fp1,"Bad status from tp_ChangeToGroup,status=%d**\n",i);
                fclose(fp1);
          }
          MessageBox(NULL, "Failed in tp_ChangeToGroup.",
                          "Init", MB_OK | MB_ICONSTOP);
              return TRUE; //changed DMA
        }
        //initialize TOPEND

        if ((i=tp_mt_initialize(NULL,NULL,0L))!=TP_OK)
        {
          if (bLog)
          {
                fp1 = fopen(szTpccLogPath,"ab");
                fprintf(fp1,"Bad status from tp_mt_initialize,status=%d**\n",i);
                fclose(fp1);
          }

          return TRUE; //changed DMA

        }

    } //if init
    else
    LeaveCriticalSection(&CriticalSection);
```

```
    sprintf(szApp,"TPCC:%1d",(int)iTermId);
//    Term->pClientData[iTermId].dbproc=NULL;

    if((i=client_dif= tp_csi_alloc(TP_DIF_ALL))== NULL)
    {
        if (bLog)
        {
         //  SYSTEMTIME systemTime;
          fp1 = fopen(szTpccLogPath,"ab");
          fprintf(fp1,"Bad status from tp_csi_alloc,status=%d**\n",i);
          fclose(fp1);
        }

        ErrorMessage(pECB,TOPEND_SEND_ERROR,ERR_TYPE_WEBDLL,
                NULL,iTermId,iSyncId);

        return TRUE;
    }

    (tp_dif_structs_t*)Term->pClientData[iTermId].tp_structadr = client_dif;
    tclient_dif = (tp_dif_structs_t*)Term->pClientData[iTermId].tp_structadr;

    sprintf(ascii_pid,"%d",iTermId);

    tclient_dif->info->tp_user_dialogue_id = iTermId;
    tclient_dif->info->tp_user_message_id = 0L;
    tclient_dif->info->tp_system_dialogue_id.tp_sys_dialogue = 0L;
    tclient_dif->info->tp_flags = TP_NOFLAGS;

    space_fill (tclient_dif->client->tp_userid,"tpcc cli",TP_USERID_LEN);
    space_fill (tclient_dif->client->tp_endpoint,ascii_pid,TP_ENDPOINT_LEN);
    space_fill (tclient_dif->client->tp_password,"",TP_PASSWORD_LEN);
    space_fill
        (tclient_dif->service->tp_product_name,"tpcc",TP_PROD_NAME_LEN);
    space_fill
        (tclient_dif->output_format->tp_format_name,"",TP_FMT_NAME_LEN);
    space_fill
        (tclient_dif->service->tp_function_name,"tpcc1",TP_FUNC_NAME_LEN);

    tclient_dif->service->tp_function_qualifier = 0;
    wait_time = TP_BLOCK;
    msglen = 0L;

    if ((i = tp_mt_client_signon(tclient_dif->info,tclient_dif->client,
                inactivity_time,NULL,NULL,0,NULL)) != TP_OK)
    {
        if (bLog)
        {
          fp1 = fopen(szTpccLogPath,"ab");
          fprintf(fp1,"client signon failed,status=%d,iTermId=%d,iSyncId=%d\n",
                i,iTermId,iSyncId);
          fclose(fp1);
        }

        ErrorMessage(pECB,TOPEND_SEND_ERROR,ERR_TYPE_WEBDLL,
                NULL,iTermId,iSyncId);

        return TRUE;
    }

    if ((i = tp_mt_client_receive(tclient_dif->info,wait_time,
                tclient_dif->output_format,tclient_dif->service,
                tclient_dif->location,&msglen,NULL)) != TP_OK)
    {
        if (bLog)
        {
          fp1 = fopen(szTpccLogPath,"ab");
          fprintf(fp1,"client receive from signon failed,status=%d\n",i);
          fclose(fp1);
        }

        ErrorMessage(pECB,TOPEND_SEND_ERROR,ERR_TYPE_WEBDLL,
                NULL,iTermId,iSyncId);

        return TRUE;
    }

    if (bLog)
    {
        fp1 = fopen(szTpccLogPath,"ab");
        fprintf(fp1,"client %d signed on.\n",iTermId);
        fclose(fp1);
    }

#else
  #ifdef TUX
```

```
    #ifndef LOCAL_ALLOC    /* only do if not doing local alloc of tuxedo data
structures */
//      Add initialization of Tuxedo Structures

//          NEWORDER

                    if ((Term->pClientData[iTermId].newOrderDataPtr =
(NewOrderData *)tpalloc("CARRAY", NULL, sizeof(NewOrderData))) == NULL)
                    {
                              iRc = tperrno;
                              sprintf(buf, "Tpcalloc %d", iRc);
                              ErrorMessage(pECB, ERR_TPALLOC_BAD,
ERR_TYPE_TUXEDO, buf, 0, 0);
                              {
                                        FILE *fp;

                                        fp = fopen(szTpccLogPath, "ab");
                                        fprintf(fp, ">>>> Init Thread %d : NewOrder
tpalloc failed: iRc = %d \r\n",
                                                  GetCurrentThreadId(), iRc);
                                        fclose(fp);
                              }
                              sprintf(buf, "Tpcalloc %d", iRc);
                              ErrorMessage(pECB, ERR_TPALLOC_BAD,
ERR_TYPE_TUXEDO, buf, 0, 0);
                              return TRUE;
                    }

                    if ( dLog )
                    {
                              FILE *fp;

                              fp = fopen(szTpccLogPath, "ab");
                              fprintf(fp, "* Init Thread %d iTermId %d *
NewOrderDataPtr: %x \r\n",
                                        GetCurrentThreadId(), iTermId, &Term-
>pClientData[iTermId].newOrderDataPtr);
                              fclose(fp);
                    }

//          PAYMENT

                    if ((Term->pClientData[iTermId].paymentDataPtr = (PaymentData
*)tpalloc("CARRAY", NULL, sizeof(PaymentData))) == NULL)
                    {
                              iRc = tperrno;
                              {
                                        FILE *fp;

                                        fp = fopen(szTpccLogPath, "ab");
                                        fprintf(fp, ">>>> Init Thread %d : Payment
tpalloc failed: iRc = %d \r\n",
                                                  GetCurrentThreadId(), iRc);
                                        fclose(fp);
                              }
                              sprintf(buf, "Tpcalloc %d", iRc);
                              ErrorMessage(pECB, ERR_TPALLOC_BAD,
ERR_TYPE_TUXEDO, buf, 0, 0);
                              return TRUE;
                    }

                    if ( dLog )
                    {
                              FILE *fp;

                              fp = fopen(szTpccLogPath, "ab");
                              fprintf(fp, "* Init Thread %d iTermId %d *
PaymentDataPtr: %x \r\n",
                                        GetCurrentThreadId(), iTermId, &Term-
>pClientData[iTermId].paymentDataPtr);
                              fclose(fp);
                    }

//          ORDERSTATUS

                    if ((Term->pClientData[iTermId].orderStatusDataPtr =
(OrderStatusData *)tpalloc("CARRAY", NULL, sizeof(OrderStatusData))) ==
NULL)
                    {
                              iRc = tperrno;
                              {
                                        FILE *fp;

                                        fp = fopen(szTpccLogPath, "ab");
                                        fprintf(fp, ">>>> Init Thread %d : OrderStatus
tpalloc failed: iRc = %d \r\n",
```

```
                                        GetCurrentThreadId(), iRc);
                                fclose(fp);
                        }
                        sprintf(buf, "Tpcalloc %d", iRc);
                        ErrorMessage(pECB, ERR_TPALLOC_BAD,
ERR_TYPE_TUXEDO, buf, 0, 0);
                        return TRUE;
                }

                if ( dLog )
                {
                        FILE *fp;

                        fp = fopen(szTpccLogPath, "ab");
                        fprintf(fp, "* Init Thread %d iTermId %d *
OrderStatusDataPtr: %x \r\n",
                                GetCurrentThreadId(), iTermId, &Term-
>pClientData[iTermId].orderStatusDataPtr);
                        fclose(fp);
                }

//              DELIVERY

                if ((Term->pClientData[iTermId].deliveryDataPtr = (DeliveryData
*)tpalloc("CARRAY", NULL, sizeof(DeliveryData))) == NULL)
                {
                        iRc = tperrno;
                        {
                                FILE *fp;

                                fp = fopen(szTpccLogPath, "ab");
                                fprintf(fp, ">>>> Init Thread %d : Delivery
tpalloc failed: iRc = %d \r\n",
                                        GetCurrentThreadId(), iRc);
                                fclose(fp);
                        }
                        sprintf(buf, "Tpcalloc %d", iRc);
                        ErrorMessage(pECB, ERR_TPALLOC_BAD,
ERR_TYPE_TUXEDO, buf, 0, 0);
                        return TRUE;
                }

                if ( dLog )
                {
                        FILE *fp;

                        fp = fopen(szTpccLogPath, "ab");

                        fprintf(fp, "* Init Thread %d iTermId %d *
DeliveryDataPtr: %x \r\n",
                                GetCurrentThreadId(), iTermId, &Term-
>pClientData[iTermId].deliveryDataPtr);
                        fclose(fp);
                }

//              STOCKLEVEL

                if ((Term->pClientData[iTermId].stockLevelDataPtr =
(StockLevelData *)tpalloc("CARRAY", NULL, sizeof(StockLevelData))) == NULL)
                {
                        iRc = tperrno;
                        {
                                FILE *fp;

                                fp = fopen(szTpccLogPath, "ab");
                                fprintf(fp, ">>>> Init Thread %d : StockLevel
tpalloc failed: iRc = %d \r\n",
                                        GetCurrentThreadId(), iRc);
                                fclose(fp);
                        }
                        sprintf(buf, "Tpcalloc %d", iRc);
                        ErrorMessage(pECB, ERR_TPALLOC_BAD,
ERR_TYPE_TUXEDO, buf, 0, 0);
                        return TRUE;
                }

                if ( dLog )
                {
                        FILE *fp;

                        fp = fopen(szTpccLogPath, "ab");

                        fprintf(fp, "* Init Thread %d iTermId %d *
StockLevelDataPtr: %x \r\n",
                                GetCurrentThreadId(), iTermId, &Term-
>pClientData[iTermId].stockLevelDataPtr);
```

```
                        fclose(fp);
                }
    #endif
  #endif
#endif
    return FALSE;
}


/* FUNCTION: BOOL Close(EXTENSION_CONTROL_BLOCK*pECB, int
iTermId, int iSyncId)
 *
 * PURPOSE:     This function closes the sql connection for use.
 *
 * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB   passed in structure
pointer from
 *                              inetsrv.
 *      int     iTermId         id of browser client that this cxn is for.
 *      int     iSyncId                 sync id of client browser
 *
 * RETURNS:     BOOL    FALSE   if successfull
 *              TRUE    if an error occurs and connection cannot be
 *                      terminated.
 *
 * COMMENTS:    None
 *
 */

static BOOL Close(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId)
{

#ifdef TOPEND

    tp_dif_structs_t *tclient_dif;
    tclient_dif = (tp_dif_structs_t *)Term->pClientData[iTermId].tp_structadr;
        tclient_dif->info->tp_flags = TP_SIGNOFF_IMMED;


    tp_client_signoff(tclient_dif->info);
    tp_csi_free (tclient_dif);
    return TRUE;

#else  /* do any database special closing, nothing at this time */
    return 0;
#endif
    UNUSEDPARAM(iSyncId);

}

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

        szPic++;
    }

    *szDest = 0;
    return;
}
```

```
/* FUNCTION: char *MakeStockLevelForm(int iTermId, int iSyncId, BOOL bInput)
 *
 * PURPOSE:     This function constructs the Stock Level HTML page.
 *
 * ARGUMENTS:   int     iTermId         client browser terminal id
 *              int     iSyncId         client browser sync id
 *              BOOL    bInput          TRUE if form is being constructed for
 *                                      input else FALSE
 *
 * RETURNS:     char *                  A pointer to buffer inside client
 *                                      structure where HTML form is built.
 *
 * COMMENTS:    The internal client buffer is created when the terminal id is
 *              assigned and should not be freed except when the client
 *              terminal id is no longer needed.
 */

static char *MakeStockLevelForm(int iTermId, int iSyncId, BOOL bInput)
{
    char      *szForm;
    szForm = (char *)Term->pClientData[iTermId].szBuffer;
    Term->pClientData[iTermId].trans.stocklevelData.stoin.w_id = (int)Term-
>pClientData[iTermId].w_id; //change 2003/7/19
    Term->pClientData[iTermId].trans.stocklevelData.stoin.d_id = (short)Term-
>pClientData[iTermId].d_id;


        if ( bInput)
        {
                strcpy(szForm, "<HTML><HEAD><TITLE>TPC-C Stock
Level</TITLE></HEAD>\
<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">\
<INPUT TYPE=\"hidden\" NAME=\"PI*\" VALUE=\"\">\
<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"ERROR\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"7\">"); //
STOCK_LEVEL_FORM = 7

                wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">\
<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iTermId,
iSyncId);

                strcat(szForm, "<PRE>                    Stock-Level<BR>");
                wsprintf(szForm+strlen(szForm), "Warehouse: %6.6d
District: %2.2d<BR><BR>",
                Term->pClientData[iTermId].trans.stocklevelData.stoin.w_id,
                Term->pClientData[iTermId].trans.stocklevelData.stoin.d_id);

                strcat(szForm,  "Stock Level Threshold: <INPUT NAME=\"TT*\"
SIZE=2><BR><BR>\
low stock:    <BR><HR><INPUT TYPE=\"submit\" NAME=\"CMD\"
VALUE=\"Process\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\"
VALUE=\"Menu\"></FORM></HTML>");
                strcat(szForm,
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
"                                                                "
                );
        }
        else // Not bInput
        {
```

```
			strcpy(szForm, "<HTML><HEAD><TITLE>TPC-C Stock
Level</TITLE></HEAD>\
<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">\
<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"ERROR\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"7\">"); //
STOCK_LEVEL_FORM = 7

			wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">\
<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iTermId,
iSyncId);

			strcat(szForm, "<PRE>                    Stock-Level<BR>");

			wsprintf(szForm+strlen(szForm), "Warehouse: %6.6d
District: %2.2d<BR><BR>\
Stock Level Threshold: %2.2d<BR><BR>low stock: %3.3d</PRE><BR><HR>",
					Term-
>pClientData[iTermId].trans.stocklevelData.stoin.w_id,
					Term-
>pClientData[iTermId].trans.stocklevelData.stoin.d_id,
					Term-
>pClientData[iTermId].trans.stocklevelData.stoin.threshold,
					Term-
>pClientData[iTermId].trans.stocklevelData.stoout.low_stock);

			strcat(szForm,  "<INPUT TYPE=\"submit\" NAME=\"CMD\"
VALUE=\"..NewOrder..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Payment..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Delivery..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Order-Status..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Stock-Level..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Exit..\">\
</FORM></HTML>");
			strcat(szForm,
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
"                                                                      "
				);
		}


	return szForm;
}


/* FUNCTION: char *MakeMainMenuForm(int iTermId, int iSyncId)
 *
 * PURPOSE:  This function makes the main menu HTML page
 *
 * ARGUMENTS:   int     iTermId        client browser terminal id
 *              int     iSyncId        client browser sync id
 *
 * RETURNS:     char *              A pointer to buffer inside client
 *                                  structure where HTML form is built.
 *
 * COMMENTS:    The internal client buffer is created when the terminal id is
 *              assigned and should not be freed except when the client
 *              terminal id is no longer needed.
 */

static char *MakeMainMenuForm(int iTermId, int iSyncId)
{
	char      *szForm;
	szForm = (char *)Term->pClientData[iTermId].szBuffer;
```

```
    strcpy(szForm, "<HTML><HEAD><TITLE>TPC-C
MainMenu</TITLE></HEAD><BODY>"
                   "Select Desired Transaction.<BR><HR>"
                   "<FORM ACTION=\"tpcc.dll\"METHOD=\"GET\">");

    strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\"
VALUE=\"0\">");
    strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"ERROR\"
VALUE=\"0\">");
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">", MAIN_MENU_FORM);
    strcat(szForm, "<INPUT TYPE=\"submit\" NAME=\"CMD\"
VALUE=\"..NewOrder..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\"VALUE=\"..Payment..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\"VALUE=\"..Delivery..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\"VALUE=\"..Order-Status..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\"VALUE=\"..Stock-Level..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\"VALUE=\"..Exit..\">"
          "                                                    "
          "                                                    "
          "                                                    "
          "                                                    "
          "                                                    "
          "                                                    "
          "                                                    "
          "                                                    "
          "                                                    "
          "                                                    "
          "                                                    "
          "                                                    "
          "                                                    "
          "                                                    "
          "                                                    "
                "                                                    "
                "                                                    "
                "                                                    "
                "                                                    "
                                                    "
                                                    "
                                                    "
    "</FORM>"
    "</HTML>" );
    return szForm;
}

/* FUNCTION: char *MakeWelcomeForm(void)
 *
 * PURPOSE: This function makes the HTML based welcome form
 *
 * ARGUMENTS:   None
 *
 * RETURNS:     char *  A pointer to the static HTML welcome form.
 *
 * COMMENTS:    The welcome form is static.
 */

static char *MakeWelcomeForm(void)
{
    return szWelcomeForm;
}


/* FUNCTION: char *MakeNewOrderForm(int iTermId, BOOL bInput, BOOL
bValid, char *execution_status)
 *
 * PURPOSE:  This function makes/presents the HTML based New Order Form
 *
 * ARGUMENTS:   int     iTermId         client browser terminal id
 *              int     iSyncId         client browser sync id
 *              BOOL    bInput          TRUE if form is being constructed for
 *                                      input else FALSE
 *              BOOL    bValid          TRUE if NeworderData valid, ELSE FALSE
 *                                      effects output only
 *
 * RETURNS:     char *                  A pointer to buffer inside client
 *                                      structure where HTML form is built.
 *
 * COMMENTS:    The internal client buffer is created when the terminal id is
 *              assigned and should not be freed except when the client
 *              terminal id is no longer needed.
```

```
 */

static char *MakeNewOrderForm(int iTermId, int iSyncId, BOOL bInput, BOOL
bValid, char *execution_status)
{
    char     *szForm;
//  char     szName[146];
//  char     szCredit[14];
    int      i;

    szForm = (char *)Term->pClientData[iTermId].szBuffer;

    Term->pClientData[iTermId].trans.newOrderData.newin.w_id = Term-
>pClientData[iTermId].w_id;

        if ( bInput)
        {
                strcpy(szForm,"<HTML><HEAD><TITLE>TPC-C New
Order</TITLE>\
</HEAD><BODY><FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">\
<INPUT TYPE=\"hidden\" NAME=\"PI*\" VALUE=\"\">\
<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"ERROR\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"3\">"); //
NEW_ORDER_FORM = 3

                sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">\
<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">\
<PRE>                        New Order<BR>\
Warehouse: %6.6d   District: <INPUT NAME=\"DID*\" SIZE=1>
Date:<BR>\
Customer:  <INPUT NAME=\"CID*\" SIZE=4>   Name:
Credit:     %Disc:<BR>\
Order Number:         Number of Lines:         W_tax:        D_tax:<BR><BR>\
 Supp_W  Item_Id  Item Name              Qty  Stock  B/G  Price    Amount<BR>",
                                        iTermId,
                                        iSyncId,
                                        Term-
>pClientData[iTermId].trans.newOrderData.newin.w_id);


                strcat(szForm,     "<INPUT NAME=\"SP00*\" SIZE=6> <INPUT
NAME=\"IID00*\" SIZE=6>                    <INPUT NAME=\"Qty00*\"
SIZE=1><BR>\
<INPUT NAME=\"SP01*\" SIZE=6> <INPUT NAME=\"IID01*\" SIZE=6>
<INPUT NAME=\"Qty01*\" SIZE=1><BR>\
<INPUT NAME=\"SP02*\" SIZE=6> <INPUT NAME=\"IID02*\" SIZE=6>
<INPUT NAME=\"Qty02*\" SIZE=1><BR>\
<INPUT NAME=\"SP03*\" SIZE=6> <INPUT NAME=\"IID03*\" SIZE=6>
<INPUT NAME=\"Qty03*\" SIZE=1><BR>\
<INPUT NAME=\"SP04*\" SIZE=6> <INPUT NAME=\"IID04*\" SIZE=6>
<INPUT NAME=\"Qty04*\" SIZE=1><BR>\
<INPUT NAME=\"SP05*\" SIZE=6> <INPUT NAME=\"IID05*\" SIZE=6>
<INPUT NAME=\"Qty05*\" SIZE=1><BR>\
<INPUT NAME=\"SP06*\" SIZE=6> <INPUT NAME=\"IID06*\" SIZE=6>
<INPUT NAME=\"Qty06*\" SIZE=1><BR>\
<INPUT NAME=\"SP07*\" SIZE=6> <INPUT NAME=\"IID07*\" SIZE=6>
<INPUT NAME=\"Qty07*\" SIZE=1><BR>\
<INPUT NAME=\"SP08*\" SIZE=6> <INPUT NAME=\"IID08*\" SIZE=6>
<INPUT NAME=\"Qty08*\" SIZE=1><BR>\
<INPUT NAME=\"SP09*\" SIZE=6> <INPUT NAME=\"IID09*\" SIZE=6>
<INPUT NAME=\"Qty09*\" SIZE=1><BR>\
<INPUT NAME=\"SP10*\" SIZE=6> <INPUT NAME=\"IID10*\" SIZE=6>
<INPUT NAME=\"Qty10*\" SIZE=1><BR>\
<INPUT NAME=\"SP11*\" SIZE=6> <INPUT NAME=\"IID11*\" SIZE=6>
<INPUT NAME=\"Qty11*\" SIZE=1><BR>\
<INPUT NAME=\"SP12*\" SIZE=6> <INPUT NAME=\"IID12*\" SIZE=6>
<INPUT NAME=\"Qty12*\" SIZE=1><BR>\
<INPUT NAME=\"SP13*\" SIZE=6> <INPUT NAME=\"IID13*\" SIZE=6>
<INPUT NAME=\"Qty13*\" SIZE=1><BR>\
<INPUT NAME=\"SP14*\" SIZE=6> <INPUT NAME=\"IID14*\" SIZE=6>
<INPUT NAME=\"Qty14*\" SIZE=1><BR>\
Execution Status:                        Total:<BR><HR>\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Process\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Menu\">\
</FORM></HTML>" );
                strcat(szForm,""
                );
        }
                else // not bInput
        {
                if (bValid)
                {
//              FormatHTMLString(szName, Term-
>pClientData[iTermId].NewOrderData.c_last, 16),
```

```
//              FormatHTMLString(szCredit, Term-
>pClientData[iTermId].NewOrderData.c_credit, 2);

                    strcpy(szForm,"<HTML><HEAD><TITLE>TPC-C New
Order</TITLE>\
</HEAD><BODY><FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">\
<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"ERROR\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"3\">"); //
NEW_ORDER_FORM = 3

sprintf(szForm+strlen(szForm),"<INPUT TYPE=\"hidden\" NAME=\"TERMID\"
VALUE=\"%d\">\
<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">\
<PRE>                        New Order<BR>\
Warehouse: %6.6d   District: %2.2d                    Date: %s
<BR>\
Customer:  %4.4d   Name: %s   Credit: %2.2s   %%Disc: %5.2f          <BR>\
Order Number: %8.8d  Number of Lines: %2.2d       W_tax: %5.2f   D_tax: %5.2f
<BR><BR>\
 Supp_W  Item_Id  Item Name              Qty  Stock  B/G  Price    Amount<BR>",
                                    iTermId,
                                    iSyncId,
                                    Term-
>pClientData[iTermId].trans.newOrderData.newin.w_id,
                                    Term-
>pClientData[iTermId].trans.newOrderData.newin.d_id,
                                    Term-
>pClientData[iTermId].trans.newOrderData.newout.o_entry_d,
                                    Term-
>pClientData[iTermId].trans.newOrderData.newin.c_id,
                                    Term-
>pClientData[iTermId].trans.newOrderData.newout.c_last,
                                    Term-
>pClientData[iTermId].trans.newOrderData.newout.c_credit,
                                    Term-
>pClientData[iTermId].trans.newOrderData.newout.c_discount * 100,
                                    Term-
>pClientData[iTermId].trans.newOrderData.newout.o_id,
                                    Term-
>pClientData[iTermId].trans.newOrderData.newout.o_ol_cnt,
                                    Term-
>pClientData[iTermId].trans.newOrderData.newout.w_tax * 100,
                                    Term-
>pClientData[iTermId].trans.newOrderData.newout.d_tax * 100);

// We could make 0-4 be hard coded in one sprintf, since we know that there is at least
5
// order lines, or is this a benchmark special??
// etw
                    for(i=0; i<Term-
>pClientData[iTermId].trans.newOrderData.newout.o_ol_cnt; i++)
                                    {
//                                  FormatHTMLString(szName, Term-
>pClientData[iTermId].NewOrderData.Ol[i].ol_i_name, 24);

                                    sprintf(szForm+strlen(szForm),
"%6.6d   %6.6d   %-24.24s  %2.2d    %3.3d    %1c   $%6.2f  $%7.2f  <BR>",
                                    //                Term-
>pClientData[iTermId].trans.newOrderData.newin.ol_supply_w_id[i],//change
                                    //                Term-
>pClientData[iTermId].trans.newOrderData.newin.ol_i_id[i],//change
                                    Term-
>pClientData[iTermId].trans.newOrderData.newout.ol_supply_w_id[i],//change
                                    Term-
>pClientData[iTermId].trans.newOrderData.newout.ol_i_id[i],//change
                                    Term-
>pClientData[iTermId].trans.newOrderData.newout.i_name[i],
                                    Term-
>pClientData[iTermId].trans.newOrderData.newout.ol_quantity[i],//change
                                    //                Term-
>pClientData[iTermId].trans.newOrderData.newin.ol_quantity[i],//change
                                    Term-
>pClientData[iTermId].trans.newOrderData.newout.s_quantity[i],
                                    Term-
>pClientData[iTermId].trans.newOrderData.newout.brand_generic[i],
                                    Term-
>pClientData[iTermId].trans.newOrderData.newout.i_price[i],
                                    Term-
>pClientData[iTermId].trans.newOrderData.newout.ol_amount[i] );
                                    }
                            for(; i<15; i++)
                                    strcat(szForm, "<BR>");

                                    sprintf(szForm+strlen(szForm), "Execution
Status: %24.24s                Total:  $%8.2f  ",
```

```
                                                execution_status,
                                                Term-
>pClientData[iTermId].trans.newOrderData.newout.total_amount);

                                strcat(szForm, "</PRE><HR><BR>\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..NewOrder..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Payment..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Delivery..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Order-Status..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Stock-Level..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Exit..\">\
</FORM></HTML>");

                        }
                        else // not bValid
                        {

//              FormatHTMLString(szName, Term-
>pClientData[iTermId].NewOrderData.c_last, 16),
//              FormatHTMLString(szCredit, Term-
>pClientData[iTermId].NewOrderData.c_credit, 2);
// statusid set to 1 for benchcraft - indicator for invalid item
                        strcpy(szForm,"<HTML><HEAD><TITLE>TPC-C New
Order</TITLE>\
</HEAD><BODY><FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">\
<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"1\">\
<INPUT TYPE=\"hidden\" NAME=\"ERROR\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"3\">");  //
NEW_ORDER_FORM = 3

                        sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">\
<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">\
<PRE>                           New Order<BR>\
Warehouse: %6.6d   District: %2.2d                  Date:<BR>\
Customer:  %4.4d   Name: %s   Credit: %s   %Disc:<BR>\
Order Number: %8.8d  Number of Lines:          W_tax:        D_tax:<BR><BR>",
                                        iTermId,
                                        iSyncId,
                                        Term-
>pClientData[iTermId].trans.newOrderData.newin.w_id,
                                          Term-
>pClientData[iTermId].trans.newOrderData.newin.d_id,
                                        Term-
>pClientData[iTermId].trans.newOrderData.newin.c_id,
                                        Term-
>pClientData[iTermId].trans.newOrderData.newout.c_last,
                                        Term-
>pClientData[iTermId].trans.newOrderData.newout.c_credit,
                                        Term-
>pClientData[iTermId].trans.newOrderData.newout.o_id
                                        );

                        strcat(szForm, " Supp_W  Item_Id  Item Name              Qty  Stock
B/G  Price    Amount<BR>\
<BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR><BR>
<BR>");

                        sprintf(szForm+strlen(szForm), "Execution Status: %24.24s
Total:",
                                        execution_status);

                        strcat(szForm, "</PRE><HR><BR>\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..NewOrder..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Payment..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Delivery..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Order-Status..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Stock-Level..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Exit..\">\
</FORM></HTML>");
                        strcat(szForm,
"                                                               "
"                                                               "
"                                                               "
"                                                               "
                        );
                        }
                }

        return szForm;
}

/* FUNCTION: char *MakePaymentForm(int iTermId, int iSyncId, BOOL bInput)
 *
 * PURPOSE:  This function has the Make Payment HTTP form
 *
```

```
 * ARGUMENTS:   int     iTermId client browser terminal id
 *              int     iSyncId client browser sync id
 *              BOOL    bInput  TRUE if form is being constructed for input
 *                      else FALSE
 *
 * RETURNS:     char *  A pointer to buffer inside client structure where HTML
 *                      form is built.
 *
 * COMMENTS: The internal client buffer is created when the terminal id is
 *           assigned and should not be freed except when the client terminal
 *           id is no longer needed.
 */
static char *MakePaymentForm(int iTermId, int iSyncId, BOOL bInput)
{
    char     *szForm;
    char     *ptr;
    char     szTmp[64];
    char     szW_Zip[26];
    char     szD_Zip[26];
    char     szC_Zip[26];
    char     szC_Phone[26];
//  char     szTmpStr1[122];
//  char     szTmpStr2[122];
//  char     szTmpStr3[122];
//  char     szTmpStr4[122];
    int      i;
    int      l;
    char     *szZipPic = "XXXXX-XXXX";


    szForm = (char *)Term->pClientData[iTermId].szBuffer;

    Term->pClientData[iTermId].trans.paymentData.payin.w_id = Term-
>pClientData[iTermId].w_id;

        if ( bInput )
        {
                strcpy(szForm,  "<HTML><HEAD><TITLE>TPC-C
Payment</TITLE></HEAD><BODY>\
<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">\
<INPUT TYPE=\"hidden\" NAME=\"PI*\" VALUE=\"\">\
<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"ERROR\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"4\">"); //
PAYMENT_FORM = 4

                sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">\
<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">\
<PRE>                   Payment<BR>\
Date:<BR><BR>\
Warehouse: %6.6d",
                                                        iTermId,
                                                        iSyncId,
                                                        Term-
>pClientData[iTermId].trans.paymentData.payin.w_id);

                strcat(szForm,
"               District: <INPUT NAME=\"DID*\"
SIZE=1><BR><BR><BR><BR><BR>\
Customer: <INPUT NAME=\"CID*\" SIZE=4>\
Cust-Warehouse: <INPUT NAME=\"CWI*\" SIZE=6>  \
Cust-District: <INPUT NAME=\"CDI*\" SIZE=1><BR>\
Name:           <INPUT NAME=\"CLT*\" SIZE=16>           Since:<BR>\
                                                        Credit:<BR>\
                                                        Disc:<BR>\
                                                        Phone:<BR><BR>\
Amount Paid:    $<INPUT NAME=\"HAM*\" SIZE=7>   New Cust
Balance:<BR>\
Credit Limit:<BR><BR>Cust-Data: <BR><BR><BR><BR></PRE><HR>\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Process\"><INPUT
TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Menu\">\
</BODY></FORM></HTML>");
                strcat(szForm,""
                );
        }
        else // Not bInput
        {

//          FormatHTMLString(szTmpStr1, Term-
>pClientData[iTermId].PaymentData.w_street_1, 20);
//          FormatHTMLString(szTmpStr2, Term-
>pClientData[iTermId].PaymentData.d_street_1, 20);

//          FormatHTMLString(szTmpStr3, Term-
>pClientData[iTermId].PaymentData.w_street_2, 20);
```

```
//          FormatHTMLString(szTmpStr4, Term-
>pClientData[iTermId].PaymentData.d_street_2, 20);

                    FormatString(szW_Zip, szZipPic, Term-
>pClientData[iTermId].trans.paymentData.payout.w_zip);
                    FormatString(szD_Zip, szZipPic, Term-
>pClientData[iTermId].trans.paymentData.payout.d_zip);

//          FormatHTMLString(szTmpStr5, Term-
>pClientData[iTermId].PaymentData.w_city, 20);
//          FormatHTMLString(szTmpStr6, Term-
>pClientData[iTermId].PaymentData.w_state, 2);
//          FormatHTMLString(szTmpStr7, Term-
>pClientData[iTermId].PaymentData.d_city, 20);
//          FormatHTMLString(szTmpStr8, Term-
>pClientData[iTermId].PaymentData.d_state, 2);

//          FormatHTMLString(szTmpStr9, Term-
>pClientData[iTermId].PaymentData.c_first, 16);
//          FormatHTMLString(szTmpStr10, Term-
>pClientData[iTermId].PaymentData.c_middle, 2);
//          FormatHTMLString(szTmpStr11, Term-
>pClientData[iTermId].PaymentData.c_last, 16);

//          FormatHTMLString(szTmpStr12, Term-
>pClientData[iTermId].PaymentData.c_street_1, 20);
//          FormatHTMLString(szTmpStr13, Term-
>pClientData[iTermId].PaymentData.c_credit, 2);
//
//          FormatHTMLString(szTmpStr14, Term-
>pClientData[iTermId].PaymentData.d_street_2, 20);

                    FormatString(szC_Zip, szZipPic, Term-
>pClientData[iTermId].trans.paymentData.payout.c_zip);
                    FormatString(szC_Phone, "XXXXXX-XXX-XXX-XXXX",Term-
>pClientData[iTermId].trans.paymentData.payout.c_phone);

//          FormatHTMLString(szTmpStr15, Term-
>pClientData[iTermId].PaymentData.c_city, 20);
//          FormatHTMLString(szTmpStr16, Term-
>pClientData[iTermId].PaymentData.c_state, 2);

                    strcpy(szForm,  "<HTML><HEAD><TITLE>TPC-C
Payment</TITLE></HEAD><BODY>\
<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">\
<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"ERROR\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"4\">"); //
PAYMENT_FORM = 4

                    sprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">\
<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">\
<PRE>                    Payment<BR>\
Date: %19s<BR><BR>\
Warehouse: %6.6d                District: %2.2d<BR>\
%-20s              %-20s<BR>\
%-20s              %-20s<BR>\
%-20s %-2s %10.10s      %-20s %-2s %10.10s<BR><BR>\
Customer: %4.4d  Cust-Warehouse: %6.6d  Cust-District: %2.2d<BR>\
Name:   %-16s %-2s %-16s     Since:  %10s<BR>\
          %-20s              Credit: %s<BR>\
          %-20s              %%Disc:  %5.2f<BR>\
          %-20s %-2s %10.10s      Phone:  %-19.19s<BR><BR>\
Amount Paid:        $%7.2f      New Cust Balance: $%14.2f<BR>\
Credit Limit:   $%13.2f<BR><BR>",

                    iTermId,
                    iSyncId,
                    Term-
>pClientData[iTermId].trans.paymentData.payout.h_date,
                    Term-
>pClientData[iTermId].trans.paymentData.payin.w_id,
                    Term-
>pClientData[iTermId].trans.paymentData.payin.d_id,
                    Term-
>pClientData[iTermId].trans.paymentData.payout.w_street_1,
                    Term-
>pClientData[iTermId].trans.paymentData.payout.d_street_1,
                    Term-
>pClientData[iTermId].trans.paymentData.payout.w_street_2,
                    Term-
>pClientData[iTermId].trans.paymentData.payout.d_street_2,
                    Term-
>pClientData[iTermId].trans.paymentData.payout.w_city,
                    Term-
```

```
>pClientData[iTermId].trans.paymentData.payout.w_state,
                        szW_Zip,
                        Term-
>pClientData[iTermId].trans.paymentData.payout.d_city,
                        Term-
>pClientData[iTermId].trans.paymentData.payout.d_state,
                        szD_Zip,
                        Term-
>pClientData[iTermId].trans.paymentData.payout.c_id,
                        Term-
>pClientData[iTermId].trans.paymentData.payin.c_w_id,
                        Term-
>pClientData[iTermId].trans.paymentData.payin.c_d_id,
                        Term-
>pClientData[iTermId].trans.paymentData.payout.c_first,
                        Term-
>pClientData[iTermId].trans.paymentData.payout.c_middle,
                        Term-
>pClientData[iTermId].trans.paymentData.payout.c_last,
                        Term-
>pClientData[iTermId].trans.paymentData.payout.c_since,
                        Term-
>pClientData[iTermId].trans.paymentData.payout.c_street_1,
                        Term-
>pClientData[iTermId].trans.paymentData.payout.c_credit,
                        Term-
>pClientData[iTermId].trans.paymentData.payout.c_street_2,
                        Term-
>pClientData[iTermId].trans.paymentData.payout.c_discount*100,
                        Term-
>pClientData[iTermId].trans.paymentData.payout.c_city,
                        Term-
>pClientData[iTermId].trans.paymentData.payout.c_state,
                        szC_Zip, szC_Phone,
                        (float)(Term-
>pClientData[iTermId].trans.paymentData.payin.h_amount)/100.0,
                        Term-
>pClientData[iTermId].trans.paymentData.payout.c_balance,
                        Term-
>pClientData[iTermId].trans.paymentData.payout.c_credit_lim);

// This part can probably be made a little more efficient
//etw
                ptr = Term-
>pClientData[iTermId].trans.paymentData.payout.c_credit;
//for dedug katsuf
                        if (bLog){
                                FILE *fp1;
                                fp1 = fopen(szTpccLogPath, "ab");
                                fprintf(fp1, "c_cre= %s", ptr);
                                fprintf(fp1, "c_da = %s",Term-
>pClientData[iTermId].trans.paymentData.payout.c_data);
                                fclose(fp1);
                        }
                if ( *ptr == 'B' && *(ptr+1) == 'C' )
                {
                        ptr = Term-
>pClientData[iTermId].trans.paymentData.payout.c_data;
                        l = strlen( ptr ) / 50;
                        for(i=0; i<4; i++, ptr += 50)
                        {
                                if ( i <= l )
                                        UtilStrCpy(szTmp, ptr, 50);
                                else
                                        szTmp[0] = 0;
                                if ( !i )
                                {
//              FormatHTMLString(szTmpStr1, szTmp, 50);
                                        wsprintf(szForm+strlen(szForm), "Cust-
Data: %-50s<BR>", szTmp);
                                }
                                else
                                {
//              FormatHTMLString(szTmpStr1, szTmp, 50);
                                        wsprintf(szForm+strlen(szForm),
"       %-50s<BR>", szTmp);
                                }
                        }
                }
                else {
                        strcat(szForm, "Cust-Data: <BR><BR><BR><BR>");
                }

                strcat(szForm,  "</PRE><HR><BR>\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..NewOrder..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Payment..\">\
```

```
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Delivery..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Order-Status..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Stock-Level..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Exit..\">\
</BODY></FORM></HTML>");

        }

        return szForm;
}


/* FUNCTION: char *MakeOrderStatusForm(int iTermId, int iSyncId, BOOL bInput)
 *
 * PURPOSE: This function makes the HTML based Order Status Form
 *
 * ARGUMENTS:   int     iTermId client browser terminal id
 *          int     iSyncId client browser sync id
 *          BOOL    bInput  TRUE if form is being constructed for input
 *                  else FALSE
 *
 * RETURNS:     char *          A pointer to buffer inside client structure
 *                      where HTML form is built.
 *
 * COMMENTS:    The internal client buffer is created when the terminal id is
 *          assigned and should not be freed except when the client
 *          terminal id is no longer needed.
 */
static char *MakeOrderStatusForm(int iTermId, int iSyncId, BOOL bInput)
{
    char     *szForm;
//     char    c_first[98];
//     char    c_middle[14];
//     char    c_last[98];
    int      i;

    szForm = (char *)Term->pClientData[iTermId].szBuffer;

    Term->pClientData[iTermId].trans.orderStatusData.ordin.w_id = Term-
>pClientData[iTermId].w_id;

        if ( bInput )
        {
                strcpy(szForm,  "<HTML><HEAD><TITLE>TPC-C Order-
Status</TITLE></HEAD><BODY>\
<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">\
<INPUT TYPE=\"hidden\" NAME=\"PI*\" VALUE=\"\">\
<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"ERROR\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"6\">"); //
ORDER_STATUS_FORM = 6

                wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">\
<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">",
                                                        iTermId,
                                                        iSyncId);

                strcat(szForm,  "<PRE>                          Order-Status<BR>" );
                wsprintf(szForm+strlen(szForm), "Warehouse: %6.6d   ", Term-
>pClientData[iTermId].trans.orderStatusData.ordin.w_id);

                strcat(szForm,  "District: <INPUT NAME=\"DID*\"
SIZE=1><BR>\
Customer: <INPUT NAME=\"CID*\" SIZE=4>   Name:                  <INPUT
NAME=\"CLT*\" SIZE=23><BR>\
Cust-Balance:<BR><BR>\
Order-Number:           Entry-Date:                 Carrier-Number:<BR>\
Supply-W     Item-Id    Qty     Amount      Delivery-Date<BR></PRE>\
<HR><INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Process\"><INPUT
TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Menu\">\
</BODY></FORM></HTML>" );
                strcat(szForm,
"                                                               "
"                                                               "
"                                                               "
"                                                               "
"                                                               "
"                                                               "
                );
        }
        else // Not bInput
        {
                strcpy(szForm,  "<HTML><HEAD><TITLE>TPC-C Order-
Status</TITLE></HEAD><BODY>\
<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">\
```

```
<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"ERROR\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"6\">"); //
ORDER_STATUS_FORM = 6

                wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">\
<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">",
                                                iTermId,
                                                iSyncId);
        strcat(szForm,  "<PRE>                  Order-Status<BR>" );

        sprintf(szForm+strlen(szForm), "Warehouse: %6.6d   District: %2.2d<BR>\
Customer: %4.4d   Name: %-16s %-2s %-16s<BR>\
Cust-Balance: $%9.2f<BR><BR>\
Order-Number: %8.8d   Entry-Date: %19s                  Carrier-
Number: %2.2d<BR>",
                                Term-
>pClientData[iTermId].trans.orderStatusData.ordin.w_id,
                                Term-
>pClientData[iTermId].trans.orderStatusData.ordin.d_id,
                                Term-
>pClientData[iTermId].trans.orderStatusData.ordout.c_id,
                                Term-
>pClientData[iTermId].trans.orderStatusData.ordout.c_first,
                                Term-
>pClientData[iTermId].trans.orderStatusData.ordout.c_middle,
                                Term-
>pClientData[iTermId].trans.orderStatusData.ordout.c_last,
                                Term-
>pClientData[iTermId].trans.orderStatusData.ordout.c_balance,
                                Term-
>pClientData[iTermId].trans.orderStatusData.ordout.o_id,
                                Term-
>pClientData[iTermId].trans.orderStatusData.ordout.o_entry_d,
                                Term-
>pClientData[iTermId].trans.orderStatusData.ordout.o_carrier_id);

                strcat(szForm+strlen(szForm), "Supply-W     Item-Id    Qty
Amount      Delivery-Date<BR>");

                for(i=0; i<Term-
>pClientData[iTermId].trans.orderStatusData.ordout.o_ol_cnt; i++)
                {
                        sprintf(szForm+strlen(szForm),
"  %6.6d     %6.6d     %2.2d     $%8.2f      %10s            <BR>",
                                Term-
>pClientData[iTermId].trans.orderStatusData.ordout.ol_supply_w_id[i],
                                Term-
>pClientData[iTermId].trans.orderStatusData.ordout.ol_i_id[i],
                                Term-
>pClientData[iTermId].trans.orderStatusData.ordout.ol_quantity[i],
                                Term-
>pClientData[iTermId].trans.orderStatusData.ordout.ol_amount[i],
                                Term-
>pClientData[iTermId].trans.orderStatusData.ordout.ol_delivery_d[i]);

                }

                strcat(szForm,  "<BR></PRE><HR><INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..NewOrder..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Payment..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Delivery..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Order-Status..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Stock-Level..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Exit..\">\
</BODY></FORM></HTML>" );
                                strcat(szForm,
"                                                       "
"                                                       "
"                                                       "
                );
        }

        return szForm;
}

/* FUNCTION: char *MakeDeliveryForm(int iTermId, int iSyncId, BOOL bInput,
char *execution_status)
 *
 * PURPOSE:     This function puts out the HTML for the delivery form
 *
 * ARGUMENTS:   int     iTermId client browser terminal id
 *              int     iSyncId client browser sync id
 *              BOOL    bInput  TRUE if form is being constructed for input
 *                      else FALSE
```

```
 *
 * RETURNS:     char *  A pointer to buffer inside client structure where HTML
 *                   form is built.
 *
 * COMMENTS: The internal client buffer is created when the terminal id is
 *          assigned and should not be freed except when the client terminal
 *          id is no longer needed.
 */
static char *MakeDeliveryForm(int iTermId, int iSyncId, BOOL bInput, char
*execution_status)
{

     char      *szForm;


     szForm = (char *)Term->pClientData[iTermId].szBuffer;
     Term->pClientData[iTermId].trans.deliveryData.delin.w_id = Term-
>pClientData[iTermId].w_id;

            if ( bInput )
            {
                         strcpy( szForm, "<HTML><HEAD><TITLE>TPC-C
Delivery</TITLE></HEAD><BODY>\
<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">\
<INPUT TYPE=\"hidden\" NAME=\"PI*\" VALUE=\"\">\
<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"ERROR\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"5\">"); //
DELIVERY_FORM = 5

                         wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">\
<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">",
                                                                  iTermId,
                                                                  iSyncId);

                         strcat(szForm,  "<PRE>                     Delivery<BR>" );

                         wsprintf(szForm+strlen(szForm), "Warehouse: %6.6d<BR><BR>",
Term->pClientData[iTermId].trans.deliveryData.delin.w_id);

                         strcat( szForm, "Carrier Number: <INPUT NAME=\"OCD*\"
SIZE=1><BR><BR>\
Execution Status:<BR></PRE>\
<HR><INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Process\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Menu\">\
</BODY></FORM></HTML>" );
                         strcat(szForm,
"                                                                  "
"                                                                  "
"                                                                  "
"                                                                  "
"                                                                  "
"                                                                  "
"                                                                  "
"                                                                  "
"                                                                  "
"                                                                  "
"                                                                  "
"                                                                  "
"                                                                  "
"                                                                  "
"                                                                  "
"                                                                  "
"                                                                  "
"                                                                  "
                         );

            }
            else // Not bInput
            {
                         strcpy( szForm, "<HTML><HEAD><TITLE>TPC-C
Delivery</TITLE></HEAD><BODY>\
<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">\
<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"ERROR\" VALUE=\"0\">\
<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"5\">"); //
DELIVERY_FORM = 5

                         wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">\
<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">",
                                                                  iTermId,
                                                                  iSyncId);

            strcat(szForm,  "<PRE>                     Delivery<BR>" );
```

```
        wsprintf(szForm+strlen(szForm), "Warehouse: %6.6d<BR><BR>", Term-
>pClientData[iTermId].trans.deliveryData.delin.w_id);



                wsprintf(szForm+strlen(szForm), "Carrier
Number: %2.2d<BR><BR>\
Execution Status: %25.25s<BR></PRE>",
                                                        Term-
>pClientData[iTermId].trans.deliveryData.delin.o_carrier_id,
                                                        execution_status);


                strcat(szForm,  "<HR><INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..NewOrder..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Payment..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Delivery..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Order-Status..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Stock-Level..\">\
<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Exit..\">\
</BODY></FORM></HTML>" );

                strcat(szForm,
"                                                       "
"                                                       "
"                                                       "
"                                                       "
"                                                       "
"                                                       "
"                                                       "
"                                                       "
"                                                       "
"                                                       "
"                                                       "
"                                                       "
"                                                       "
"                                                       "
"                                                       "
                );
        }

        return szForm;
}
```

```
/* FUNCTION: void UtilStrCpy(char * pDest, char * pSrc, int n)
 *
 * PURPOSE: This function copies n characters from string pSrc to pDst and
 *          places a null character at the end of the destination string.
 *
 * ARGUMENTS:   char    *pDest  destination string pointer
 *              char    *pSrc   source string pointer
 *              int     n       number of characters to copy
 *
 * RETURNS:     None
 *
 * COMMENTS: Unlike strncpy this function ensures that the result string is
 *          always null terminated.
 *
 */

static void UtilStrCpy(char * pDest, char * pSrc, int n)
{
   strncpy(pDest, pSrc, n);
   pDest[n] = '\0';

   return;
}


/* FUNCTION: void ProcessNewOrderForm(EXTENSION_CONTROL_BLOCK
*pECB, int
 *                              iTermId, int iSyncId)
 *
 * PURPOSE: This function gets and validates the input data from the new order
 * form filling in the required input variables. it then calls the SQLNewOrder
 * transaction, constructs the output form and writes it back to client browser.
 *
 * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB   passed in structure
pointer
 *                                      from inetsrv.
 *      int     iTermId client browser terminal id
 *      int     iSyncId client browser sync id
 *
 * RETURNS:     None
 *
```

```
 * COMMENTS:    None
 *
 */

static void ProcessNewOrderForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId)
{
#ifdef TOPEND
     int                 i,iRc;
     int msglen;
     tp_dif_structs_t       *tclient_dif;
     void space_fill();
#else  // assume tuxedo at this point
          int                 iRc, iError;
          long ilen, *olen;
          char buf[40];
#endif

#ifdef LOCAL_ALLOC
     NewOrderData *newOrderDataPtr;
#endif


#ifdef TUX

    // write_err_log(pECB->r,"in new order");

    if((iRc = ThrTpInit(pECB)) <0)
    {
          // This is bad
                {
                     FILE *fp;

                     fp = fopen(szTpccLogPath, "ab");
                     fprintf(fp, "* ProcessNewOrder Thread %d iTermId %d
Failed ThrTpInit \r\n",
                               GetCurrentThreadId(), iTermId );
                     fclose(fp);
                }
                sprintf(buf, "NewOrder ThrTpInit < 0 >> Thread %d",
GetCurrentThreadId());
          ErrorMessage(pECB, ERR_TPINIT_BAD, ERR_TYPE_TUXEDO, buf, 0,
0);
          return;//change 2003/7/11
    }

#endif

     memset(&Term->pClientData[iTermId].trans.newOrderData, 0,
sizeof(NewOrderData));
     Term->pClientData[iTermId].trans.newOrderData.newin.w_id = Term-
>pClientData[iTermId].w_id;

     if ( (iError=GetNewOrderData(pECB->lpszQueryString, &Term-
>pClientData[iTermId].trans.newOrderData)) != ERR_SUCCESS )
          {
               ErrorMessage(pECB, iError, ERR_TYPE_WEBDLL, NULL,
iTermId, iSyncId);
               return;
          }

#ifdef TOPEND

  tclient_dif = (tp_dif_structs_t *)Term->pClientData[iTermId].tp_structadr;

  // Input top end neworder data
  /* Initialize Top End for tpcc transactions */
     tclient_dif->info->tp_user_dialogue_id = iTermId;
     if (bGeneric)
          space_fill(tclient_dif->service->tp_function_name,
"all_txn",TP_FUNC_NAME_LEN);
     else
          space_fill(tclient_dif->service->tp_function_name,
"neworder",TP_FUNC_NAME_LEN);

     space_fill(tclient_dif->service->tp_product_name, "tpcc",
TP_PROD_NAME_LEN);
     space_fill(tclient_dif->input_format->tp_format_name, "",
TP_FMT_NAME_LEN);
     space_fill(tclient_dif->input_format->tp_format_language, "", TP_LANG_LEN);
     space_fill(tclient_dif->input_format->tp_format_type, "", TP_FMTTYPE_LEN);
     tclient_dif->service->tp_function_qualifier = 0;
     msglen = sizeof(CLIENTDATA) - TOPEND_STRUCT_OFFSET;

     Term->pClientData[iTermId].type = 'N';
     if ((i = tp_mt_client_send(tclient_dif->info, tclient_dif->service, tclient_dif-
```

```
>input_format, msglen,  (char *)&Term->pClientData[iTermId].type)) != TP_OK)
    {

        if (bLog)
            {
             FILE *fp1;
             fp1 = fopen(szTpccLogPath, "ab");
             fprintf(fp1, "tp_mt_client_send returned error status = %d\n, i");
             fclose(fp1);
            }

        ErrorMessage(pECB, TOPEND_SEND_ERROR, ERR_TYPE_WEBDLL,
NULL, iTermId, iSyncId);
        return;
    }

    msglen = sizeof(CLIENTDATA) - TOPEND_STRUCT_OFFSET;

    if ((i = tp_mt_client_receive(tclient_dif->info, wait_time, tclient_dif-
>output_format, tclient_dif->service, tclient_dif->location, &msglen, (char *)&Term-
>pClientData[iTermId].type)) != TP_OK)
    {
        if (bLog)
            {
             FILE *fp1;
             fp1 = fopen(szTpccLogPath, "ab");
             fprintf(fp1, "Received error status = %d from client receive for
neworder transaction\n", i);
             fclose(fp1);
            }

        ErrorMessage(pECB, TOPEND_RECEIVE_ERROR,
ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
            return;
    }


    iRc = Term->pClientData[iTermId].retval


#else    // assume we are running tuxedo at this point
  #ifdef LOCAL_ALLOC    // using local alloc method for tuxedo

        if ((newOrderDataPtr = (NewOrderData *) tpalloc("CARRAY", NULL,
sizeof(NewOrderData))) == NULL)
        {
            iRc = tperrno;
            {
                FILE *fp;

                fp = fopen(szTpccLogPath, "ab");
                fprintf(fp, "* ProcessNewOrder Thread %d iTermId %d
tpalloc tperrno %d \r\n",
                    GetCurrentThreadId(), iTermId, iRc);
                fclose(fp);
            }
            sprintf(buf, "NewOrder Tpalloc %d", iRc);
            ErrorMessage(pECB, ERR_TPALLOC_BAD,
ERR_TYPE_TUXEDO, buf, 0, 0);
            tpfree((char *)newOrderDataPtr);
            return;
        }

        *newOrderDataPtr = Term->pClientData[iTermId].trans.newOrderData;
         ilen = sizeof(NewOrderData);
         olen = &ilen;

    if ( dLog )
        {
            FILE *fp;

            fp = fopen(szTpccLogPath, "ab");
            fprintf(fp, "* ProcessNewOrder Thread %d iTermId %d
newOrderDataPtr: %x \r\n",
                GetCurrentThreadId(), iTermId, &newOrderDataPtr);
            fclose(fp);
        }

    if (tpcall("OPSTUXSERVER", (char *)newOrderDataPtr, ilen, (char
**)&newOrderDataPtr, (long *)olen, TPSIGRSTRT) == -1)
        {
       iRc = tperrno;
            {
                FILE *fp;
```

```
                        fp = fopen(szTpccLogPath, "ab");
                        fprintf(fp, "* ProcessNewOrder Thread %d iTermId %d
tpcall tperrno %d \r\n",
                                GetCurrentThreadId(), iTermId, iRc);
                        fclose(fp);
                }
                sprintf(buf, "Neworder tpcall %d", iRc);
          ErrorMessage(pECB, ERR_TPCALL_BAD, ERR_TYPE_TUXEDO, buf,
iTermId, iSyncId);
                tpfree((char *)newOrderDataPtr);
      return;
   }

        if ( dLog )
        {
                FILE *fp;

                fp = fopen(szTpccLogPath, "ab");
                fprintf(fp, "** ProcessNewOrder Thread %d iTermId %d
newOrderDataPtr: %x \r\n",
                        GetCurrentThreadId(), iTermId, &newOrderDataPtr);
                fclose(fp);
        }


   Term->pClientData[iTermId].trans.newOrderData = *newOrderDataPtr;

   iRc = newOrderDataPtr->retval;

   tpfree((char *)newOrderDataPtr);


  #else             // using global alloc method for tuxedo

   *Term->pClientData[iTermId].newOrderDataPtr = Term-
>pClientData[iTermId].trans.newOrderData;

   ilen = sizeof(NewOrderData);
   olen = &ilen;

        if ( dLog )
        {
                FILE *fp;

                fp = fopen(szTpccLogPath, "ab");
                fprintf(fp, "* ProcessNewOrder Thread %d iTermId %d
NewOrderDataPtr: %x \r\n",
                        GetCurrentThreadId(), iTermId, &Term-
>pClientData[iTermId].newOrderDataPtr);
                fclose(fp);
        }

   if (tpcall("NEWORDER", (char *)Term->pClientData[iTermId].newOrderDataPtr,
ilen,
                (char **)&Term->pClientData[iTermId].newOrderDataPtr, (long
*)olen, TPSIGRSTRT) == -1)
        {
        iRc = tperrno;
                {
                        FILE *fp;

                        fp = fopen(szTpccLogPath, "ab");
                        fprintf(fp, "* ProcessNewOrder Thread %d iTermId %d
tpcall tperrno %d \r\n",
                                GetCurrentThreadId(), iTermId, iRc);
                        fclose(fp);
                }
                sprintf(buf, "Neworder tpcall %d", iRc);
        ErrorMessage(pECB, ERR_TPCALL_BAD, ERR_TYPE_TUXEDO, buf,
iTermId, iSyncId);
//              tpfree((char *)newOrderDataPtr);
        return;
   }

        if ( dLog )
        {
                FILE *fp;

                fp = fopen(szTpccLogPath, "ab");
                fprintf(fp, "** ProcessNewOrder Thread %d iTermId %d
NewOrderDataPtr: %x \r\n",
                        GetCurrentThreadId(), iTermId, &Term-
>pClientData[iTermId].newOrderDataPtr);
                fclose(fp);
        }
```

```
    Term->pClientData[iTermId].trans.newOrderData = *Term-
>pClientData[iTermId].newOrderDataPtr;

    iRc = Term->pClientData[iTermId].newOrderDataPtr->retval;

  #endif
#endif

          if(bLog){
                    FILE *fp1;
                    fp1 = fopen(szTpccLogPath, "ab");
                    fprintf(fp1, ">> ProcessNewOrder iRc = %d \r\n", iRc);
                    fclose(fp1);
          }

          if ( iRc < 0)
   {
          ErrorMessage(pECB, ERR_NEW_ORDER_NOT_PROCESSED,
ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
          return;
   }



          if (iRc == 0)
          WriteZString(pECB, MakeNewOrderForm(iTermId, iSyncId, FALSE,
FALSE, "Item number is not valid"));
          else

          WriteZString(pECB, MakeNewOrderForm(iTermId, iSyncId, FALSE,
TRUE, "transaction committed"));

          return;

}


/* FUNCTION: void ProcessPaymentForm(EXTENSION_CONTROL_BLOCK
*pECB, int iTermId,
 *                          int iSyncId)
 *
 * PURPOSE: This function gets and validates the input data from the payment
 *          form filling in the required input variables. It then calls the
 *          SQLPayment transaction, constructs the output form and writes it
 *          back to client browser.
 *
 * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB   passed in structure
pointer
 *                                   from inetsrv.
 *      int     iTermId                  client browser terminal id
 *      int     iSyncId                  client browser sync id
 *
 * RETURNS:     None
 *
 * COMMENTS:    None
 *
 */

static void ProcessPaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId)
{
#ifdef TOPEND
     int                 i,iRc;
     tp_dif_structs_t       *tclient_dif;
     int msglen;
#else // assume tuxedo at this point
          int                 iRc, iError;
          long           ilen, *olen;
//      PECBINFO       pEcbInfo;
          char buf[40];
#endif

#ifdef LOCAL_ALLOC
          PaymentData *paymentDataPtr;
#endif


#ifdef TUX
    if((iRc = ThrTpInit(pECB)) <0)
    {
          // This is bad
                {
                          FILE *fp;

                          fp = fopen(szTpccLogPath, "ab");
                          fprintf(fp, "* ProcessPayment Thread %d iTermId %d
Failed ThrTpInit \r\n",
```

```
                                        GetCurrentThreadId(), iTermId );
                                fclose(fp);
                        }
                        sprintf(buf, "Payment ThrTpInit < 0 >> Thread %d",
GetCurrentThreadId());
            ErrorMessage(pECB, ERR_TPINIT_BAD, ERR_TYPE_TUXEDO, buf, 0,
0);
            return;//change 2003/7/11
    }
#endif


    memset(&Term->pClientData[iTermId].trans.paymentData, 0,
sizeof(PaymentData));
    Term->pClientData[iTermId].trans.paymentData.payin.w_id = Term-
>pClientData[iTermId].w_id;

    if ( (iError=GetPaymentData(pECB->lpszQueryString, &Term-
>pClientData[iTermId].trans.paymentData)) !=ERR_SUCCESS )
    {
            ErrorMessage(pECB, iError, ERR_TYPE_WEBDLL, NULL, iTermId,
iSyncId);
            return;
    }

   //  TPCC_LOG("after get payment data");
#ifdef TOPEND

    tclient_dif = (tp_dif_structs_t *)Term->pClientData[iTermId].tp_structadr;

    // Input topend information for call to process payment form

    tclient_dif->info->tp_user_dialogue_id = iTermId;

    if (bGeneric)
            space_fill(tclient_dif->service->tp_function_name,
"all_txn",TP_FUNC_NAME_LEN);
    else
            space_fill(tclient_dif->service->tp_function_name,
"payment",TP_FUNC_NAME_LEN);

    space_fill(tclient_dif->service->tp_product_name, "tpcc",
TP_PROD_NAME_LEN);
    space_fill(tclient_dif->input_format->tp_format_name, "",
TP_FMT_NAME_LEN);
    space_fill(tclient_dif->input_format->tp_format_language, "", TP_LANG_LEN);
    space_fill(tclient_dif->input_format->tp_format_type, "", TP_FMTTYPE_LEN);
    tclient_dif->service->tp_function_qualifier = 0;
    msglen = sizeof(CLIENTDATA) - TOPEND_STRUCT_OFFSET;

    Term->pClientData[iTermId].type = 'P';

            // tp_system_log_text("ProcessPayment:","before tp_mt_client_send");

    if ((i = tp_mt_client_send(tclient_dif->info, tclient_dif->service,
            tclient_dif->input_format, msglen, (char *)&Term-
>pClientData[iTermId].type)) != TP_OK)
    {

            ErrorMessage(pECB, TOPEND_SEND_ERROR, ERR_TYPE_WEBDLL,
NULL, iTermId, iSyncId);

            if (bLog)
            {
              FILE *fp1;
              fp1 = fopen(szTpccLogPath, "ab");
              fprintf(fp1, "Received error status = %d on payment send fnt\n", i);
              fclose(fp1);
            }
            return;
    }


    msglen = sizeof(CLIENTDATA) - TOPEND_STRUCT_OFFSET;
    // tp_system_log_text("ProcessPayment:","before tp_mt_client_receive");
    if ((i = tp_mt_client_receive(tclient_dif->info, wait_time,
            tclient_dif->output_format, tclient_dif->service,
            tclient_dif->location, &msglen, (char *)&Term-
>pClientData[iTermId].type)) != TP_OK)
    {
                    //tp_system_log_text("ProcessPayment:","Error on
tp_mt_client_receive");
            if (bLog)
            {
              FILE *fp1;
              fp1 = fopen(szTpccLogPath, "ab");
```

```
            fprintf(fp1, "Received error status = %d from client receive for payment
transaction\n", i);
            fclose(fp1);
          }

          ErrorMessage(pECB, TOPEND_RECEIVE_ERROR,
ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
          return;
     }


     iRc = Term->pClientData[iTermId].retval;


#else  // assume tuxedo running here

 #ifdef LOCAL_ALLOC

          if ((paymentDataPtr = (PaymentData *) tpalloc("CARRAY", NULL,
sizeof(PaymentData))) == NULL)
          {
                    iRc = tperrno;
                    {
                              FILE *fp;

                              fp = fopen(szTpccLogPath, "ab");
                              fprintf(fp, "* ProcessPayment Thread %d iTermId %d
tpalloc tperrno %d \r\n",
                              GetCurrentThreadId(), iTermId, iRc);
                              fclose(fp);
                    }
                    sprintf(buf, "Payment Tpalloc %d", iRc);
                    ErrorMessage(pECB, ERR_TPALLOC_BAD,
ERR_TYPE_TUXEDO, buf, 0, 0);
                    tpfree((char *)paymentDataPtr);
                    return;
          }

    *paymentDataPtr= Term->pClientData[iTermId].trans.paymentData;

     ilen = sizeof(PaymentData);
     olen = &ilen;


          if ( dLog )
          {
                    FILE *fp;

                    fp = fopen(szTpccLogPath, "ab");
                    fprintf(fp, "* ProcessPayment Thread %d iTermId %d
paymentDataPtr: %x \r\n",
                              GetCurrentThreadId(), iTermId, &paymentDataPtr);
                    fclose(fp);
          }


     if (tpcall("OPSTUXSERVER", (char *)paymentDataPtr, ilen, (char
**)&paymentDataPtr, (long *)olen, TPSIGRSTRT) == -1)
          {
        iRc = tperrno;
                    {
                              FILE *fp;

                              fp = fopen(szTpccLogPath, "ab");
                              fprintf(fp, "* ProcessPayment Thread %d iTermId %d
tpcall tperrno %d \r\n",
                                        GetCurrentThreadId(), iTermId, iRc);
                              fclose(fp);
                    }
                    sprintf(buf, "Payment tpcall %d", iRc);
        ErrorMessage(pECB, ERR_TPCALL_BAD, ERR_TYPE_TUXEDO, buf,
iTermId, iSyncId);
        tpfree((char *)paymentDataPtr);

        return;
     }

          if ( dLog )
          {
                    FILE *fp;

                    fp = fopen(szTpccLogPath, "ab");
                    fprintf(fp, "** ProcessPayment Thread %d iTermId %d
paymentDataPtr: %x \r\n",
                              GetCurrentThreadId(), iTermId, &paymentDataPtr);
                    fclose(fp);
          }
```

```
    Term->pClientData[iTermId].trans.paymentData = *paymentDataPtr;

    iRc = paymentDataPtr->retval;

            tpfree((char *)paymentDataPtr);

 #else

    *Term->pClientData[iTermId].paymentDataPtr = Term-
>pClientData[iTermId].trans.paymentData;

    ilen = sizeof(PaymentData);
    olen = &ilen;

            if ( dLog )
            {
                    FILE *fp;

                    fp = fopen(szTpccLogPath, "ab");
                    fprintf(fp, "* ProcessPayment Thread %d iTermId %d
PaymentDataPtr: %x \r\n",
                            GetCurrentThreadId(), iTermId, &Term-
>pClientData[iTermId].paymentDataPtr);
                    fclose(fp);
            }

    if (tpcall("PAYMENT", (char *)Term->pClientData[iTermId].paymentDataPtr,
ilen,
                    (char **)&Term->pClientData[iTermId].paymentDataPtr, (long
*)olen, TPSIGRSTRT) == -1)
            {
            iRc = tperrno;
                    {
                            FILE *fp;

                            fp = fopen(szTpccLogPath, "ab");
                            fprintf(fp, "* ProcessPayment Thread %d iTermId %d
tpcall tperrno %d \r\n",
                                    GetCurrentThreadId(), iTermId, iRc);
                            fclose(fp);
                    }
                    sprintf(buf, "Payment tpcall %d", iRc);
                    ErrorMessage(pECB, ERR_TPCALL_BAD, ERR_TYPE_TUXEDO, buf,
iTermId, iSyncId);
//          tpfree((char *)PaymentDataPtr);
                    return;
            }

            if ( dLog )
            {
                    FILE *fp;

                    fp = fopen(szTpccLogPath, "ab");
                    fprintf(fp, "** ProcessPayment Thread %d iTermId %d
PaymentDataPtr: %x \r\n",
                            GetCurrentThreadId(), iTermId, &Term-
>pClientData[iTermId].paymentDataPtr);
                    fclose(fp);
            }

    Term->pClientData[iTermId].trans.paymentData = *Term-
>pClientData[iTermId].paymentDataPtr;

    iRc = Term->pClientData[iTermId].paymentDataPtr->retval;

  #endif
#endif


    if ( iRc == 0 )
            ErrorMessage(pECB, ERR_PAYMENT_INVALID_CUSTOMER,
ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
    else if ( iRc < 0 )
            ErrorMessage(pECB, ERR_PAYMENT_NOT_PROCESSED,
ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
    else
    {
            WriteZString(pECB, MakePaymentForm(iTermId, iSyncId, FALSE) );
    }

    return;

}
```

```
/* FUNCTION: void ProcessOrderStatusForm(EXTENSION_CONTROL_BLOCK
*pECB, int
 *                              iTermId, int iSyncId)
 *
 * PURPOSE:This function gets and validates the input data from the Order Status
 *      form filling in the required input variables. It then calls the
 *      SQLOrderStatus transaction, constructs the output form and writes it
 *      back to client browser.
 *
 * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB   passed in structure
pointer from
 *                                  inetsrv.
 *          int             iTermId client browser terminal id
 *          int             iSyncId client browser sync id
 *
 * RETURNS:     None
 *
 * COMMENTS:    None
 *
 */
static void ProcessOrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId)
{
#ifdef TOPEND
    int             i,iRc;
    int             iError;
    void space_fill();
    tp_dif_structs_t       *tclient_dif;
    int msglen;
#else  // assume tuxedo
        int             iRc, iError;
        long ilen, *olen;
//     PECBINFO       pEcbInfo;
        char buf[40];
#endif

#ifdef LOCAL_ALLOC
        OrderStatusData *orderStatusDataPtr;
#endif

#ifdef TUX

    if((iRc = ThrTpInit(pECB)) <0)
    {
        // This is bad
            {
                FILE *fp;

                fp = fopen(szTpccLogPath, "ab");
                fprintf(fp, "* ProcessOrderStatus Thread %d iTermId %d
Failed ThrTpInit \r\n",
                        GetCurrentThreadId(), iTermId );
                fclose(fp);
            }
            sprintf(buf, "OrderStatus ThrTpInit < 0 >> Thread %d",
GetCurrentThreadId());
        ErrorMessage(pECB, ERR_TPINIT_BAD, ERR_TYPE_TUXEDO, buf, 0,
0);
        return;//change 2003/7/11
    }

#endif

    memset(&Term->pClientData[iTermId].trans.orderStatusData, 0,
sizeof(OrderStatusData));
    Term->pClientData[iTermId].trans.orderStatusData.ordin.w_id = Term-
>pClientData[iTermId].w_id;
    if ( (iError=GetOrderStatusData(pECB->lpszQueryString,
          &Term->pClientData[iTermId].trans.orderStatusData)) !=
ERR_SUCCESS )
    {
        ErrorMessage(pECB, iError, ERR_TYPE_WEBDLL, NULL, iTermId,
iSyncId);
        return;
    }

#ifdef TOPEND

    tclient_dif = (tp_dif_structs_t *)Term->pClientData[iTermId].tp_structadr;

    /* Initialize Top End for tpcc transactions */
    tclient_dif->info->tp_user_dialogue_id = iTermId;

    if (bGeneric)
        space_fill(tclient_dif->service->tp_function_name,
"all_txn",TP_FUNC_NAME_LEN);
```

```
    else
            space_fill(tclient_dif->service->tp_function_name,
"ordstat",TP_FUNC_NAME_LEN);

    space_fill(tclient_dif->service->tp_product_name, "tpcc",
TP_PROD_NAME_LEN);
    space_fill(tclient_dif->input_format->tp_format_name, "",
TP_FMT_NAME_LEN);
    space_fill(tclient_dif->input_format->tp_format_language, "", TP_LANG_LEN);
    space_fill(tclient_dif->input_format->tp_format_type, "", TP_FMTTYPE_LEN);
    tclient_dif->service->tp_function_qualifier = 0;
    msglen = sizeof(CLIENTDATA) - TOPEND_STRUCT_OFFSET;

    Term->pClientData[iTermId].type = 'O';

    if ((i = tp_mt_client_send(tclient_dif->info, tclient_dif->service,
            tclient_dif->input_format, msglen, (char *)&Term-
>pClientData[iTermId].type)) != TP_OK)
    {

        if (bLog)
        {
          FILE *fp1;
          fp1 = fopen(szTpccLogPath, "ab");
          fprintf(fp1, "Received error status = %d on client send for order status
transaction\n", i);
          fclose(fp1);
        }

        ErrorMessage(pECB, TOPEND_SEND_ERROR, ERR_TYPE_WEBDLL,
NULL, iTermId, iSyncId);
        return;
    }

    msglen = sizeof(CLIENTDATA) - TOPEND_STRUCT_OFFSET;

    if ((i = tp_mt_client_receive(tclient_dif->info, wait_time, tclient_dif-
>output_format, tclient_dif->service, tclient_dif->location, &msglen, (char *)&Term-
>pClientData[iTermId].type)) != TP_OK)
    {
        if (bLog)
        {
          FILE *fp1;
          fp1 = fopen(szTpccLogPath, "ab");
          fprintf(fp1, "Received error status = %d from client receive for order status
transaction\n", i);
          fclose(fp1);
        }
        fprintf(stderr, "Received error status = %d from client receive for order
status transaction\n", i);
        ErrorMessage(pECB, TOPEND_RECEIVE_ERROR,
ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
        return;
    }

    iRc = Term->pClientData[iTermId].retval;


#else

 #ifdef LOCAL_ALLOC


        if ((orderStatusDataPtr = (OrderStatusData *) tpalloc("CARRAY", NULL,
sizeof(OrderStatusData))) == NULL)
        {
                iRc = tperrno;
                {
                        FILE *fp;

                        fp = fopen(szTpccLogPath, "ab");
                        fprintf(fp, "* ProcessOrderStatus Thread %d iTermId %d
tpalloc tperrno %d \r\n",
                        GetCurrentThreadId(), iTermId, iRc);
                        fclose(fp);
                }
                sprintf(buf, "OrderStatus Tpcalloc %d", iRc);
                ErrorMessage(NULL, ERR_TPALLOC_BAD,
ERR_TYPE_TUXEDO, buf, 0, 0);
                tpfree((char *)orderStatusDataPtr);
                return;
        }

    *orderStatusDataPtr = Term->pClientData[iTermId].trans.orderStatusData;

    ilen = sizeof(OrderStatusData);
```

```
    olen = &ilen;

            if ( dLog )
            {
                    FILE *fp;

                    fp = fopen(szTpccLogPath, "ab");
                    fprintf(fp, "* ProcessOrderStatus Thread %d iTermId %d
orderStatusDataPtr: %x \r\n",
                            GetCurrentThreadId(), iTermId, &orderStatusDataPtr);
                    fclose(fp);
            }

    if (tpcall("OPSTUXSERVER", (char *)orderStatusDataPtr, ilen, (char
**)&orderStatusDataPtr, (long *)olen, TPSIGRSTRT) == -1)
            {
        iRc = tperrno;
                    {
                            FILE *fp;

                            fp = fopen(szTpccLogPath, "ab");
                            fprintf(fp, "* ProcessOrderStatus Thread %d iTermId %d
tpcall tperrno %d \r\n",
                                    GetCurrentThreadId(), iTermId, iRc);
                            fclose(fp);
                    }
                    sprintf(buf, "OrderStatus tpcall %d", iRc);
        ErrorMessage(pECB, ERR_TPCALL_BAD, ERR_TYPE_TUXEDO, buf,
iTermId, iSyncId);
        tpfree((char *)orderStatusDataPtr);

        return;
    }

            if ( dLog )
            {
                    FILE *fp;

                    fp = fopen(szTpccLogPath, "ab");
                    fprintf(fp, "** ProcessOrderStatus Thread %d iTermId %d
orderStatusDataPtr: %x \r\n",
                            GetCurrentThreadId(), iTermId, &orderStatusDataPtr);
                    fclose(fp);
            }

    Term->pClientData[iTermId].trans.orderStatusData = *orderStatusDataPtr;

    iRc = orderStatusDataPtr->retval;

            tpfree((char *)orderStatusDataPtr);



 #else

    *Term->pClientData[iTermId].orderStatusDataPtr = Term-
>pClientData[iTermId].trans.orderStatusData;

    ilen = sizeof(OrderStatusData);
    olen = &ilen;

            if ( dLog )
            {
                    FILE *fp;

                    fp = fopen(szTpccLogPath, "ab");
                    fprintf(fp, "* ProcessOrderStatus Thread %d iTermId %d
OrderStatusDataPtr: %x \r\n",
                            GetCurrentThreadId(), iTermId, &Term-
>pClientData[iTermId].orderStatusDataPtr);
                    fclose(fp);
            }

    if (tpcall("ORDERSTATUS", (char *)Term-
>pClientData[iTermId].orderStatusDataPtr, ilen,
                    (char **)&Term->pClientData[iTermId].orderStatusDataPtr, (long
*)olen, TPSIGRSTRT) == -1)
            {
            iRc = tperrno;
                    {
                            FILE *fp;

                            fp = fopen(szTpccLogPath, "ab");
                            fprintf(fp, "* ProcessOrderStatus Thread %d iTermId %d
tpcall tperrno %d \r\n",
```

```
                                GetCurrentThreadId(), iTermId, iRc);
                        fclose(fp);
                    }
                    sprintf(buf, "OrderStatus tpcall %d", iRc);
        ErrorMessage(pECB, ERR_TPCALL_BAD, ERR_TYPE_TUXEDO, buf,
iTermId, iSyncId);
//      tpfree((char *)Term->pClientData[iTermId].orderStatusDataPtr);
        return;
    }

        if ( dLog )
        {
            FILE *fp;

            fp = fopen(szTpccLogPath, "ab");
            fprintf(fp, "** ProcessOrderStatus Thread %d iTermId %d
OrderStatusDataPtr: %x \r\n",
                        GetCurrentThreadId(), iTermId, &Term-
>pClientData[iTermId].orderStatusDataPtr);
            fclose(fp);
        }

    Term->pClientData[iTermId].trans.orderStatusData = *Term-
>pClientData[iTermId].orderStatusDataPtr;

    iRc = Term->pClientData[iTermId].orderStatusDataPtr->retval;

  #endif
#endif

    if ( iRc == 0 )
        ErrorMessage(pECB, ERR_NOSUCH_CUSTOMER,
ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
     else if ( iRc < 0 )
        ErrorMessage(pECB, ERR_ORDER_STATUS_NOT_PROCESSED,
ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
     else
        WriteZString(pECB, MakeOrderStatusForm(iTermId, iSyncId, FALSE) );

     return;

}
```

```
/* FUNCTION: void ProcessDeliveryForm(EXTENSION_CONTROL_BLOCK
*pECB,
 *                          int iTermId, int iSyncId)
 *
 * PURPOSE:This function gets and validates the input data from delivery form
 * filling in the required input variables. It then calls the PostDeliveryInfo
 * Api, The client is then informed that the transaction has been posted.
 *
 * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB   passed in structure
pointer from
 *                              inetsrv.
 *          int             iTermId client browser terminal id
 *          int             iSyncId clinet browser sync id
 *
 * RETURNS:     None
 *
 * COMMENTS:    None
 *
 */

static void ProcessDeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId)
{

#ifdef TOPEND
        char    szTmp[26];
        int     i;
        void space_fill();
        tp_dif_structs_t *tclient_dif;
        int msglen;
#else
        int         iRc;
//      int         iError;
        char    szTmp[26];
        long ilen, *olen;
//      int         tmp;
        char buf[40];
#endif

#ifdef LOCAL_ALLOC
        DeliveryData *deliveryDataPtr;
#endif
```

```
#ifdef TUX


   if((iRc = ThrTpInit(pECB)) <0)
    {
            // This is bad
                    {
                            FILE *fp;

                            fp = fopen(szTpccLogPath, "ab");
                            fprintf(fp, "* ProcessDelivery Thread %d iTermId %d
Failed ThrTpInit \r\n",
                                    GetCurrentThreadId(), iTermId );
                            fclose(fp);
                    }
                    sprintf(buf, "Delivery ThrTpInit < 0 >> Thread %d",
GetCurrentThreadId());
            ErrorMessage(pECB, ERR_TPINIT_BAD, ERR_TYPE_TUXEDO, buf, 0,
0);
            return;//change 2003/7/11
    }

#endif

    memset(&Term->pClientData[iTermId].trans.deliveryData, 0,
sizeof(DeliveryData));
    Term->pClientData[iTermId].trans.deliveryData.delin.w_id = Term-
>pClientData[iTermId].w_id;


    if ( !GetKeyValue(pECB->lpszQueryString, "OCD*", szTmp, sizeof(szTmp)) )

    {
            ErrorMessage(pECB,
ERR_DELIVERY_MISSING_OCD_KEY,ERR_TYPE_WEBDLL, NULL, iTermId,
iSyncId);
            return;
    }

     if ( !IsNumeric (szTmp) )
    {
            ErrorMessage(pECB,
ERR_DELIVERY_CARRIER_INVALID,ERR_TYPE_WEBDLL, NULL, iTermId,
iSyncId);
            return;
    }

    Term->pClientData[iTermId].trans.deliveryData.delin.o_carrier_id = atoi(szTmp);
    if ( Term->pClientData[iTermId].trans.deliveryData.delin.o_carrier_id > 10 ||
            Term->pClientData[iTermId].trans.deliveryData.delin.o_carrier_id < 1 )
    {
    ErrorMessage(pECB,
ERR_DELIVERY_CARRIER_ID_RANGE,ERR_TYPE_WEBDLL, NULL,
iTermId, iSyncId);
    return;
    }

    //post delivery info

#ifdef TOPEND

    tclient_dif = (tp_dif_structs_t *)Term->pClientData[iTermId].tp_structadr;

    /* Initialize Top End for tpcc transactions */
    tclient_dif->info->tp_user_dialogue_id = iTermId;
    tclient_dif->info->tp_flags = TP_NO_RESPONSE;
    space_fill(tclient_dif->service->tp_function_name,
"del_bat",TP_FUNC_NAME_LEN);
    space_fill(tclient_dif->service->tp_product_name, "tpcc",
TP_PROD_NAME_LEN);
    space_fill(tclient_dif->input_format->tp_format_name, "",
TP_FMT_NAME_LEN);
    space_fill(tclient_dif->input_format->tp_format_language, "", TP_LANG_LEN);
    space_fill(tclient_dif->input_format->tp_format_type, "", TP_FMTTYPE_LEN);
    tclient_dif->service->tp_function_qualifier = 0;
    msglen = sizeof(CLIENTDATA) - TOPEND_STRUCT_OFFSET;

    Term->pClientData[iTermId].type = 'D';

    /* get the time the entry is put on the queue */
#ifdef WIN32
    _ftime(&timebuffer);
#else
    ftime(&timebuffer);
#endif
```

```
    Term->pClientData[iTermId].trans.deliveryData.delin.qtime = timebuffer.time +
timebuffer.millitm/1000.0;

    if ((i = tp_mt_client_send(tclient_dif->info, tclient_dif->service,
            tclient_dif->input_format, msglen, (char *)&Term-
>pClientData[iTermId].type)) != TP_OK)
    {

            if (bLog)
            {
              FILE *fp1;
              fp1 = fopen(szTpccLogPath, "ab");
              fprintf(fp1, "Received error status = %d on client send for delivery
transaction\n", i);
              fclose(fp1);
            }

            ErrorMessage(pECB, TOPEND_SEND_ERROR, ERR_TYPE_WEBDLL,
NULL, iTermId, iSyncId);
            return;
    }

    /* msglen might need to be 0, as not receiving anything put ack from server */
    msglen = sizeof(CLIENTDATA) - TOPEND_STRUCT_OFFSET;

    if ((i = tp_mt_client_receive(tclient_dif->info, wait_time, tclient_dif-
>output_format, tclient_dif->service, tclient_dif->location, &msglen, (char *)&Term-
>pClientData[iTermId].type)) != TP_OK)
    {
            if (bLog)
            {
              FILE *fp1;
              fp1 = fopen(szTpccLogPath, "ab");
              fprintf(fp1, "Received error status = %d from client receive for order status
transaction\n", i);
              fclose(fp1);
            }

            ErrorMessage(pECB, TOPEND_RECEIVE_ERROR,
ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
            return;
    }
    /* reset flags for next txn */
    tclient_dif->info->tp_flags = TP_NOFLAGS;
#else

 #ifdef LOCAL_ALLOC

          if ((deliveryDataPtr = (DeliveryData *) tpalloc("CARRAY", NULL,
sizeof(DeliveryData))) == NULL)
          {
                    iRc = tperrno;
                    {
                              FILE *fp;

                              fp = fopen(szTpccLogPath, "ab");
                              fprintf(fp, "* ProcessDelivery Thread %d iTermId %d
tpalloc tperrno %d \r\n",
                              GetCurrentThreadId(), iTermId, iRc);
                              fclose(fp);
                    }
                    sprintf(buf, "Delivery Tpcalloc %d", iRc);
                    ErrorMessage(NULL, ERR_TPALLOC_BAD,
ERR_TYPE_TUXEDO, buf, 0, 0);
                    tpfree((char *)deliveryDataPtr);

                    return;
          }


          *deliveryDataPtr = Term->pClientData[iTermId].trans.deliveryData;

    ilen = sizeof(DeliveryData);
    olen = &ilen;

           if ( dLog )
           {
                    FILE *fp;

                    fp = fopen(szTpccLogPath, "ab");
                    fprintf(fp, "* ProcessDelivery Thread %d iTermId %d
deliveryDataPtr: %x \r\n",
                              GetCurrentThreadId(), iTermId, &deliveryDataPtr);
                    fclose(fp);
           }
```

```
    /* get the time the entry is put on the queue */
    ftime(&timebuffer);
    deliveryDataPtr->delin.qtime = timebuffer.time + timebuffer.millitm/1000.0;


   if (tpacall("OPSTUXSERVER2", (char *)deliveryDataPtr, ilen,
TPNOREPLY|TPSIGRSTRT) != 0) {
      iRc = tperrno;
                  {
                        FILE *fp;

                        fp = fopen(szTpccLogPath, "ab");
                        fprintf(fp, "* ProcessDelivery Thread %d iTermId %d
tpcall tperrno %d \r\n",
                              GetCurrentThreadId(), iTermId, iRc);
                        fclose(fp);
                  }
                  sprintf(buf, "Delivery tpcall %d", iRc);
      ErrorMessage(pECB, ERR_TPCALL_BAD, ERR_TYPE_TUXEDO, buf,
iTermId, iSyncId);
      tpfree((char *)deliveryDataPtr);

      return;
   }



         if ( dLog )
         {
               FILE *fp;

               fp = fopen(szTpccLogPath, "ab");
               fprintf(fp, "** ProcessDelivery Thread %d iTermId %d
deliveryDataPtr: %x \r\n",
                     GetCurrentThreadId(), iTermId, &deliveryDataPtr);
               fclose(fp);
         }

         tpfree((char *)deliveryDataPtr);
```

```
 #else
   *Term->pClientData[iTermId].deliveryDataPtr = Term-
>pClientData[iTermId].trans.deliveryData;

   ilen = sizeof(DeliveryData);
   olen = &ilen;

         if ( dLog )
         {
               FILE *fp;

               fp = fopen(szTpccLogPath, "ab");
               fprintf(fp, "* ProcessDelivery Thread %d iTermId %d
DeliveryDataPtr: %x \r\n",
                     GetCurrentThreadId(), iTermId, &Term-
>pClientData[iTermId].deliveryDataPtr);
               fclose(fp);
         }

   /* get the time the entry is put on the queue */
   ftime(&timebuffer);
   Term->pClientData[iTermId].deliveryDataPtr->delin.qtime = timebuffer.time +
timebuffer.millitm/1000.0;

   if (tpcall("DELIVERY", (char *)Term->pClientData[iTermId].deliveryDataPtr,
ilen,
         (char **)&Term->pClientData[iTermId].deliveryDataPtr, (long
*)olen, TPSIGRSTRT) == -1) {
      iRc = tperrno;
               {
                     FILE *fp;

                     fp = fopen(szTpccLogPath, "ab");
                     fprintf(fp, "* ProcessDelivery Thread %d iTermId %d
tpcall tperrno %d \r\n",
                           GetCurrentThreadId(), iTermId, iRc);
                     fclose(fp);
               }
               sprintf(buf, "Delivery tpcall %d", iRc);
      ErrorMessage(pECB, ERR_TPCALL_BAD, ERR_TYPE_TUXEDO, buf,
iTermId, iSyncId);
//    tpfree((char *)Term->pClientData[iTermId].deliveryDataPtr);
      return;
```

```
    }

        if ( dLog )
        {
                FILE *fp;

                fp = fopen(szTpccLogPath, "ab");
                fprintf(fp, "** ProcessDelivery Thread %d iTermId %d
DeliveryDataPtr: %x \r\n",
                        GetCurrentThreadId(), iTermId, &Term-
>pClientData[iTermId].deliveryDataPtr);
                fclose(fp);
        }

   iRc = Term->pClientData[iTermId].deliveryDataPtr->retval;


 #endif
#endif

    WriteZString(pECB, MakeDeliveryForm(iTermId, iSyncId, FALSE, "Delivery
has been queued."));

    return;
}


/* FUNCTION: void ProcessStockLevelForm(EXTENSION_CONTROL_BLOCK
*pECB, int
 *                      iTermId, int iSyncId)
 *
 * PURPOSE: This function gets and validates the input data from the Stock Level
 * form filling in the required input variables. It then calls the
 * SQLStockLevel transaction, constructs the output form and writes it
 * back to client browser.
 *
 * ARGUMENTS: EXTENSION_CONTROL_BLOCK   *pECB   passed in structure
pointer from
 *                          inetsrv.
 *          int             iTermId client browser terminal id
 *          int             iSyncId client browser sync id
 *
 * RETURNS:    None
 *
 * COMMENTS:    None
 *
 */

static void ProcessStockLevelForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId)
{
#ifdef TOPEND
    char    szTmp[26];
    int             i,iRc;
    int msglen;
    void space_fill();
    tp_dif_structs_t        *tclient_dif;
#else
        int         iRc;
//      int         iError;
        char        szTmp[26];
//      BOOL        bRc;
        long ilen, *olen;
//      PECBINFO        pEcbInfo;
        char buf[40];
#endif

#ifdef LOCAL_ALLOC
        StockLevelData *stockLevelDataPtr;
#endif


#ifdef TUX
   if((iRc = ThrTpInit(pECB)) <0)
   {
        // This is bad
                {
                        FILE *fp;

                        fp = fopen(szTpccLogPath, "ab");
                        fprintf(fp, "* ProcessStockLevel Thread %d iTermId %d
Failed ThrTpInit \r\n",
                                GetCurrentThreadId(), iTermId );
                        fclose(fp);
                }
                sprintf(buf, "StockLevel ThrTpInit < 0 >> Thread %d",
```

```
GetCurrentThreadId());
            ErrorMessage(pECB, ERR_TPINIT_BAD, ERR_TYPE_TUXEDO, buf, 0,
0);
            return;//change 2003/7/11
        }
#endif

    memset(&Term->pClientData[iTermId].trans.stocklevelData, 0,
sizeof(StockLevelData));
    Term->pClientData[iTermId].trans.stocklevelData.stoin.w_id = Term-
>pClientData[iTermId].w_id;
    Term->pClientData[iTermId].trans.stocklevelData.stoin.d_id = Term-
>pClientData[iTermId].d_id;
    if ( !GetKeyValue(pECB->lpszQueryString, "TT*", szTmp, sizeof(szTmp)) )
    {
            ErrorMessage(pECB,
ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY,ERR_TYPE_WEBDLL,
NULL, iTermId, iSyncId);
            return;
    }
    if ( !IsNumeric(szTmp) )
    {
            ErrorMessage(pECB,
ERR_STOCKLEVEL_THRESHOLD_INVALID,ERR_TYPE_WEBDLL, NULL,
iTermId, iSyncId);
            return;
    }
    Term->pClientData[iTermId].trans.stocklevelData.stoin.threshold = atoi(szTmp);
    if ( Term->pClientData[iTermId].trans.stocklevelData.stoin.threshold >= 100 ||
             Term->pClientData[iTermId].trans.stocklevelData.stoin.threshold <= 0 )
    {
            ErrorMessage(pECB,
ERR_STOCKLEVEL_THRESHOLD_RANGE,ERR_TYPE_WEBDLL, NULL,
iTermId, iSyncId);
            return;
    }

#ifdef TOPEND

    tclient_dif = (tp_dif_structs_t *)Term->pClientData[iTermId].tp_structadr;

    /* Initialize Top End for tpcc transactions */
    tclient_dif->info->tp_user_dialogue_id = iTermId;
    if (bGeneric)
            space_fill(tclient_dif->service->tp_function_name,
"all_txn",TP_FUNC_NAME_LEN);
    else
            space_fill(tclient_dif->service->tp_function_name,
"stklevel",TP_FUNC_NAME_LEN);

    space_fill(tclient_dif->service->tp_product_name, "tpcc",
TP_PROD_NAME_LEN);
    space_fill(tclient_dif->input_format->tp_format_name, "",
TP_FMT_NAME_LEN);
    space_fill(tclient_dif->input_format->tp_format_language, "", TP_LANG_LEN);
    space_fill(tclient_dif->input_format->tp_format_type, "", TP_FMTTYPE_LEN);
    tclient_dif->service->tp_function_qualifier = 0;
    msglen = sizeof(CLIENTDATA) - TOPEND_STRUCT_OFFSET;

    Term->pClientData[iTermId].type = 'S';

      if ((i = tp_mt_client_send(tclient_dif->info, tclient_dif->service, tclient_dif-
>input_format, msglen, (char *)&Term->pClientData[iTermId].type)) != TP_OK)
    {

        if (bLog)
        {
          FILE *fp1;
          fp1 = fopen(szTpccLogPath, "ab");
          fprintf(fp1, "Received error status = %d from client send for stock level
transaction\n", i);
          fclose(fp1);
        }

        ErrorMessage(pECB, TOPEND_SEND_ERROR, ERR_TYPE_WEBDLL,
NULL, iTermId, iSyncId);
        return;
    }

    msglen = sizeof(CLIENTDATA) - TOPEND_STRUCT_OFFSET;

    if ((i = tp_mt_client_receive(tclient_dif->info, wait_time, tclient_dif-
>output_format, tclient_dif->service, tclient_dif->location, &msglen, (char *)&Term-
>pClientData[iTermId].type)) != TP_OK)
    {
```

```
        if (bLog)
        {
          FILE *fp1;
          fp1 = fopen(szTpccLogPath, "ab");
          fprintf(fp1, "Received error status = %d from client receive for stock level
transaction\n", i);
          fclose(fp1);
        }
        ErrorMessage(pECB, TOPEND_RECEIVE_ERROR,
ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
        return;
    }

    iRc = Term->pClientData[iTermId].retval;

#else

 #ifdef LOCAL_ALLOC

        if ((stockLevelDataPtr = (StockLevelData *) tpalloc("CARRAY", NULL,
sizeof(StockLevelData))) == NULL)
        {
                iRc = tperrno;
                {
                        FILE *fp;

                        fp = fopen(szTpccLogPath, "ab");
                        fprintf(fp, "* ProcessStockLevel Thread %d iTermId %d
tpalloc tperrno %d \r\n",
                        GetCurrentThreadId(), iTermId, iRc);
                        fclose(fp);
                }
                sprintf(buf, "Tpcalloc %d", iRc);
                ErrorMessage(NULL, ERR_TPALLOC_BAD,
ERR_TYPE_TUXEDO, buf, 0, 0);
                tpfree((char *)stockLevelDataPtr);

                return;
        }

    *stockLevelDataPtr = Term->pClientData[iTermId].trans.stocklevelData;

    ilen = sizeof(StockLevelData);
    olen = &ilen;

    if (tpcall("OPSTUXSERVER", (char *) stockLevelDataPtr, ilen, (char **)
&stockLevelDataPtr, (long *) olen, TPSIGRSTRT) == -1) {
        iRc = tperrno;
                {
                        FILE *fp;

                        fp = fopen(szTpccLogPath, "ab");
                        fprintf(fp, "* ProcessStockLevel Thread %d iTermId %d
tpcall tperrno %d \r\n",
                                GetCurrentThreadId(), iTermId, iRc);
                        fclose(fp);
                }
                sprintf(buf, "StockLevel tpcall %d", iRc);
        ErrorMessage(pECB, ERR_TPCALL_BAD, ERR_TYPE_TUXEDO, buf,
iTermId, iSyncId);
                tpfree((char *)stockLevelDataPtr);

        return;
    }

        if ( dLog )
        {
                FILE *fp;

                fp = fopen(szTpccLogPath, "ab");
                fprintf(fp, "** ProcessStockLevel Thread %d iTermId %d
stockLevelDataPtr: %x \r\n",
                        GetCurrentThreadId(), iTermId, &stockLevelDataPtr);
                fclose(fp);
        }

    Term->pClientData[iTermId].trans.stocklevelData = *stockLevelDataPtr;

    iRc = stockLevelDataPtr->retval;

        tpfree((char *)stockLevelDataPtr);


 #else
```

```
   *Term->pClientData[iTermId].stockLevelDataPtr = Term-
>pClientData[iTermId].trans.stocklevelData;

    ilen = sizeof(StockLevelData);
    olen = &ilen;

            if ( dLog )
            {
                    FILE *fp;

                    fp = fopen(szTpccLogPath, "ab");
                    fprintf(fp, "* ProcessStockLevel Thread %d iTermId %d
StockLevelDataPtr: %x \r\n",
                            GetCurrentThreadId(), iTermId, &Term-
>pClientData[iTermId].stockLevelDataPtr);
                    fclose(fp);
            }

    if (tpcall("STOCKLEVEL", (char *)Term-
>pClientData[iTermId].stockLevelDataPtr, ilen,
                    (char **)&Term->pClientData[iTermId].stockLevelDataPtr, (long
*) olen, TPSIGRSTRT) == -1) {
            iRc = tperrno;
                    {
                            FILE *fp;

                            fp = fopen(szTpccLogPath, "ab");
                            fprintf(fp, "* ProcessStockLevel Thread %d iTermId %d
tpcall tperrno %d \r\n",
                                    GetCurrentThreadId(), iTermId, iRc);
                            fclose(fp);
                    }
                    sprintf(buf, "StockLevel tpcall %d", iRc);
            ErrorMessage(pECB, ERR_TPCALL_BAD, ERR_TYPE_TUXEDO, buf,
iTermId, iSyncId);
//              tpfree((char *)StockLevelDataPtr);
            return;
    }

            if ( dLog )
            {
                    FILE *fp;

                    fp = fopen(szTpccLogPath, "ab");
                    fprintf(fp, "** ProcessStockLevel Thread %d iTermId %d
StockLevelDataPtr: %x \r\n",
                            GetCurrentThreadId(), iTermId, &Term-
>pClientData[iTermId].stockLevelDataPtr);
                    fclose(fp);
            }

    Term->pClientData[iTermId].trans.stocklevelData = *Term-
>pClientData[iTermId].stockLevelDataPtr;

    iRc = Term->pClientData[iTermId].stockLevelDataPtr->retval;

  #endif
#endif

    if ( iRc  < 0) {
      ErrorMessage(pECB, ERR_STOCKLEVEL_NOT_PROCESSED,
ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
    }
    else {
     WriteZString(pECB, MakeStockLevelForm(iTermId, iSyncId, FALSE) );
    }

     return;
}


/* FUNCTION: int GetNewOrderData(LPSTR lpszQueryString, NewOrderData
*pNewOrderData)
 *
 * PURPOSE: This function extracts and validates the new order form data from
 *          an http command string.
 *
 * ARGUMENTS: LPSTR     lpszQueryString client browser http command string
 *            NewOrderData  *pNewOrderData pointer to new order data structure
 *
 * RETURNS:     int             error code indicating reason for failure
 *              ERR_SUCCESS     new order input data successfully parsed
 *
 *
 * COMMENTS:    None
 *
```

```
 */
static int GetNewOrderData(LPSTR lpszQueryString, NewOrderData
*pNewOrderData)
{
        char    szTmp[26];
        int             i,j,skipped;
        short   items;

//      BOOL    bCheck;
//      char *  s;

//      s=lpszQueryString;

        if ( !GetKeyValue(lpszQueryString, "DID*", szTmp, sizeof(szTmp)) )
                return ERR_NEWORDER_FORM_MISSING_DID;

        if ( !IsNumeric(szTmp) )
                return ERR_NEWORDER_DISTRICT_INVALID;


        pNewOrderData->newin.d_id = atoi(szTmp);

//      if ( pNewOrderData->newin.d_id > 10 || pNewOrderData->newin.d_id < 1)
//              return ERR_NEWORDER_DISTRICT_INVALID;

        if ( !GetKeyValue(lpszQueryString, "CID*", szTmp, sizeof(szTmp)) )
                return ERR_NEWORDER_CUSTOMER_KEY;

        if ( !IsNumeric(szTmp) )
                return ERR_NEWORDER_CUSTOMER_INVALID;

    pNewOrderData->newin.c_id = atoi(szTmp);

//      if ( pNewOrderData->newin.c_id > 3000 || pNewOrderData->newin.c_id <1)
//              return ERR_NEWORDER_CUSTOMER_INVALID;
    /*pNewOrderData->o_all_local=1;*/

//   bCheck = FALSE;
        skipped = 0;
    for(i=0, items=0; i<15; i++)
    {

        if ( !GetKeyValue(lpszQueryString, szIID[i], szTmp, sizeof(szTmp)) )
                return ERR_NEWORDER_MISSING_IID_KEY;

        if ( szTmp[0] )
        {
                //if blank lines between item ids
//              if ( bCheck )
//                      return
ERR_NEWORDER_ITEM_BLANK_LINES;
                if ( !IsNumeric(szTmp) )
                        return ERR_NEWORDER_ITEMID_INVALID;
                pNewOrderData->newin.ol_i_id[i-skipped] = atoi(szTmp);

                if ( !GetKeyValue(lpszQueryString, szSP[i], szTmp,
sizeof(szTmp)) )
                        return
ERR_NEWORDER_MISSING_SUPPW_KEY;
                if ( !IsNumeric(szTmp) )
                        return ERR_NEWORDER_SUPPW_INVALID;
                pNewOrderData->newin.ol_supply_w_id[i-skipped] =
(int)atoi(szTmp); //2003/07/19
/*              if ( pNewOrderData->o_all_local && pNewOrderData-
>o_ol[i].ol_supply_w_id != pNewOrderData->w_id )
                  pNewOrderData->o_all_local = 0; */


                if ( !GetKeyValue(lpszQueryString, szQty[i], szTmp,
sizeof(szTmp)) )
                        return
ERR_NEWORDER_MISSING_QTY_KEY;

                if ( !IsNumeric(szTmp) )
                        return ERR_NEWORDER_QTY_INVALID;

                pNewOrderData->old_quantity[i-skipped] = atoi(szTmp);
                pNewOrderData->newin.ol_quantity[i-skipped] =
atoi(szTmp);

                items++;

                if ( pNewOrderData->newin.ol_i_id[i-skipped] >=
1000000 || pNewOrderData->newin.ol_i_id[i-skipped] < 1 )
                        return ERR_NEWORDER_ITEMID_RANGE;
                if ( pNewOrderData->newin.ol_quantity[i-skipped] >= 100
```

```
|| pNewOrderData->newin.ol_quantity[i-skipped] < 1 )
                                        return ERR_NEWORDER_QTY_RANGE;


                                /*for (j = i-1-skipped; j >= 0;  j-- ) //bug?
                                  if (pNewOrderData->newin.ol_i_id[i-skipped] ==
pNewOrderData->newin.ol_i_id[j]){
                                        pNewOrderData->newin.ol_quantity[i-skipped]
+= pNewOrderData->newin.ol_quantity[j];
                                        j = -1;
                                  }
                                */
                        //      skipped = 0;
                        }
                        else
                        {
                                if ( !GetKeyValue(lpszQueryString, szSP[i], szTmp,
sizeof(szTmp)) )
                                        return
ERR_NEWORDER_MISSING_QTY_KEY;

                                if ( szTmp[0] )
                                        return
ERR_NEWORDER_SUPPW_WITHOUT_ITEMID;

                                if ( !GetKeyValue(lpszQueryString, szQty[i], szTmp,
sizeof(szTmp)) )
                                        return
ERR_NEWORDER_MISSING_QTY_KEY;

                                if ( szTmp[0] )
                                        return
ERR_NEWORDER_QTY_WITHOUT_ITEMID;

//                              bCheck = TRUE;
                                skipped ++;
                        }
                }
                  skipped = 0;
                if ( items == 0 )
                        return ERR_NEWORDER_NOITEMS_ENTERED;

        // pNewOrderData->newout.o_ol_cnt = items;
        return ERR_SUCCESS;
}

/* FUNCTION: int GetPaymentData(LPSTR lpszQueryString, PaymentData
*pPaymentData)
 *
 *
 * PURPOSE: This function extracts and validates the payment form data from an
 *          http command string.
 *
 * ARGUMENTS: LPSTR     lpszQueryString client browser http command string
 *          PaymentData       *pPaymentData   ptr to payment data structure
 *
 * RETURNS:   int               error code indicating reason for failure
 *          ERR_SUCCESS       all input data successfully parsed
 *
 * COMMENTS:    None
 *
 */
static int GetPaymentData(LPSTR lpszQueryString, PaymentData *pPaymentData)
{
        char    szTmp[26];
        char    *ptr;
//      char    *s;
        double fTmp;


//      s=lpszQueryString;
        if(!GetKeyValue(lpszQueryString, "DID*", szTmp, sizeof(szTmp)))
                return ERR_PAYMENT_MISSING_DID_KEY;

        if ( !IsNumeric(szTmp) )
                return ERR_PAYMENT_DISTRICT_INVALID;
        pPaymentData->payin.d_id = atoi(szTmp);

//      if ( pPaymentData->payin.d_id > 10 || pPaymentData->payin.d_id <1 )
//              return ERR_PAYMENT_DISTRICT_INVALID;

        if(!GetKeyValue(lpszQueryString, "CID*", szTmp, sizeof(szTmp)))
                return ERR_PAYMENT_MISSING_CID_KEY;

        if ( szTmp[0] && !IsNumeric(szTmp) )
```

```
                return ERR_PAYMENT_CUSTOMER_INVALID;

        pPaymentData->payin.c_id = atoi(szTmp);

//      if ( pPaymentData->payin.c_id > 3000 || pPaymentData->payin.c_id <1 )
//              return ERR_PAYMENT_CUSTOMER_INVALID;

        if (bLog){
                        FILE *fp1;
                        fp1 = fopen(szTpccLogPath, "ab");
                        fprintf(fp1, "cid = %s\n",szTmp);
                        fclose(fp1);
        }

        if ( szTmp[0] == 0 )
        {
                if(!GetKeyValue(lpszQueryString, "CLT*", szTmp,
sizeof(szTmp)))
                        return ERR_PAYMENT_MISSING_CLT;
                _strupr( szTmp );

                strcpy(pPaymentData->payin.c_last, szTmp);
                if ( strlen(pPaymentData->payin.c_last) > 16 )
                        return ERR_PAYMENT_LAST_NAME_TO_LONG;
                pPaymentData->payin.bylastname=1;
        }
        else
        {
                if(!GetKeyValue(lpszQueryString, "CLT*", szTmp,
sizeof(szTmp)))
                        return ERR_PAYMENT_MISSING_CLT_KEY;
                if ( szTmp[0] )
                        return ERR_PAYMENT_CID_AND_CLT;
        }

//      s=lpszQueryString;

        if(!GetKeyValue(lpszQueryString, "CWI*", szTmp, sizeof(szTmp)))
                return ERR_PAYMENT_MISSING_CWI_KEY;

        if ( !IsNumeric(szTmp) )
                return ERR_PAYMENT_CWI_INVALID;

        pPaymentData->payin.c_w_id = atoi(szTmp);

//      if ( pPaymentData->payin.c_w_id > iMaxWareHouses || pPaymentData-
>payin.c_w_id <1)
//              return ERR_PAYMENT_CWI_INVALID;

        if (bLog){
                        FILE *fp1;
                        fp1 = fopen(szTpccLogPath, "ab");
                        fprintf(fp1, "cid = %s\n",szTmp);
                        fclose(fp1);
        }

        if(!GetKeyValue(lpszQueryString, "CDI*", szTmp, sizeof(szTmp)))
                return ERR_PAYMENT_MISSING_CDI_KEY;

        if ( !IsNumeric(szTmp) )
                return ERR_PAYMENT_CDI_INVALID;

        pPaymentData->payin.c_d_id = atoi(szTmp);

//      if ( pPaymentData->payin.c_d_id > 10 || pPaymentData->payin.c_d_id <1)
//              return ERR_PAYMENT_CDI_INVALID;

        if(!GetKeyValue(lpszQueryString, "HAM*", szTmp, sizeof(szTmp)))
                return ERR_PAYMENT_MISSING_HAM_KEY;

        ptr = szTmp;

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

```
                        if ( !*ptr )
                                return ERR_PAYMENT_HAM_INVALID;
                }
                else if ( *ptr < '0' || *ptr > '9' )
                                return ERR_PAYMENT_HAM_INVALID;
                ptr++;
        }

        fTmp = atof(szTmp);
        pPaymentData->payin.h_amount = fTmp * 100;
        if ( pPaymentData->payin.h_amount >= 1000000 || pPaymentData-
>payin.h_amount < 0 )
          return ERR_PAYMENT_HAM_RANGE;

        if (bLog){
                                FILE *fp1;
                                fp1 = fopen(szTpccLogPath, "ab");
                                fprintf(fp1, "cid = %s\n",szTmp);
                                fclose(fp1);
        }
        if (bLog){
                                FILE *fp1;
                                fp1 = fopen(szTpccLogPath, "ab");
                                fprintf(fp1, "ouma = %s\n",lpszQueryString);
                                fclose(fp1);
        }

        return ERR_SUCCESS;
}


/* FUNCTION: int GetOrderStatusData(LPSTR lpszQueryString,
 *                          OrderStatusData *pOrderStatusData)
 *
 * PURPOSE: This function extracts and validates the payment form data from an
 *         http command string.
 *
 * ARGUMENTS:LPSTR             lpszQueryString    client browser http command
 *                                 string
 *         OrderStatusData *pOrderStatusData   pointer to order status
 *                                 data structure
 *
 * RETURNS:     int            error code indicating reason for failure
 *             ERR_SUCCESS     successfully parsed all required input data
 *
 * COMMENTS:    None
 *
 */
static int GetOrderStatusData(LPSTR lpszQueryString, OrderStatusData
*pOrderStatusData)
{
        char    szTmp[26];
//      char    *s;
//      s=lpszQueryString;

        if(!GetKeyValue(lpszQueryString, "DID*", szTmp, sizeof(szTmp)))
                return ERR_ORDERSTATUS_MISSING_DID_KEY;
        if ( !IsNumeric(szTmp) )
                return ERR_ORDERSTATUS_DID_INVALID;
        pOrderStatusData->ordin.d_id = atoi(szTmp);

        if(!GetKeyValue(lpszQueryString, "CID*", szTmp, sizeof(szTmp)))
                return ERR_ORDERSTATUS_MISSING_CID_KEY;

        if ( szTmp[0] == 0 )
        {
                pOrderStatusData->ordin.c_id = 0;
                if(!GetKeyValue(lpszQueryString, "CLT*", szTmp,
sizeof(szTmp)))
                        return ERR_ORDERSTATUS_MISSING_CLT_KEY;
                _strupr( szTmp );
                strcpy(pOrderStatusData->ordin.c_last, szTmp);
                if ( strlen(pOrderStatusData->ordin.c_last) > 16 )
                        return ERR_ORDERSTATUS_CLT_RANGE;
                pOrderStatusData->ordin.bylastname = 1;
        }
        else
        {
                if ( !IsNumeric(szTmp) )
                        return ERR_ORDERSTATUS_CID_INVALID;
                pOrderStatusData->ordin.c_id = atoi(szTmp);
                if(!GetKeyValue(lpszQueryString, "CLT*", szTmp,
sizeof(szTmp)))
                        return ERR_ORDERSTATUS_MISSING_CLT_KEY;
                if ( szTmp[0] )
                        return ERR_ORDERSTATUS_CID_AND_CLT;
```

```
        }

    return ERR_SUCCESS;
}


#ifdef WIN32
/* FUNCTION: BOOL ReadRegistrySettings(void)
 *
 * PURPOSE: This function reads the NT registry for startup parameters. There
 *          parameters are under the TPCC key.
 *
 * ARGUMENTS:   None
 *
 * RETURNS:     None
 *
 * COMMENTS:    This function also sets up required operation variables to
 *              their default value so if registry is not setup the default
 *              values will be used.
 *
 */
static BOOL ReadRegistrySettings(void)
{
    HKEY      hKey;
    DWORD     size;
    DWORD     type;
    char      szTmp[256];

    bLog      = FALSE;
    dLog      = FALSE;
    bGeneric  = FALSE;
    iMaxWareHouses    = 625;
    iThreads          = 5;
    iDelayMs          = 100;
    iDeadlockRetry    = (short)3;
    strcpy(szTpccLogPath, "tpcclog.");

    if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\ORACLE\\tpcc", 0, KEY_READ,&hKey) != ERROR_SUCCESS )
            return TRUE;
    size = sizeof(szTmp);

    if ( RegQueryValueEx(hKey, "PATH", 0, &type, szTmp, &size) ==
ERROR_SUCCESS )
    {
            strcpy(szTpccLogPath, szTmp);
            strcat(szTpccLogPath, "tpcclog.");
            strcpy(szErrorLogPath, szTmp);
            strcat(szErrorLogPath, "tpccerr.");
    }

    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "LOG", 0, &type, szTmp, &size) ==
ERROR_SUCCESS )
    {
            if ( !stricmp(szTmp, "ON") )
            bLog = TRUE;
    }

    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "DEBUG", 0, &type, szTmp, &size) ==
ERROR_SUCCESS )
    {
             if ( !stricmp(szTmp, "ON") )
             dLog = TRUE;
    }

    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "MaximumWarehouses", 0, &type, szTmp, &size)
== ERROR_SUCCESS )
    {
             iMaxWareHouses = atoi(szTmp);
             if ( iMaxWareHouses == 0 )
                      iMaxWareHouses = 500;
    }

    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "NumberOfDeliveryThreads", 0, &type, szTmp,
&size) == ERROR_SUCCESS )
            iThreads = atoi(szTmp);
    if ( !iThreads )
            iThreads = 5;

    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "BackoffDelay", 0, &type, szTmp, &size) ==
ERROR_SUCCESS )
```

```
        iDelayMs = atoi(szTmp);
    if ( !iDelayMs )
        iDelayMs = 100;

    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "DeadlockRetry", 0, &type, szTmp, &size) ==
ERROR_SUCCESS )
        iDeadlockRetry = (short)atoi(szTmp);
    if ( !iDeadlockRetry )
        iDeadlockRetry = (short)3;

    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "MaxConnections", 0, &type, szTmp, &size) ==
ERROR_SUCCESS )
        iMaxConnections = (short)atoi(szTmp);
    if ( !iMaxConnections )
        iMaxConnections = (short)25;

    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "GenSrvc", 0, &type, szTmp, &size) ==
ERROR_SUCCESS )
    {
        if ( !stricmp(szTmp, "ON") )
        bGeneric = TRUE;
    }

    RegCloseKey(hKey);
    return FALSE;
}
#endif

/* FUNCTION: BOOL IsNumeric(char *ptr)
 *
 * PURPOSE: This function determines if a string is numeric. It fails if any
 *      characters other than numeric and null terminator are present.
 *
 * ARGUMENTS:   char    *ptr    pointer to string to check.
 *
 * RETURNS:     BOOL    FALSE   if string is not all numeric
 *          TRUE    if string contains all numeric characters i.e. '0' - '9'
 *
 * COMMENTS:    None
 *
 */

static BOOL IsNumeric( char *ptr)
{
    if ( *ptr == 0 )
        return FALSE;
    while( *ptr && isdigit(*ptr) )
        ptr++;
    return ( !*ptr );
}

/* FUNCTION: void FormatHTMLString(char *szBuff, int iLen, char *szStr)
 *
 * PURPOSE:  This function Handles translation of HTML specific character field
 *      data when an HTML output form is generated.
 *
 * ARGUMENTS:   char    *szBuff     Returned string information
 *          char    *szStr      input string to be formatted.
 *          int     iLen        Length of returned string
 *
 * RETURNS:     none
 *
 * COMMENTS:    The length parameter is the absolute length of the returned
 *          string in HTML characters. For example the input string > would
 *          be returned as &gt; which would be counted as 1 character.If
 *          the number of input characters is less than the iLen parameter
 *          spaces are appended to the end of the string to ensure that at
 *          least iLen characters are returned in the szBuff parameter.
 *
 */

static void FormatHTMLString(char *szBuff, char *szStr, int iLen)
{
    while( iLen && *szStr )
    {
        switch( *szStr )
        {
          case '>':
                *szBuff++ = '&';
                *szBuff++ = 'g';
                *szBuff++ = 't';
                *szBuff++ = ';';
                szStr++;
```

```
                break;
        case '<':
                *szBuff++ = '&';
                *szBuff++ = 'l';
                *szBuff++ = 't';
                *szBuff++ = ';';
                szStr++;
                break;
        case '&':
                *szBuff++ = '&';
                *szBuff++ = 'a';
                *szBuff++ = 'm';
                *szBuff++ = 'p';
                *szBuff++ = ';';
                szStr++;
                break;
        case '\"':
                *szBuff++ = '&';
                *szBuff++ = 'q';
                *szBuff++ = 'u';
                *szBuff++ = 'o';
                *szBuff++ = 't';
                *szBuff++ = ';';
                szStr++;
                break;
        default:
                *szBuff++ = *szStr++;
                break;
      }
      iLen--;
    }
    while( iLen-- )
    *szBuff++ = ' ';
    *szBuff = 0;
    return;
}

/*static int ThrTpInit(EXTENSION_CONTROL_BLOCK *pECB)
{
    // use CS for tptpalloc ?
    static int num_tpinits=0;
    static int x=1;
    static int once=0;
    static CRITICAL_SECTION     TpCriticalSection;
        int iRc, TpRc, lasterr;
        char buf[40];
        int retry = 0;

        BOOL Success = FALSE;

    if(!TlsGetValue(TLSIsTpInitedKey))
    {
                lasterr = GetLastError();
                if (lasterr != NO_ERROR)
                {
                        sprintf(buf,"TlsGetValue %d",lasterr);
                        ErrorMessage(pECB, ERR_TPINIT_BAD,
ERR_TYPE_TUXEDO, buf, 0, 0);
                        return -1;
                }
                if (!once)
                {
                        InitializeCriticalSection(&TpCriticalSection);
                                                once=1;
                }


                if ( dLog )
                {
                        FILE *fp;

                        fp = fopen(szTpccLogPath, "ab");
                        fprintf(fp, "* In ThrTpInit Thread %d * \r\n",
GetCurrentThreadId());
                        fclose(fp);
                }

                while ( retry < TP_MAX_RETRIES )
                {
                        EnterCriticalSection(&TpCriticalSection);

                        if(tpinf == NULL)
                        {
                          if ((tpinf = ( TPINIT *)tpalloc("TPINIT", NULL,
sizeof(TPINIT))) == NULL)
                            {
```

```
                TpRc = tperrno;
                if ( dLog )
                {
                        FILE *fp;

                        fp = fopen(szTpccLogPath, "ab");
                        fprintf(fp, ">>>> ThrTpInit %d : tpalloc
of tpinit failed: %d \r\n",
                                GetCurrentThreadId(), TpRc);
                        fclose(fp);


                }


                tpinf = NULL;
                retry++;
                LeaveCriticalSection(&TpCriticalSection);
                Sleep(50); // Relinquish thread timeslice
                continue;
          }

          tpinf->flags |= TPMULTICONTEXTS;
        }


        // Do the TPINIT
        itoa(++num_tpinits, tpinf->cltname, 10);
        iRc = tpinit(tpinf);

        // check tpinit() ?
        if (iRc < 0 )
        {
                TpRc = tperrno;

                lasterr = GetLastError();
                retry++ ;

                if ( dLog )
                {
                        FILE *fp;

                        fp = fopen(szTpccLogPath, "ab");
                        fprintf(fp, "* ThrTpInit Thread %d
Retry : count %d * \r\n",
                                GetCurrentThreadId(), retry);
                        fclose(fp);
                }
        }
        else
        {
                Success = TRUE;
                tpfree( ( char * ) tpinf);
                tpinf=NULL;
                break;
        }
        --num_tpinits;
        LeaveCriticalSection(&TpCriticalSection);
        Sleep(50);              // Relinquish thread timeslice

} // retry the tpinit if it failed the first time

LeaveCriticalSection(&TpCriticalSection);

if ( Success == FALSE )
{
        TpRc = tperrno;
        {
                FILE *fp;

                fp = fopen(szTpccLogPath, "ab");
                fprintf(fp, ">>>> ThrTpInit Thread %d Failed
tperrno= %d Last Err= %d \r\n",
                        GetCurrentThreadId(), TpRc, lasterr);
                fclose(fp);
        }
                sprintf(buf,"Thread %d: In ThrTpInit iRc=%d
Tperrno=%d",
                        GetCurrentThreadId(),iRc, TpRc);
                ErrorMessage(pECB, ERR_TPINIT_BAD,
ERR_TYPE_TUXEDO, buf, 0, 0);

                return -1;
        }
        if ( Success == TRUE )
        {
```

```
                        if ( dLog )
                        {
                                FILE *fp;

                                fp = fopen(szTpccLogPath, "ab");
                                fprintf(fp, "* ThrTpInit Thread %d Success retry
count %d * \r\n",
                                        GetCurrentThreadId(), retry);
                                fclose(fp);
                        }

                        if ( ( iRc=TlsSetValue(TLSIsTpInitedKey,&x)) == 0)
                        {
                                {
                                        FILE *fp;

                                        fp = fopen(szTpccLogPath, "ab");
                                        fprintf(fp, ">>>> ThrTpInit %d :
TlsSetValue Failed iRc: %d \r\n",
                                        GetCurrentThreadId(), iRc);
                                        fclose(fp);
                                }
                                sprintf(buf,"TlsSetValue %d",iRc);
                                ErrorMessage(pECB, ERR_TPINIT_BAD,
ERR_TYPE_TUXEDO, buf, 0, 0);
                                return -1;
                        }
                                if (dLog)
                                {
                                        FILE *fp;

                                        fp = fopen(szTpccLogPath, "ab");
                                        fprintf(fp, "*ThrTpInit %d : TlsSetValue
Failed iRc: v=%d \r\n",
                                        GetCurrentThreadId(),
TlsGetValue(TLSIsTpInitedKey));
                                        fclose(fp);
                                }
                        }
                }
                else
                {
                        if ( dLog )
                        {
                                FILE *fp;

                                fp = fopen(szTpccLogPath, "ab");
                                fprintf(fp, "* ThrTpInit Thread %d already tpinited * \r\n",
GetCurrentThreadId());
                                fclose(fp);
                        }
                }

        return 0;
}*/


static int ThrTpInit(EXTENSION_CONTROL_BLOCK *pECB)
{
        // use CS for tptpalloc ?
        int ret=0;
                static int num_tpinits=0;
        static int x=1;
        static int once=0;
        static apr_os_thread_t tid;
        unsigned long TLSIsTpInit;
        //static CRITICAL_SECTION     TpCriticalSection;
        static apr_thread_mutex_t*     TpCriticalSection=NULL;
                int iRc, TpRc, lasterr;
                char buf[40];
                int retry = 0;

                BOOL Success = FALSE;
                apr_pool_t* p;
                apr_thread_mutex_lock(loaded.lock);

                p = pECB->r->server->process->pool;
                // create critical section for this process
                if (!TpCriticalSection)

                apr_thread_mutex_create(&TpCriticalSection,APR_THREAD_MUTEX_D
EFAULT,p);
                {

                        FILE *fp;
                        tls_t key;
```

```
                key.pid = getpid();
                key.tid=apr_os_thread_current();
                TLSIsTpInit = (unsigned long) apr_hash_get(loaded.hash, &key,
sizeof(key));
                if (dLog)
                {
                fp = fopen(szTpccLogPath, "ab");
                fprintf(fp, "TlsGet pid=%d,tid= %d, count=%d,value=%ld \r\n",
                getpid(),GetCurrentThreadId(),
apr_hash_count(loaded.hash),TLSIsTpInit);
                fclose(fp);
                }
        }
        apr_thread_mutex_unlock(loaded.lock);

    //if(!TlsGetValue(TLSIsTpInitedKey))
   if(!TLSIsTpInit)
   {
                lasterr = GetLastError();
                if (lasterr != NO_ERROR)
                {
                        sprintf(buf,"TlsGetValue %d", lasterr);
                        ErrorMessage(pECB, ERR_TPINIT_BAD,
ERR_TYPE_TUXEDO, buf, 0, 0);
                        return -1;
                }
                if (!once)
                {
                        once=1;
                        if (dLog)
                {
                        FILE *fp;

                        fp = fopen(szTpccLogPath, "ab");
                        fprintf(fp, "* In ThrTpInit create cs pid=%d, tid= %d *
\r\n",getpid(), GetCurrentThreadId());
                        fclose(fp);
                }
                        // this is done in module_init
                        //mcInitializeCriticalSection(&(TpCriticalSection));

                }
```

```
                if ( dLog )
                {
                        FILE *fp;

                        fp = fopen(szTpccLogPath, "ab");
                        fprintf(fp, "* In ThrTpInit Thread %d * \r\n",
GetCurrentThreadId());
                        fclose(fp);
                }

                for ( retry = 0; retry < TP_MAX_RETRIES; retry++ )
                {
                        mcEnterCriticalSection(TpCriticalSection);


                        if(tpinf == NULL)
                        {
                          if ((tpinf = ( TPINIT *)tpalloc("TPINIT", NULL,
sizeof(TPINIT))) == NULL)
                            {
                                TpRc = tperrno;
                                lasterr = GetLastError();

                                if ( dLog )
                                {
                                        FILE *fp;

                                        fp = fopen(szTpccLogPath, "ab");
                                        fprintf(fp, ">>>> ThrTpInit %d : tpalloc
of tpinit failed: %d \r\n",
                                                GetCurrentThreadId(), TpRc);
                                        fclose(fp);


                                }


                                tpinf=NULL;
                                Sleep(50); // Relinquish thread timeslice
                                mcLeaveCriticalSection(TpCriticalSection);
                                continue;
                          }
                        }
```

```
                        // Do the TPINIT
        tpinf->flags |= TPMULTICONTEXTS;
                        //itoa(++num_tpinits, tpinf->cltname, 10);
                        sprintf(tpinf->cltname,"%d",++num_tpinits);
                        iRc = tpinit(tpinf);
                        if (iRc < 0 )
                        {
                                TpRc = tperrno;
                                lasterr = GetLastError();

                                if ( dLog )
                                {
                                        FILE *fp;

                                        fp = fopen(szTpccLogPath, "ab");
                                        fprintf(fp, "* ThrTpInit Thread %d
Retry : count %d * \r\n",
                                                GetCurrentThreadId(), retry);
                                        fclose(fp);
                                }

                                --num_tpinits;
                                Sleep(50);              // Relinquish thread timeslice
                                mcLeaveCriticalSection(TpCriticalSection);
                                continue;
                        }
                        if (dLog)
                                {
                                        FILE *fp;

                                        fp = fopen(szTpccLogPath, "ab");
                                        fprintf(fp, "* ThrTpInit
cltname=%s,pid=%d,Thread %d Retry : count %d * \r\n",tpinf->cltname,getpid(),
                                                GetCurrentThreadId(), retry);
                                        fclose(fp);
}

                        Success = TRUE;
                        tpfree( ( char * ) tpinf);
                        tpinf=NULL;
                        Sleep(50);              // Relinquish thread timeslice
                        mcLeaveCriticalSection(TpCriticalSection);
                                break;

                } // retry the tpinit if it failed the first time

                if ( Success == FALSE )
                {
                        if ( dLog )
                        {
                                FILE *fp;

                                fp = fopen(szTpccLogPath, "ab");
                                fprintf(fp, ">>>> ThrTpInit Thread %d Failed
tperrno= %d Last Err= %d, oser=%d \r\n",
                                        GetCurrentThreadId(), TpRc,
lasterr,Uunixerr);
                                fclose(fp);
                        }
                        sprintf(buf,"Thread %d: In ThrTpInit iRc=%d
Tperrno=%d",
                                GetCurrentThreadId(),iRc, TpRc);
                        ErrorMessage(pECB, ERR_TPINIT_BAD,
ERR_TYPE_TUXEDO, buf, 0, 0);

                //              return -1;
                        ret = -1;
                }
                else
                if ( Success == TRUE )
                {
                        if ( dLog )
                        {
                                FILE *fp;

                                fp = fopen(szTpccLogPath, "ab");
                                fprintf(fp, "* ThrTpInit Thread %d Success retry
count %d * \r\n",
                                        GetCurrentThreadId(), retry);
                                fclose(fp);
                        }

        //              if ( ( iRc=TlsSetValue(pECB->r->server,10)) == 0)

        apr_thread_mutex_lock(loaded.lock);
```

```
        {
                FILE *fp;
                unsigned long  v=10;
                tls_t* key;
                key=apr_pcalloc(loaded.pool, sizeof (tls_t));
                key->pid=getpid();
                key->tid=apr_os_thread_current();
                apr_hash_set(loaded.hash,key, sizeof(tls_t),(void*)v);
                v =(unsigned long)apr_hash_get(loaded.hash,key, sizeof(tls_t));
                if (dLog)
                {
                        fp = fopen(szTpccLogPath, "ab");
                        fprintf(fp, "TlsSet pid=%d,tid= %d, count=%d,value=%ld
\r\n",
                        getpid(),GetCurrentThreadId(),
apr_hash_count(loaded.hash),v);
                        fclose(fp);
                }
        }
        apr_thread_mutex_unlock(loaded.lock);
                        if (0)
                        {
                                {
                                        FILE *fp;

                                        fp = fopen(szTpccLogPath, "ab");
                                        fprintf(fp, ">>>> ThrTpInit %d :
TlsSetValue Failed iRc: %d \r\n",
                                        GetCurrentThreadId(), iRc);
                                        fclose(fp);
                                }
                                sprintf(buf,"TlsSetValue %d",iRc);
                                ErrorMessage(pECB, ERR_TPINIT_BAD,
ERR_TYPE_TUXEDO, buf, 0, 0);
                                ret= -1;
                        }
                }
        }
        else
        {
                if ( dLog )
                {
                        FILE *fp;
                        fp = fopen(szTpccLogPath, "ab");
                        fprintf(fp, "* ThrTpInit Thread %d already tpinited * \r\n",
GetCurrentThreadId());
                        fclose(fp);
                }
        }
    return ret;
}


#ifdef TOPEND

void space_fill(s1, s2, i)

char *s1, *s2;
int i;

{
    int ii;

    strncpy(s1,s2,i);

    for(ii=0; ii<i; ii++)
    {
        if (s1[ii] == '\0') s1[ii] = 0x20;
    }
    return;
}

/* This is the function for the new thread that is added to the process. */
/* The error handling needs to be improved.  Almost every problem is fatal. */
static void TEReceiveThread(void *ptr)
{
    tp_dif_structs_t *dif_struct;
    long buffer_length;
    char *message_buffer;
    int rc;
    FILE *fp1;

   if ((rc = (int)tp_ChangeToGroup ((LPSTR)getenv("TP_SYSTEM"),
        LOGON32_LOGON_INTERACTIVE,
        LOGON32_PROVIDER_DEFAULT)) != 0)
```

```
    {
        if (bLog)
        {
          fp1 = fopen(szTpccLogPath, "ab");
          fprintf(fp1, "Received bad status from tp_ChangeToGroup, status = %d
***\n", rc);
          fclose(fp1);
        }
                MT_LOG_ERROR;
        return;
    }

    dif_struct = tp_csi_alloc (TP_DIF_DIAL_INFO |
                               TP_DIF_SERVICE_NAME |
                               TP_DIF_OUTPUT_FORMAT |
                               TP_DIF_LOCATION);

    if (dif_struct == NULL)
    {
        MT_LOG_ERROR;
        return;
    }

    if ((message_buffer = (char *) malloc (TP_MAX_BUF_LEN)) == NULL)
    {
        MT_LOG_ERROR;
        return;
    }

    /* receive loop.  Should there be a way to shut this down? */
    for (;;)
    {
        if (tp_WaitForSingleObject (NULL, INFINITE) == WAIT_TIMEOUT)
        {
          MT_LOG_ERROR;
          return;
        }
        dif_struct->info->tp_user_dialogue_id = TP_ANY_DIALOGUE;
        dif_struct->info->tp_flags = TP_NOFLAGS;
        buffer_length = TP_MAX_BUF_LEN;

        rc = tp_client_receive (dif_struct->info,TP_NOBLOCK,
                                dif_struct->output_format,
                                dif_struct->service,
                                dif_struct->location,
                                &buffer_length,
                                message_buffer);
    switch (rc)
    {
      case TP_TIMEOUT:
      {
            break;
      }
      case TP_ADMIN:
      {
    /* This assumes that admin messages aren't used in the application. */
    /* Receive the message and throw it away. */
            long flags, buf_len;
            char buffer[1];

            flags = TP_TRUNCATE;
            buf_len = 1;
            tp_system_admin (&flags, &buf_len, buffer);
            break;
      }
      case TP_DISSOLVED:
      case TP_OK:
      case TP_RESET:
      case TP_SERVICE:
      case TP_SIGNON_INHIBITED:
      case TP_USER:
      {
            DialogTableEntry_t *DTE;
            void *ptr;

            DTE = FindOnList (dif_struct->info->tp_user_dialogue_id,
&DTActiveList);
            if (DTE == NULL)
            {
            /* We received in a dialog without outstanding work???? */
              MT_LOG_ERROR;
              return;
            }

            /* trade data with DTE. */
            ptr = DTE->dif_struct->info;
```

```
        DTE->dif_struct->info = dif_struct->info;
        dif_struct->info = ptr;

        ptr = DTE->dif_struct->output_format;
        DTE->dif_struct->output_format = dif_struct->output_format;
        dif_struct->output_format = ptr;

        ptr = DTE->dif_struct->service;
        DTE->dif_struct->service = dif_struct->service;
        dif_struct->service = ptr;

        ptr = DTE->dif_struct->location;
        DTE->dif_struct->location = dif_struct->location;
        dif_struct->location = ptr;

        DTE->buffer_length = buffer_length;

        if (DTE->message_buffer == NULL)
        {
          if ((DTE->message_buffer = malloc (buffer_length)) == NULL)
          {
                MT_LOG_ERROR;
                return;
          }
        }
        else
        {
          /* NONPORTABLE */
          if (_msize (DTE->message_buffer) < (size_t) buffer_length)
          {
                if ((DTE->message_buffer = realloc (DTE->message_buffer,
                    buffer_length)) == NULL)
                {
                  MT_LOG_ERROR;
                  return;
                }
          }
        }
        memcpy (DTE->message_buffer, message_buffer, buffer_length);
        DTE->rc = rc;
        /* Signal that the data is ready. */
        SetEvent (DTE->hSync);
                break;
          }
          case TP_SHUTDOWN:
          {
                DialogTableEntry_t *DTE;
                /* tell everyone */
                if (WaitForSingleObject (DTActiveList.hSync, INFINITE)  ==
WAIT_TIMEOUT)
                {
                  MT_LOG_ERROR;
                  return;
                }
                DTE = DTActiveList.next;
                while (DTE != NULL)
                {
                  DTE->rc = TP_SHUTDOWN;
                  SetEvent (DTE->hSync);
                  DTE = DTE->next;
                }
                if (!ReleaseMutex(DTActiveList.hSync))
                {
                  MT_LOG_ERROR;
                  return;
                }
                break;
          }
          default:
          {
                if (bLog)
                {
                  fp1 = fopen(szTpccLogPath, "ab");
                  fprintf(fp1, "TEReceiveThread received error status = %d\n", rc);
                  fclose(fp1);
                }
                MT_LOG_ERROR;
                return;
          }
        }
    }
}
```

```
int tp_mt_initialize(tp_application_info_t *application_info,
                     tp_funct_struct_t     function_array[],
                     long                  number_functions)
{
    static HANDLE hReceiveThread = NULL;
    int rc;

          rc = tp_initialize(application_info, function_array, number_functions);

    if (rc == TP_OK)
    {
    /* Make sure that the global data is initialized. */
        if (DTFreeList.hSync == NULL)
        {
          if ((DTFreeList.hSync = CreateMutex(NULL, FALSE, NULL)) ==
NULL)
          {
                MT_LOG_ERROR;
                tp_terminate();
                return TP_DIFERR;
          }
        }
        if (DTActiveList.hSync == NULL)
        {
          if ((DTActiveList.hSync = CreateMutex(NULL, FALSE, NULL)) ==
NULL)
          {
                MT_LOG_ERROR;
                tp_terminate();
                return TP_DIFERR;
          }
        }
        /* Make sure that the receive thread is running */
        if (hReceiveThread == NULL)
        {
          if ((hReceiveThread = (HANDLE) _beginthread(TEReceiveThread, 0,
NULL)) == (HANDLE) -1 )
          {
                MT_LOG_ERROR;
                tp_terminate();
                return TP_DIFERR;
          }
          if (!SetThreadPriority (hReceiveThread,
RECEIVE_THREAD_PRIORITY))
          {
                MT_LOG_ERROR;
          }
        }
    }
    return rc;
}


int tp_mt_client_signon(tp_dialogue_info_t *info,
                        tp_dialogue_user_t *client,
                        long               inactivity_time,
                        tp_service_name_t  *service,
                        tp_input_format_t  *input_format,
                        long               message_length,
                        char               *message_text)
{
    DialogTableEntry_t *DTE = GetDTE();
    int rc;

    if (DTE == NULL)
    {
        MT_LOG_ERROR;
        return TP_DIFERR;
    }
    if (!ResetEvent (DTE->hSync))
    {
        MT_LOG_ERROR;
        return TP_DIFERR;
    }

    /* This is to handle TP_UNIQUE_DIALOGUE */
    DTE->usr_dlg_ptr = &(info->tp_user_dialogue_id);
    AddToList(DTE, &DTActiveList);

    rc = tp_client_signon(info, client, inactivity_time, service,
                          input_format, message_length, message_text);

    switch (rc)
    {
        case TP_DISSOLVING:
        case TP_OK:
```

```
        {
          DTE->usr_dlg_storage = info->tp_user_dialogue_id;
          DTE->usr_dlg_ptr = &(DTE->usr_dlg_storage);
          break;
        }
        default:
        {
        /* there is nothing to receive */
          RemoveFromList (DTE, &DTActiveList);
          AddToList (DTE, &DTFreeList);
        }
    }
    return rc;
}


int tp_mt_client_send(tp_dialogue_info_t *info,
                      tp_service_name_t  *service,
                      tp_input_format_t  *input_format,
                      long               message_length,
                      char               *message_text)
{
    DialogTableEntry_t *DTE = GetDTE();
    int rc;

    if (DTE == NULL)
    {
        MT_LOG_ERROR;
        return TP_DIFERR;
    }
    if (!ResetEvent (DTE->hSync))
    {
        MT_LOG_ERROR;
        return TP_DIFERR;
    }
    DTE->usr_dlg_storage = info->tp_user_dialogue_id;
    DTE->usr_dlg_ptr = &(DTE->usr_dlg_storage);
    AddToList(DTE, &DTActiveList);

    rc = tp_client_send(info, service, input_format, message_length, message_text);
    switch (rc)
    {
        case TP_DISSOLVING:
        case TP_OK:
        case TP_RESET:
        {
          break;
        }
        default:
        {
          /* there is nothing to receive */
          RemoveFromList (DTE, &DTActiveList);
          AddToList (DTE, &DTFreeList);
        }
    }
    return rc;
}


int tp_mt_client_receive(tp_dialogue_info_t *info,
                         long               wait_time,
                         tp_output_format_t *output_format,
                         tp_service_name_t  *service,
                         tp_location_t      *location,
                         long               *buffer_length,
                         char               *message_buffer)
{
    DialogTableEntry_t *DTE = FindOnList(info->tp_user_dialogue_id,
&DTActiveList);
    int rc;
    FILE *fp1;

    if (DTE == NULL)
    {
        MT_LOG_ERROR;
        return TP_DIFERR;
    }
    wait_time = TP_BLOCK;

    /* Wait for the receive to actually happen */
    if (WaitForSingleObject(DTE->hSync, INFINITE) == WAIT_TIMEOUT)
    {
        if (bLog)
        {
          fp1 = fopen(szTpccLogPath, "ab");
```

```
            fprintf(fp1, "tp_mt_client_receive got a timeout error\n");
            fclose(fp1);
          }
          return TP_TIMEOUT;
    }

    if (DTE->buffer_length > *buffer_length)
    {
          *buffer_length = DTE->buffer_length;
          return TP_BUFFSIZE;
    }

    RemoveFromList (DTE, &DTActiveList);

    memcpy (info, DTE->dif_struct->info, sizeof (tp_dialogue_info_t));
    memcpy (output_format, DTE->dif_struct->output_format, sizeof
(tp_output_format_t));
    memcpy (service, DTE->dif_struct->service, sizeof (tp_service_name_t));
    memcpy (location, DTE->dif_struct->location, sizeof (tp_location_t));
    *buffer_length = DTE->buffer_length;
    memcpy (message_buffer, DTE->message_buffer, *buffer_length);
    rc = DTE->rc;

    AddToList (DTE, &DTFreeList);
    return rc;
}


void AddToList (DialogTableEntry_t *DTE, DialogTableEntry_t *head)
{
    if (WaitForSingleObject (head->hSync, INFINITE) == WAIT_TIMEOUT)
    {
          MT_LOG_ERROR;
          return;
    }
    DTE->next = head->next;
    head->next = DTE;
    if (!ReleaseMutex (head->hSync))
    {
          MT_LOG_ERROR;
    }
    return;
}
```

```
void RemoveFromList (DialogTableEntry_t *DTE, DialogTableEntry_t *head)
{
    DialogTableEntry_t *ptr;

    if (WaitForSingleObject (head->hSync, INFINITE) == WAIT_TIMEOUT)
    {
          MT_LOG_ERROR;
          return;
    }
    ptr = head;

    while ((ptr != NULL) && (ptr->next != DTE))
    {
          ptr = ptr->next;
    }
    if (ptr == NULL)
    {
          MT_LOG_ERROR;
          ReleaseMutex (head->hSync);
          return;
    }
    ptr->next = DTE->next;

    if (!ReleaseMutex (head->hSync))
    {
    MT_LOG_ERROR;
    }
    return;
}


DialogTableEntry_t *FindOnList (long tp_user_dialogue_id, DialogTableEntry_t
*head)
{
    DialogTableEntry_t *ptr;

    if (WaitForSingleObject (head->hSync, INFINITE) == WAIT_TIMEOUT)
    {
          MT_LOG_ERROR;
          return NULL;
    }
```

```
    ptr = head->next;

    while ((ptr != NULL) && (*(ptr->usr_dlg_ptr) != tp_user_dialogue_id))
    {
        ptr = ptr->next;
    }
    if (ptr == NULL)
    {
        MT_LOG_ERROR;
        ReleaseMutex (head->hSync);
        return NULL;
    }
    if (!ReleaseMutex (head->hSync))
    {
        MT_LOG_ERROR;
    }
    return ptr;
}

DialogTableEntry_t *GetDTE()
{
    DialogTableEntry_t *ptr = NULL;

    if (WaitForSingleObject (DTFreeList.hSync, INFINITE) == WAIT_TIMEOUT)
    {
        MT_LOG_ERROR;
        return NULL;
    }

    if (DTFreeList.next != NULL)
    {
        ptr = DTFreeList.next;
        DTFreeList.next = ptr->next;
    }

    if (!ReleaseMutex (DTFreeList.hSync))
    {
        MT_LOG_ERROR;
    }

    if (ptr == NULL)
    {
        if ((ptr = (DialogTableEntry_t *) malloc (sizeof (DialogTableEntry_t)))
==NULL)
        {
          MT_LOG_ERROR;
          return NULL;
        }

        if ((ptr->dif_struct = tp_csi_alloc (TP_DIF_DIAL_INFO |
                                             TP_DIF_SERVICE_NAME |
                                             TP_DIF_OUTPUT_FORMAT |
                                             TP_DIF_LOCATION)) == NULL)
        {
          MT_LOG_ERROR;
          free(ptr);
          return NULL;
        }
        ptr->buffer_length = 0;
        ptr->message_buffer = NULL;
        ptr->rc = 0;
        if ((ptr->hSync = CreateEvent (NULL, TRUE, FALSE,NULL)) == NULL)
        {
          MT_LOG_ERROR;
          tp_csi_free (ptr->dif_struct);
          free (ptr);
          return NULL;
        }
        ptr->next = NULL;
    }
    return ptr;
}

#endif
```

tpccerr.h

```
#ifndef TPCCERR_H
#define TPCCERR_H

/*============================================================
======+
|       Copyright (c) 1997  Oracle Corp, Redwood Shores, CA     |
|                    All Rights Reserved                        |
```

```
+=============================================================
======+
| FILENAME
|   tpccerr.h
| DESCRIPTION
|   Include file for TPC-C benchmark programs.

+=============================================================
======*/

//#define LIMIT_TUXQUEUE

#define TP_MAX_RETRIES  1
#define  ERR_TYPE_WEBDLL    1
#define  ERR_TYPE_SQL       2
#define  ERR_TYPE_OCI3
#define ERR_TYPE_TUXEDO 4
#define ERR_DB_SUCCESS      0
/* Database transaction succeeded */

#define ERR_DB_ERROR                1       /* A database error has occured
*/
#define ERR_TRANSPORT_ERROR         2       /* A transport error has occured
*/
#define ERR_DB_INTERFACE    3       /* An error occured setting up */
                                    /* connection to DB */
#define ERR_DB_DEADLOCK_LIMIT       4       /* The DB deadlock retry limit
was */
                                    /* reached */
#define ERR_DB_NOT_COMMITTED        5       /* Transaction not committed */
#define ERR_DB_DEAD         6       /* Database connection is invalid */
#define  ERR_SUCCESS        1000    //"Success, no error.
#define  ERR_COMMAND_UNDEFINED 1001     //"Command undefined.
#define  ERR_NOT_IMPLEMENTED_YET        1002     //"Not Implemented
Yet.
#define  ERR_CANNOT_INIT_TERMINAL  1003  //"Cannot initialize client
connection.
#define  ERR_OUT_OF_MEMORY      1004     //"insufficient memory.
#define  ERR_NEW_ORDER_NOT_PROCESSED 1005  //"Cannot process new
Order form.
#define  ERR_PAYMENT_NOT_PROCESSED  1006        //"Cannot process
payment form.
#define  ERR_NO_SERVER_SPECIFIED  1007     //"No Server name specified.
#define  ERR_ORDER_STATUS_NOT_PROCESSED 1008 //"Cannot process order
status form.
#define  ERR_W_ID_INVALID    1009     //"Invalid Warehouse ID.
#define  ERR_CAN_NOT_SET_MAX_CONNECTIONS     1010     //"Insufficient
memory to
                                                //allocate # connections.
#define  ERR_NOSUCH_CUSTOMER     1011     //"No such customer.
#define  ERR_D_ID_INVALID    1012     //"Invalid District ID Must be 1 to 10.
#define  ERR_MAX_CONNECT_PARAM 1013     //"Max client connections
exceeded,
                                        //run install to increase.
#define  ERR_INVALID_SYNC_CONNECTION 1014 //"Invalid Terminal Sync ID.
#define  ERR_INVALID_TERMID          1015     //"Invalid Terminal ID.
#define  ERR_PAYMENT_INVALID_CUSTOMER 1016 //"Payment Form, No
such Customer.
#define  ERR_SQL_OPEN_CONNECTION            1017
        //"SQLOpenConnection API Failed.
#define  ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY 1018  //"Stock
Level missing
                                                //Threshold key "TT*".
#define ERR_STOCKLEVEL_THRESHOLD_INVALID 1019 //"Stock Level
Threshold invalid
                                                //data type range = 1 - 99.
#define  ERR_STOCKLEVEL_THRESHOLD_RANGE 1020   //"Stock Level
Threshold out of
                                                //range, range must be 1 - 99.
#define ERR_STOCKLEVEL_NOT_PROCESSED 1021  //"Stock Level not
processed.
#define  ERR_NEWORDER_FORM_MISSING_DID 1022 //"NewOrder miss
District key "DID*".
#define ERR_NEWORDER_DISTRICT_INVALID 1023 //"NewOrder District ID
Invalid
                                                //range 1 - 10.
#define  ERR_NEWORDER_DISTRICT_RANGE  1024  //"New Order District ID
out of
                                                //Range. Range = 1 - 10.
#define  ERR_NEWORDER_CUSTOMER_KEY 1025 //"New Order missing
Customer key "CID*".
#define  ERR_NEWORDER_CUSTOMER_INVALID 1026 //"NewOrder customer
id invalid data
                                                //type, range = 1 to 3000.
#define  ERR_NEWORDER_CUSTOMER_RANGE 1027   //"New Order customer
id out of
```

```
                                        //range, range = 1 to 3000.
#define  ERR_NEWORDER_MISSING_IID_KEY 1028  //"NewOrder missng
ItemId key "IID*".
#define ERR_NEWORDER_ITEM_BLANK_LINES 1029 //"NewOrder blank order
lines all
                                        //orders must be continuous.
#define ERR_NEWORDER_ITEMID_INVALID 1030   //"New Order Item Id is
wrong data
                                        //type, must be numeric.
#define ERR_NEWORDER_MISSING_SUPPW_KEY 1031 //"NewOrder missing
Supp_W key
                                         //"SP##*".
#define  ERR_NEWORDER_SUPPW_INVALID 1032          //"New Order
Supp_W invalid data
                                        //type must be numeric.
#define  ERR_NEWORDER_MISSING_QTY_KEY 1033  //"NewOrder Missing
Qty key "Qty##*".
#define ERR_NEWORDER_QTY_INVALID 1034     //"NewOrder Qty invalid must
be
                                        //numeric range 1 - 99.
#define  ERR_NEWORDER_SUPPW_RANGE 1035          //"New Order
Supp_W value out of
                                        //range range = 1 - Max Warehouses.
#define  ERR_NEWORDER_ITEMID_RANGE 1036          //"New Order Item Id
is out of range.
                                        //Range = 1 to 999999.
#define  ERR_NEWORDER_QTY_RANGE          1037     //"New Order Qty is
out of range.
                                       //Range = 1 to 99.
#define ERR_PAYMENT_DISTRICT_INVALID 1038  //"Payment District ID is
invalid
                                        //must be 1 - 10.
#define ERR_NEWORDER_SUPPW_WITHOUT_ITEMID 1039 //"NewOrder
Supp_W field entered
                                         //without a corrosponding Item_Id.
#define  ERR_NEWORDER_QTY_WITHOUT_ITEMID 1040  //"NewOrder Qty
entered without a
                                         //corrosponding Item_Id.
#define  ERR_NEWORDER_NOITEMS_ENTERED          1041  //"NewOrder
Blank Items between
                                         //items, items must be continuous.
#define  ERR_PAYMENT_MISSING_DID_KEY 1042 //"Payment missing District
Key "DID*".
#define  ERR_PAYMENT_DISTRICT_RANGE 1043          //"Payment District
Out of range,
                                         //range = 1 - 10.
#define  ERR_PAYMENT_MISSING_CID_KEY 1044 //"Payment missing
Customer Key "CID*".
#define  ERR_PAYMENT_CUSTOMER_INVALID 1045 //"PaymentCustomer data
type invalid,
                                          //must be numeric.
#define  ERR_PAYMENT_MISSING_CLT 1046      //"Payment missing Customer
Last Name
                                        //Key "CLT*".
#define  ERR_PAYMENT_LAST_NAME_TO_LONG 1047   //"Payment Customer
last name
                                         // longer than 16 characters.
#define  ERR_PAYMENT_CUSTOMER_RANGE 1048        //"Payment Customer
ID out of range,
                                        //must be 1 to 3000.
#define  ERR_PAYMENT_CID_AND_CLT          1049      //"Payment Customer
ID and Last Name
                                        //entered must be one or other.
#define ERR_PAYMENT_MISSING_CDI_KEY 1050 //"Payment missing Customer
district
                                         //key "CDI*".
#define ERR_PAYMENT_CDI_INVALID  1051      //"Payment Customer district
invalid
                                        //must be numeric.
#define  ERR_PAYMENT_CDI_RANGE   1052      //"Payment Customer district
out of
                                        //range must be 1 - 10.
#define ERR_PAYMENT_MISSING_CWI_KEY 1053 //"Payment missing
Customer Warehouse
                                         //key "CWI*".
#define  ERR_PAYMENT_CWI_INVALID          1054      //"Payment Customer
Warehouse invalid
                                        //must be numeric.
#define  ERR_PAYMENT_CWI_RANGE   1055      //"Payment Customer
Warehouse out of
                                        //range, 1 to Max Warehouses.
#define  ERR_PAYMENT_MISSING_HAM_KEY 1056  //"Payment missing
Amount key "HAM*".
#define  ERR_PAYMENT_HAM_INVALID          1057      //"Payment Amount
invalid data type
                                        //must be numeric.
```

```
#define  ERR_PAYMENT_HAM_RANGE 1058      //"Payment Amount out of
range,
                                        //0 - 9999.99.
#define  ERR_ORDERSTATUS_MISSING_DID_KEY         1059 //"OrderStatus
missing District
                                          //key "DID*".
#define  ERR_ORDERSTATUS_DID_INVALID 1060 //"Order Status District
invalid,
                                         //value must be numeric 1 - 10.
#define  ERR_ORDERSTATUS_DID_RANGE 1061          //"Order Status District
out of range
                                        //must be 1 - 10.
#define ERR_ORDERSTATUS_MISSING_CID_KEY 1062 //"OrderStatus missing
Customer
                                          //key "CID*".
#define  ERR_ORDERSTATUS_MISSING_CLT_KEY 1063 //"OrderStatus missing
Customer
                                          //Last Name key "CLT*".
#define  ERR_ORDERSTATUS_CLT_RANGE 1064          //"Order Status
Customer last name
                                        //longer than 16 characters.
#define  ERR_ORDERSTATUS_CID_INVALID 1065 //"Order Status Customer ID
invalid,
                                         //range must be numeric 1 - 3000.
#define  ERR_ORDERSTATUS_CID_RANGE 1066          //"Order Status
Customer ID out of
                                        //range must be 1 - 3000.
#define  ERR_ORDERSTATUS_CID_AND_CLT 1067 //"Order Status Customer ID
and
                                         //LastName entered must be only one."
#define  ERR_DELIVERY_MISSING_OCD_KEY 1068 //"Delivery missing Carrier
ID key
                                          //\"OCD*\".
#define  ERR_DELIVERY_CARRIER_INVALID 1069 //"Delivery Carrier ID
invalid must
                                         //be numeric 1 - 10.
#define  ERR_DELIVERY_CARRIER_ID_RANGE 1070 //"Delivery Carrier ID out
of range
                                         //must be 1 - 10.
#define ERR_PAYMENT_MISSING_CLT_KEY 1071   //"Payment missing
Customer Last
                                          //Name key "CLT*".
#define TOPEND_SEND_ERROR 1072 //"TOPEND client send error".
#define TOPEND_RECEIVE_ERROR 1073 //"TOPEND client receive error".

#ifdef LIMIT_TUXQUEUE
#define ERR_TUXQUEUE_TIMEOUT 1074
#define ERR_TUXQUEUE_FAILED 1075
#endif // LIMIT_TUXQUEUE

#ifdef LIMIT_TUXQUEUE2
#define ERR_TUXQUEUE_TIMEOUT 1074
#define ERR_TUXQUEUE_FAILED 1075
#endif // LIMIT_TUXQUEUE

#define ERR_TPINIT_BAD 5001  //"Tuxedo tpinit Failed."
#define ERR_TPALLOC_BAD 5002 //"Tuxedo tpalloc Failed."
#define ERR_TPCALL_BAD  5003 //"Tuxedo tpcall Failed."

#endif /* TPCCERR_H */
```

tpccapi.h

```
#ifndef TPCCAPI_H
#define TPCCAPI_H

/*=============================================================
======+
 |      Copyright (c) 1997  Oracle Corp, Redwood Shores, CA      |
 |                OPEN SYSTEMS PERFORMANCE GROUP                 |
 |                    All Rights Reserved                        |

 +=============================================================
======+
 | FILENAME
 |   tpccapi.h
 | DESCRIPTION
 |   header file to tpcc.dll

 +=============================================================
======*/


//VERSION RESOURCE DEFINES
#define BOOL int
```

```
#define TRUE 1
#define FALSE 0
#define _APS_NEXT_RESOURCE_VALUE       101
#define _APS_NEXT_COMMAND_VALUE        40001
#define _APS_NEXT_CONTROL_VALUE        1000
#define _APS_NEXT_SYMED_VALUE          101

typedef char        *LPSTR;

    //note that the welcome form must be processed first as terminal ids
    //assigned here, once the terminal id is assigned then the forms can
    //be processed in any order.

#define WELCOME_FORM    1
                        //beginning form no term id assigned, form id
#define MAIN_MENU_FORM  2
                        //term id assigned main menu form id
#define NEW_ORDER_FORM  3
                        //new order form id
#define PAYMENT_FORM    4
                        //payment form id
#define DELIVERY_FORM   5
                        //delivery form id
#define ORDER_STATUS_FORM       6
                        //order status id
#define STOCK_LEVEL_FORM        7
                        //stock level form id

    //This macro is used to prevent the compiler error unused formal parameter
#define UNUSEDPARAM(x) (x = x)

// this macro for convert IIS struct to Apache
#define EXTENSION_CONTROL_BLOCK                 ext_ctrl_block

    //error message structure used in ErrorMessage API
typedef struct _SERRORMSG
{
    int      iError;            //error id of message
    char     szMsg[80];         //message to sent to browser
} SERRORMSG;

        //This structure defines the data necessary to keep distinct for each
        //terminal or client connection.

typedef struct _CLIENTDATA
{
    int      inUse;         //in use flag allows client entries to be reused
    int      w_id;          //warehouse id assigned at welcome form
    int      d_id;          //district id assigned at welcome form
    int      iProcessRequestCounter;        //request counter
    int      iSyncId;       //syncronization id
    int      iTickCount;    //time of last access;
    int      iTermId;       //terminal id of http stream connection
    char     szBuffer[4096]; //form buffer; HTML form built for client here
    //for TOPEND
    char     type;
    int      retval;
    int      client;
    BOOL     bFailed;
#ifdef TOPEND
    long     *tp_structadr;
#else // assume tuxedo
    struct newstruct        *newOrderDataPtr;      //new order Tuxedo buffer
    struct paystruct        *paymentDataPtr;       //payment Tuxedo buffer
    struct ordstruct        *orderStatusDataPtr;   //order status Tuxedo buffer
    struct delstruct        *deliveryDataPtr;      //delivery Tuxedo buffer
    struct stostruct        *stockLevelDataPtr;    //stock level Tuxedo buffer
#endif
    union {
    struct newstruct        newOrderData;  //new order form data
    struct paystruct        paymentData;   //payment form data
    struct ordstruct  orderStatusData;     //order status form data
    struct delstruct        deliveryData;  //delivery form data
    struct stostruct        stocklevelData; //stock level form data
    } trans;
} CLIENTDATA;

typedef CLIENTDATA *PCLIENTDATA;       //pointer to client structure
        //This structure is used to define the operational interface for
        //terminal id support
typedef struct _TERM
{
int     iAvailable;    //total allocated terminal array entries
int     iNext;         //next available terminal array element
int     iMasterSyncId;  //syncronization id
BOOL    bInit;         //structure has been initialized flag
CLIENTDATA      *pClientData;  //pointer to allocated client data
```

```
void    (*Init)(void);  //API to initialize this structure
int     (*Allocate)(void); //API to allocate new terminal entry;array id returnd
void    (*Restore)(void);       //API to free terminal data
//int     (*Add)(EXTENSION_CONTROL_BLOCK *pECB, char *pQueryString);
//API to add a terminal id to array,
int     (*Add)(EXTENSION_CONTROL_BLOCK *, char *);       //API to add a
terminal id to array,
                                        //this context will be passed from the
                                        //browser to the tpcc.dll in the
                                        //TERMID= key in the HTTP string.
void    (*Delete)(EXTENSION_CONTROL_BLOCK *pECB, int id); //API to free
                                                //resources used by a
                                                //terminal array
                                                //entry
apr_pool_t *p;  // per server config pool
} TERM;

typedef TERM *PTERM;    //pointer to terminal structure type

    //this structure allows the EXTENSION CONTROL BLOCK to be passed to the
    //msg and error handlers.
typedef struct _ECBINFO
{
    int      iTermId;       //terminal id
    int      iSyncId;       //browser sync id
    BOOL     bDeadlock;     //deadlock condition flag
    BOOL     bFailed;       //cleared before sql transaction, set in err
                            //handlers if an error occurs
    EXTENSION_CONTROL_BLOCK *pECB;    //inetsrv current connection
structure
                                        //information
} ECBINFO, *PECBINFO;

//function prototypes

#ifdef TOPEND // in tuxedo defined in atmi.h
extern int tpsvrinit (int argc, char * argv[]);
            // argv arguments are  int pid, char *uid, char *pwd, int txntype

extern int NEWORDER(CLIENTDATA *jobData, NewOrderData *neword, int
deadlock);
extern int PAYMENT(CLIENTDATA *jobData, PaymentData *paydata, int
deadlock);
extern int ORDERSTATUS(CLIENTDATA *jobData, OrderStatusData *orddata, int
deadlock);
extern int STOCKLEVEL(CLIENTDATA *jobData, StockLevelData *stodata, int
deadlock);
extern int DELIVERY(CLIENTDATA *jobData, DeliveryData *deldata, int
deadlock);
#endif

#ifdef TUX
extern void NEWORDER(TPSVCINFO *msg);
extern void PAYMENT(TPSVCINFO *msg);
extern void ORDERSTATUS(TPSVCINFO *msg);
extern void STOCKLEVEL(TPSVCINFO *msg);
extern void DELIVERY(TPSVCINFO *msg);
#endif


//BOOL APIENTRY DllMain(HANDLE hModule, DWORD ul_reason_for_call,
LPVOID lpReserved);
static BOOL IsValidTermId(int TermId);
BOOL ProcessQueryString(EXTENSION_CONTROL_BLOCK *pECB, int *pCmd,
int *pFormId, int *pTermId, int *pSyncId);
void NewOrderForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId);
void PaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId);
void DeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId);
void OrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId);
void StockLevelForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId);
void Exitcmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId,
int iSyncId);
void SubmitCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId);
void BeginCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId);
void ProcessCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId);
void ClearCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId,
int iSyncId);
void MenuCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
```

```
iTermId, int iSyncId);
static void WriteZString(EXTENSION_CONTROL_BLOCK *pECB, char *szStr);
static void h_printf(EXTENSION_CONTROL_BLOCK *pECB, char *format, ...);
void ErrorMessage(EXTENSION_CONTROL_BLOCK *pECB, int iError, int
iErrorType, char *szMsg, int iTermId, int iSyncId);
static BOOL GetKeyValue(char *pQueryString, char *pKey, char *pValue, int
iMax);
void TermInit(server_rec *s);
static void TermRestore(EXTENSION_CONTROL_BLOCK *pECB);
int TermAllocate(server_rec* s);
static int TermAdd(EXTENSION_CONTROL_BLOCK *pECB, char
*pQueryString);
static void TermDelete(EXTENSION_CONTROL_BLOCK *pECB, int id);
BOOL Init(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId,
char *szServer, char *szUser, char *szPassword, char *szDatabase);
static BOOL Close(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId);
static void FormatString(char *szDest, char *szPic, char *szSrc);
static char *MakeStockLevelForm(int iTermId, int iSyncId, BOOL bInput);
static char *MakeMainMenuForm(int iTermId, int iSyncId);
static char *MakeWelcomeForm(void);
static char *MakeNewOrderForm(int iTermId, int iSyncId, BOOL bInput, BOOL
bValid, char *execution_status);
static char *MakePaymentForm(int iTermId, int iSyncId, BOOL bInput);
static char *MakeOrderStatusForm(int iTermId, int iSyncId, BOOL bInput);
static char *MakeDeliveryForm(int iTermId, int iSyncId, BOOL bInput, char
*execution_status);
void UtilStrCpy(char * pDest, char * pSrc, int n);
static void ProcessNewOrderForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId);
static void ProcessPaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId);
static void ProcessOrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId);
static void ProcessDeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId);
static void ProcessStockLevelForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId);
static int GetNewOrderData(LPSTR lpszQueryString, NewOrderData
*pNewOrderData);
static int GetPaymentData(LPSTR lpszQueryString, PaymentData *pPaymentData);
static int GetOrderStatusData(LPSTR lpszQueryString, OrderStatusData
*pOrderStatusData);
static BOOL ReadRegistrySettings(void);
static BOOL IsNumeric(char *ptr);
static void FormatHTMLString(char *szBuff, char *szStr, int iLen);
static void Log(char *szType, char *szStr);
int do_neworder(NewOrderData *pNOData);
int do_payment(PaymentData *pPAYData);
int do_orderstatus(OrderStatusData *pOSData);
int do_delivery(pDeliveryData pDelivery);
int do_stocklevel(StockLevelData *pSTOData);
int end_neworder(NewOrderData *pNOData);
int end_payment(PaymentData *pPAYData);
int end_orderstatus(OrderStatusData *pOSData);
int end_stocklevel(StockLevelData *pSTOData);
int do_all_txns(int txn);
int do_disconnect( );
DWORD           HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB);
apr_status_t tpcc_shm_create(tpcc_shm_t* m, apr_pool_t* p);

#endif /* TPCCAPI_H */
```

mod_tpcc.c

```
/*
============================================================
=======
 * The Apache Software License, Version 1.1
 *
 * Copyright (c) 2000-2003 The Apache Software Foundation.  All rights
 * reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. The end-user documentation included with the redistribution,
```

```
 *    if any, must include the following acknowledgment:
 *       "This product includes software developed by the
 *        Apache Software Foundation (http://www.apache.org/)."
 *    Alternately, this acknowledgment may appear in the software itself,
 *    if and wherever such third-party acknowledgments normally appear.
 *
 * 4. The names "Apache" and "Apache Software Foundation" must
 *    not be used to endorse or promote products derived from this
 *    software without prior written permission. For written
 *    permission, please contact apache@apache.org.
 *
 * 5. Products derived from this software may not be called "Apache",
 *    nor may "Apache" appear in their name, without prior written
 *    permission of the Apache Software Foundation.
 *
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR
IMPLIED
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES
 * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE
 * DISCLAIMED.  IN NO EVENT SHALL THE APACHE SOFTWARE
FOUNDATION OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF
 * USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY,
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY OUT
 * OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF
 * SUCH DAMAGE.
 *
================================================================
=======
 *
 * This software consists of voluntary contributions made by many
 * individuals on behalf of the Apache Software Foundation.  For more
 * information on the Apache Software Foundation, please see
 * <http://www.apache.org/>.
 *
 * Portions of this software are based upon public domain software
 * originally written at the National Center for Supercomputing Applications,
 * University of Illinois, Urbana-Champaign.
 */

/*
 * TPCC Module.  Web server response handler for tpcc transactions
 *
 * <Location /tpcc.dll>
 * SetHandler tpcc
 * </Location>
 *
 *
 *
 * 11.01.2003 Initial Version
 *
 */

#define CORE_PRIVATE

#include "httpd.h"
#include "http_config.h"
#include "http_core.h"
#include "http_log.h"
#include "http_main.h"
#include "http_protocol.h"
#include "http_request.h"
#include "util_script.h"
#include "apr_strings.h"
#include "apr_lib.h"
#define APR_WANT_STRFUNC
#include "apr_want.h"
#include "ap_mpm.h"

#ifdef TUX
#include <tmenv.h>
#include <xa.h>
#include <atmi.h>
#include <Uunix.h>
```

```
#endif

#include "mod_tpcc.h"
#include "tpcc_info.h"
#include "tpccapi.h"

extern TERM *Term;
extern char szTpccLogPath[]; //path to html log file if logging turned on in registry.
extern char szErrorLogPath[];       //path to error log file.
extern BOOL    bLog ;
extern BOOL    dLog ;
extern BOOL    bGeneric;
extern int           iThreads ;
extern int           iMaxWareHouses;
extern int           iDelayMs  ;
extern short   iDeadlockRetry ;
extern short   iMaxConnections;
extern int     bTpccExit;      //exit delivery disconnect loop as dll exiting.
#define TPCC_MOD_CONFIG_KEY "tpcc_module"


module AP_MODULE_DECLARE_DATA tpcc_module;

tpcc_shm_t shmTerm ={"/tmp/tpcc_term",NULL,sizeof(TERM)};
tpcc_shm_t shmCData ={"/tmp/tpcc_cdata",NULL,sizeof(CLIENTDATA)*32};

loaded_t loaded;

tpcc_mod_conf* tpcc_config_global_create(server_rec *s)
{
        apr_pool_t *pool= s->process->pool;
        tpcc_mod_conf *mc;
        apr_pool_userdata_get((void **)&mc, TPCC_MOD_CONFIG_KEY, pool);
        if (mc)
        {
                return mc;
        }

        mc = (tpcc_mod_conf*) apr_pcalloc(pool, sizeof(*mc));
        mc->cs = NULL;
        mc->pool =pool;
        // initialize shm data
        memcpy(&(mc->shmTerm),&shmTerm,sizeof(shmTerm));
        memcpy(&(mc->shmCData),&shmCData,sizeof(shmCData));


    // initialize global storage
      bLog      = FALSE;
      dLog      = FALSE;
      bGeneric  = FALSE;
      iMaxWareHouses    = 625;
      iThreads          = 5;
      iDelayMs          = 100;
      iDeadlockRetry    = (short)3;
      strcpy(szTpccLogPath, "tpcclog.");

//          InitializeCriticalSection(&CriticalSection);
//          InitializeCriticalSection(&ErrorLogCriticalSection);

        apr_pool_userdata_set(mc, TPCC_MOD_CONFIG_KEY,
                              apr_pool_cleanup_null,
                              pool);
        return mc;
}

static void *create_tpcc_config(apr_pool_t *p, server_rec *s)
{
   int i;
   info_svr_conf *conf = (info_svr_conf *) apr_pcalloc(p, sizeof(info_svr_conf));

   conf->mc = tpcc_config_global_create(s);

   return conf;
}

static void *merge_tpcc_config(apr_pool_t *p, void *basev, void *overridesv)
{
/*   no merging of configuration
   info_svr_conf *new = (info_svr_conf *) apr_pcalloc(p, sizeof(info_svr_conf));
   info_svr_conf *base = (info_svr_conf *) basev;
   info_svr_conf *overrides = (info_svr_conf *) overridesv;


   return new;
*/
```

```
    return basev;
}

static int tpcc_handler(request_rec *r)
{
    ext_ctrl_block* ecb = NULL;
    module *modp = NULL;
    const char *more_info;
    const command_rec *cmd = NULL;
#ifdef NEVERMORE
    const handler_rec *hand = NULL;
#endif
    server_rec *serv = r->server;
    int comma = 0;
    tpcc_mod_conf* mc;

    if (strcmp(r->handler, "tpcc"))
        return DECLINED;

    r->allowed |= (AP_METHOD_BIT << M_GET);
    if (r->method_number != M_GET)
            return DECLINED;


    ecb = (ext_ctrl_block*)apr_palloc ( r->pool, sizeof(ext_ctrl_block));
    if (ecb)
    {  // construct a lpszQueryString for isapi code
            char *aprQueryString;
            int len = strlen(r->uri)+ (r->args ? strlen(r->args)+3:1);

            ecb->r=r;
            aprQueryString = (char*)apr_palloc(r->pool, len);
            if (aprQueryString)
            {
            strcpy(aprQueryString, r->uri);
                if (r->args)
                {
                        strcat(aprQueryString, "?");
                        strcat(aprQueryString, r->args);
                }
                ecb->lpszQueryString = aprQueryString;
            }
            else
                {
                        return 0;
                }
    }
    else
    {
                return 0;
    }

    mc = myModConfig(r->server);
    Term = (TERM*) apr_shm_baseaddr_get(mc->shmTerm.shm);
    Term->pClientData = (PCLIENTDATA) apr_shm_baseaddr_get(mc-
>shmCData.shm);

     HttpExtensionProc(ecb);

    /* Done, turn off timeout, close file and return */
    return 0;
}

static const char *set_log_path(cmd_parms *cmd, void *dummy, const char *arg)
{
            strcpy(szTpccLogPath, arg);
                    strcat(szTpccLogPath, "tpcclog");
                strcpy(szErrorLogPath, arg);
            strcat(szErrorLogPath, "tpccerr");


    return NULL;
}
static const char *set_shm_path(cmd_parms *cmd, void *dummy, const char *arg)
{

            tpcc_mod_conf* mc = myModConfig(cmd->server);
            mc->shmTerm.name= apr_psprintf(mc->pool,"%stpcc_term.%lu",
                                arg,(unsigned long) getpid());
            mc->shmCData.name= apr_psprintf(mc->pool,"%stpcc_cdata.%lu",
                                arg,(unsigned long) getpid());
            mc->szMutexFile= apr_psprintf(mc->pool, "%stpcc_mtx.%lu",
                                arg,(unsigned long) getpid());

            return NULL;
}
```

```
static const char *set_log(cmd_parms *cmd, void *dummy, const char *arg)
{

if ( !stricmp(arg, "ON") )
                bLog = TRUE;
   return NULL;
}

static const char *set_debug(cmd_parms *cmd, void *dummy, const char *arg)
{

if ( !stricmp(arg, "ON") )
                dLog = TRUE;
   return NULL;
}

static const char *set_max_warehouses(cmd_parms *cmd, void *dummy,
                                        const char *arg)
{
          iMaxWareHouses = atoi(arg);
                if ( iMaxWareHouses == 0 )
                        iMaxWareHouses = 500;

                        return NULL;
}
static const char *set_del_threads(cmd_parms *cmd, void *dummy,
                                        const char *arg)
{
        iThreads = atoi(arg);
                if ( !iThreads )
                                iThreads = 5;

                        return NULL;
}
static const char *set_backoff_delay(cmd_parms *cmd, void *dummy,
                                        const char *arg)
{
              iDelayMs = atoi(arg);
                  if ( !iDelayMs )
                                iDelayMs = 100;

                        return NULL;
}

static const char *set_deadlock_retry(cmd_parms *cmd, void *dummy,
                                        const char *arg)
{
                iDeadlockRetry = (short)atoi(arg);
                        if ( !iDeadlockRetry )
                                        iDeadlockRetry = (short)3;

                        return NULL;
}

static const char *set_max_connections(cmd_parms *cmd, void *dummy,
                                        const char *arg)
{
              iMaxConnections = (short)atoi(arg);
                  if ( !iMaxConnections )
                                iMaxConnections = (short)25;

                        return NULL;
}

static const char *set_gen_srvc(cmd_parms *cmd, void *dummy,
                                        const char *arg)
{
    if ( !stricmp(arg, "ON") )
                bGeneric = TRUE;

                        return NULL;
}


static const command_rec info_cmds[] =
{
   AP_INIT_TAKE1("Path", set_log_path, NULL, RSRC_CONF,
                        "full path for tpcclog and tpccerr log files"),
   AP_INIT_TAKE1("Shm_Path", set_shm_path, NULL, RSRC_CONF,
                        "full path for share memory file"),
   AP_INIT_TAKE1("Log", set_log, NULL, RSRC_CONF,
                        "Enable error log"),
   AP_INIT_TAKE1("Debug", set_debug, NULL, RSRC_CONF,
                        "Enable debug log"),
   AP_INIT_TAKE1("MaximumWarehouses", set_max_warehouses, NULL,
```

```
RSRC_CONF,
                        "Maximum number of warehouses allowed"),
   AP_INIT_TAKE1("NumberOfDeliveryThreads", set_del_threads, NULL,
RSRC_CONF,
                        "Number of delivery threads"),
   AP_INIT_TAKE1("BackoffDelay", set_backoff_delay, NULL, RSRC_CONF,
                        "Backoff delay"),
   AP_INIT_TAKE1("DeadlockRetry", set_deadlock_retry, NULL, RSRC_CONF,
                        "Number deadlock retry"),
   AP_INIT_TAKE1("MaxConnections", set_max_connections, NULL,
RSRC_CONF,
                        "Maximum connections"),
   AP_INIT_TAKE1("GenSrvc", set_gen_srvc, NULL, RSRC_CONF,
                        "Generic service (for TOPEND only)"),
   {NULL}
};

apr_status_t tpcc_init_modulekill(void *data)
{
        tpcc_mod_conf *mc;
        server_rec *base_server = (server_rec *)data;
        server_rec *s;

        mc = myModConfig(base_server);

        bTpccExit = TRUE;

        if (mc && mc->cs)
        {
                TERM* Term;
                Term = apr_shm_baseaddr_get(mc->shmTerm.shm);
                if (dLog)
                {
                        FILE *fp;

                        fp = fopen("/tmp/errlog", "ab");
                        fprintf(fp, "modulekill, pid=%d,
thread_id=%d,iAvail=%d\n", getpid(),GetCurrentThreadId(),Term->iAvailable);

                        fclose(fp);
                }

                apr_shm_destroy(mc->shmTerm.shm);

                apr_shm_destroy(mc->shmCData.shm);

                unlink(mc->shmTerm.name);
                unlink(mc->shmCData.name);

                apr_global_mutex_destroy(mc->cs);
                mc->cs=NULL;
        }
                if (dLog)
                {
                        FILE *fp;

                        fp = fopen("/tmp/errlog", "ab");
                        fprintf(fp, "aftermodulekill, pid=%d, thread_id=%d\n",
getpid(),GetCurrentThreadId());

                        fclose(fp);
                }
   return APR_SUCCESS;
}

int tpcc_init_module(apr_pool_t *p, apr_pool_t *plog,
                apr_pool_t *ptemp,
                        server_rec *base_server)
{
        tpcc_mod_conf* mc = myModConfig(base_server);
                /*
                 *      * Let us cleanup on restarts and exists
                 *          */
                apr_pool_cleanup_register(p, base_server,
                                tpcc_init_modulekill,
                                        apr_pool_cleanup_null);
                        if (dLog)
                                {
                                FILE *fp;

                                fp = fopen("/tmp/errlog", "ab");
                                fprintf(fp, "*init Moudle pid=%d, thread_id=%d\n",
getpid(),GetCurrentThreadId());

                                fclose(fp);
                        }
```

```
        if (!mc->cs)
        {
                TERM* Term;
                if (dLog)
                {
                        FILE *fp;

                        fp = fopen("/tmp/errlog", "ab");
                        fprintf(fp, "mccs *init Module pid=%d, thread_id=%d\n",
getpid(),GetCurrentThreadId());

                        fclose(fp);
                }
                tpcc_shm_create(&(mc->shmTerm),mc->pool);   // Term
                tpcc_shm_create(&(mc->shmCData),mc->pool);   //ClientData

                apr_global_mutex_create(&(mc->cs),"/tmp/tpcc_mtx"/*mc-
>szMutexFile*/,APR_LOCK_DEFAULT,mc->pool);
                if (dLog)
                {
                        FILE *fp;

                        fp = fopen("/tmp/errlog", "ab");
                        fprintf(fp, "before get  *init Moudle pid=%d,
thread_id=%d\n", getpid(),GetCurrentThreadId());

                        fclose(fp);
                }
                Term = (TERM*) apr_shm_baseaddr_get(mc->shmTerm.shm);
                if (Term)
                {

                int i;
                Term->iAvailable=0;
                Term->iNext=0;
                Term->iMasterSyncId=0;
                Term->bInit=FALSE;
                Term->pClientData=NULL;

                TermInit(base_server);
                if ( !TermAllocate(base_server) )
                {
                        FILE *fp;

                        fp = fopen("/tmp/errlog", "ab");
                        fprintf(fp, "Error Trm.Allocate Init, pid=%d,
thread_id=%d\n", getpid(),GetCurrentThreadId());

                        fclose(fp);
                }
                //Term = (TERM*) apr_shm_baseaddr_get(mc->shmTerm.shm);

                for(i=Term->iNext; i<Term->iAvailable; i++)
                  Term->pClientData[i].inUse = 0;
                Term->pClientData[0].inUse = 1;

                }
        }

        return APR_SUCCESS;

}

static int tpcc_pre_config(apr_pool_t* pconf, apr_pool_t *plog, apr_pool_t *ptemp)
{
        apr_status_t rv;
        apr_pool_sub_make(&loaded.pool, pconf, NULL);
                if (dLog)
                {
                        FILE *fp;

                        fp = fopen("/tmp/errlog", "ab");
                        fprintf(fp, "*init Moudle pid=%d, thread_id=%d\n",
getpid(),GetCurrentThreadId());

                        fclose(fp);
                }

  if (!loaded.pool) {
    ap_log_error(APLOG_MARK, APLOG_ERR, APR_EGENERAL, NULL,
                "tpcc: could not create the configuration pool");
    return APR_EGENERAL;
  }

  loaded.hash = apr_hash_make(loaded.pool);
```

```
    if (!loaded.hash) {
        ap_log_error(APLOG_MARK, APLOG_ERR, 0, NULL,
                "tpcc: Failed to create tls cache");
        return APR_EGENERAL;
    }

    rv = apr_thread_mutex_create(&loaded.lock,
APR_THREAD_MUTEX_DEFAULT,
                            loaded.pool);
    if (rv != APR_SUCCESS) {
        ap_log_error(APLOG_MARK, rv, 0, NULL,
                "ISAPI: Failed to create module cache lock");
        return rv;
    }
   return OK;
}

static void register_hooks(apr_pool_t *p)
{
    ap_hook_pre_config(tpcc_pre_config, NULL,NULL, APR_HOOK_MIDDLE);
    ap_hook_handler(tpcc_handler, NULL, NULL, APR_HOOK_MIDDLE);
    ap_hook_post_config(tpcc_init_module, NULL,NULL,APR_HOOK_MIDDLE);
}

module AP_MODULE_DECLARE_DATA tpcc_module =
{
    STANDARD20_MODULE_STUFF,
    NULL,                   /* dir config creater */
    NULL,                   /* dir merger --- default is to override */
    create_tpcc_config,         /* server config */
    merge_tpcc_config,          /* merge server config */
    info_cmds,                /* command apr_table_t */
    register_hooks
};
```

mod_tpcc.h

```
#ifndef MOD_TPCC
#define MOD_TPCC

#define CORE_PRIVATE

#include "httpd.h"
#include "http_config.h"
#include "http_core.h"
#include "http_log.h"
#include "http_main.h"
#include "http_protocol.h"
#include "http_request.h"
#include "util_script.h"
#include "apr_strings.h"
#include "apr_lib.h"
#define APR_WANT_STRFUNC
#include "apr_want.h"
#include "ap_mpm.h"

// map WIN32/IIS API types to apache types
typedef unsigned int DWORD;
typedef char*       LPTSTR;

#define GetCurrentThreadId()        apr_os_thread_current()
#define TlsAlloc(x)          (NULL)

#define HSE_STATUS_SUCCESS_AND_KEEP_CONN       1
#define HSE_STATUS_SUCCESS 0
#define TEXT(x)                         x

#define CRITICAL_SECTION              apr_global_mutex_t *

#define mySrvConfig(srv) (info_svr_conf*)ap_get_module_config(srv-
>module_config, &tpcc_module)
#define myModConfig(srv) (mySrvConfig((srv)))->mc

#define DestroyCriticalSection(x)       apr_global_mutex_destroy(*(x))

#define InitializeCriticalSection(x)

#define mcInitializeCriticalSection(x)


#define mcEnterCriticalSection(x)               \
{\
            apr_thread_mutex_lock(x);\
}
```

```
#define mcLeaveCriticalSection(x)   \
{\
        apr_thread_mutex_unlock(x);\
}


#define EnterCriticalSection(ax)                \
{\
        tpcc_mod_conf* mc = (tpcc_mod_conf *)myModConfig(pECB->r-
>server);\
        apr_global_mutex_lock(mc->cs);\
        Term = apr_shm_baseaddr_get(mc->shmTerm.shm);\
        Term->pClientData = (PCLIENTDATA)apr_shm_baseaddr_get(mc-
>shmCData.shm);\
}

#define LeaveCriticalSection(ax)   \
{\
        tpcc_mod_conf* mc = (tpcc_mod_conf *)myModConfig(pECB->r-
>server);\
        apr_global_mutex_unlock(mc->cs);\
}

#define SET_TERM_AND_CDATA          \
{\
        tpcc_mod_conf* mc = (tpcc_mod_conf *)myModConfig(pECB->r-
>server);\
        Term = apr_shm_baseaddr_get(mc->shmTerm.shm);\
        Term->pClientData = (PCLIENTDATA)apr_shm_baseaddr_get(mc-
>shmCData.shm);\
}


#define NO_ERROR        0
#define GetLastError()      NO_ERROR

#define wsprintf  sprintf
#define _ftime              ftime
#define stricmp             strcasecmp
#define Sleep(x)  usleep((x)*1000)
#define GetTickCount()    apr_time_as_msec(apr_time_now())

#define WriteMessageToEventLog(x)                   write_err_log(pECB->r, x)
#define WR_LOG(x)       write_err_log(pECB->r,x)
#define TPCC_LOG(x)     {           \
                        FILE *fp;\
                        fp = fopen(szTpccLogPath, "ab");\
                        fprintf(fp, x" Thread %d pid= %d\n", \
                                GetCurrentThreadId(),getpid() );\
                        fclose(fp);\
                }

typedef struct                          /* simulate ECB in apache */
{
  request_rec *r;
  char * lpszQueryString;
} ext_ctrl_block;

typedef struct
{
        pid_t pid;
        apr_os_thread_t tid;
} tls_t;

typedef struct
{
        char *name;
        apr_shm_t *shm;
        apr_size_t size;
        void *addr;
}
tpcc_shm_t;
typedef struct {
   apr_global_mutex_t *cs;
   tpcc_shm_t shmTerm;
   tpcc_shm_t shmCData;
   apr_pool_t*      pool;
   char* szMutexFile;
} tpcc_mod_conf;


typedef struct {
    apr_array_header_t *more_info;
    tpcc_mod_conf* mc;
}  info_svr_conf;
```

```
extern module AP_MODULE_DECLARE_DATA tpcc_module;

typedef struct tpcc_global_conf {
   apr_pool_t        *pool;
   apr_thread_mutex_t *lock;
   apr_hash_t        *hash;
} loaded_t;

extern loaded_t loaded;
#endif
```

Makefile.am

```
## This is the shared library to be built
lib_LTLIBRARIES = libmodtpcc.la

## Define the source file for the module
libmodtpcc_la_SOURCES = mod_tpcc.c tpcc.c

## Define that an include directory is required.
INCLUDES = -DTUX -I@apache_dir@/include  -I../tm_src -I${TUXDIR}/include

LDFLAGS = -L@apache_dir@/lib -lapr-0 -L${TUXDIR}/lib -ltux -lcobatmi -lgw -
lnative -ltmib
```

build.sh

```
make
/usr/local/apache2/bin/apxs -i -a -n tpcc libmodtpcc.la
```

configure.in

```
# Required initializer
AC_INIT

# Automake initialization
AM_INIT_AUTOMAKE(mod_tpcc, 1.0)

# Add a test for a compiler.
AC_PROG_CC
AM_PROG_LIBTOOL

# Define a macro that is used to parse a --with-apache parameter
# The macro is named "APACHE_DIR"
AC_DEFUN([APACHE_DIR],[

        AC_ARG_WITH(
                apache,
                [  --with-apache[=DIR]     Apache server directory],
                ,
                [with_apache="no"]
        )

        AC_MSG_CHECKING(for Apache directory)

        if test "$with_apache" = "no"; then
                AC_MSG_ERROR( You need to specify the apache directory using
--with-apache)
        else
                # make sure that a well known include file exists
                if test -e $with_apache/include/httpd.h; then
                        apache_dir=$with_apache
                        AC_MSG_RESULT(APACHE found!)
                else
                        AC_MSG_ERROR( $with_apache not found.  Check the
value you specified with --with-apache)
                fi
        fi

])

# Now call the APACHE_DIR macro that was just specified
APACHE_DIR

# Save the location of apache into the "apache_dir" variable.
# The AC_SUBST macro causes the variable to be saved in config.status
AC_SUBST(apache_dir)

# Write config.status and the Makefile
AC_OUTPUT(Makefile)
```

A2. Back-End Source Code, Scripts

pldel.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: pldel.c 7030100.5 96/06/24 16:26:06 plai Generic<base> $ Copyr (c)
1994 Oracle";
#endif /* RCSID */

/*=============================================================
======+
|       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
|                  OPEN SYSTEMS PERFORMANCE GROUP                |
|                      All Rights Reserved                       |

+=============================================================
======+
| FILENAME
|    pldel.c
| DESCRIPTION
|    OCI version of DELIVERY transaction in TPC-C benchmark.

+=============================================================
======*/

#include "tpcc.h"
#ifdef TUX
#include <userlog.h>
#endif

/*
extern int userlog();
*/

#define DMLRETDEL

#define SQLTXT "BEGIN inittpcc.init_del ; END;"

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

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

#define SQLTXT4 "UPDATE 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 SQLTXT6 "UPDATE cust SET c_balance = c_balance + :amt, \
   c_delivery_cnt = c_delivery_cnt + 1 WHERE c_w_id = :w_id AND \
   c_d_id = :d_id AND c_id = :c_id"

#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
  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;

static pldelctx *pldctx;


static delctx *dctx;

#ifdef DMLRETDEL
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;
amtctx *actx;

#endif


#ifdef DMLRETDEL
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)
{
  *bufpp = &dctx->del_o_id[iter];
  *indpp= &dctx->del_o_id_ind[iter];
  dctx->del_o_id_len[iter]=sizeof(dctx->del_o_id[0]);
  *alenp= &dctx->del_o_id_len[iter];
  *rcodepp = &dctx->del_o_id_rcode[iter];
  *piecep =OCI_ONE_PIECE;
  return (OCI_CONTINUE);
}
sb4 cid_data(dvoid *ctxp, OCIBind *bp, ub4 iter, ub4 index,
            dvoid **bufpp, ub4 **alenp, ub1 *piecep,
            dvoid **indpp, ub2 **rcodepp)
{
  *bufpp = &dctx->c_id[iter];
  *indpp= &dctx->c_id_ind[iter];
  dctx->c_id_len[iter]=sizeof(dctx->c_id[0]);
  *alenp= &dctx->c_id_len[iter];
  *rcodepp = &dctx->c_id_rcode[iter];
  *piecep =OCI_ONE_PIECE;
  return (OCI_CONTINUE);
}


# ifdef OLD
sb4 amt_data(dvoid *ctxp, OCIBind *bp, ub4 iter, ub4 index,
            dvoid **bufpp, ub4 **alenp, ub1 *piecep,
            dvoid **indpp, ub2 **rcodepp)
{
```

```
  amtctx *actx;
  actx =(amtctx*)ctxp;
  actx->ol_cnt=actx->ol_cnt+1;
  *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;
  if (iter ==1 )
    return (OCI_CONTINUE);
  else
    return (OCI_ERROR);
}
# else
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);
}
# endif

#endif


tkvcdinit (int plsqlflag)

{

   text stmbuf[SQL_BUF_SIZE];

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



     DISCARD OCIHandleAlloc(tpcenv,(dvoid**) &pldctx->curp2,
OCI_HTYPE_STMT,
                 0, (dvoid**)0);
#if defined(ISO5) || defined(ISO6) || defined(ISO8)
  #if defined(ISO5)
     sqlfile("../blocks/tkvcpdel_iso5.sql",stmbuf);
  #endif
  #if defined(ISO6)
     sqlfile("../blocks/tkvcpdel_iso6.sql",stmbuf);
  #endif
  #if defined(ISO8)
     sqlfile("../blocks/tkvcpdel_iso8.sql",stmbuf);
  #endif
#else
     sqlfile("../blocks/tkvcpdel.sql",stmbuf);
#endif
     DISCARD OCIStmtPrepare(pldctx->curp2, errhp, stmbuf,
                  (ub4)strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT);
     OCIBNDPL(pldctx->curp2, pldctx->w_id_bp , errhp,":w_id",
           ADR(w_id), SIZ(int), SQLT_INT,&pldctx->w_id_len);
     OCIBNDPL(pldctx->curp2, pldctx->ordcnt_bp , errhp,":ordcnt",
           ADR(pldctx->ordcnt), SIZ(int), SQLT_INT,&pldctx->ordcnt_len);
     OCIBNDPL(pldctx->curp2, pldctx->del_date_bp,errhp,":now",
           ADR(pldctx->del_date), SIZ(OCIDate), SQLT_ODT,&pldctx-
>del_date_len);
     OCIBNDPL(pldctx->curp2, pldctx->carrier_id_bp , errhp,
           ":carrier_id", ADR(o_carrier_id), SIZ(int),
           SQLT_INT, &pldctx->carrier_id_len);
```

```
    OCIBNDPLA(pldctx->curp2, pldctx->d_id_bp, errhp,":d_id",
            pldctx->del_d_id, SIZ(int),SQLT_INT, pldctx->del_d_id_len,
            NDISTS, &pldctx->del_d_id_rcnt);
    OCIBNDPLA(pldctx->curp2, pldctx->o_id_bp, errhp,":order_id",
            pldctx->del_o_id,SIZ(int),SQLT_INT, pldctx->del_o_id_len,NDISTS,
            &pldctx->del_o_id_rcnt);
#ifdef USE_IEEE_NUMBER
    OCIBNDPLA(pldctx->curp2, pldctx->sums_bp, errhp,":sums",
            pldctx->sums,SIZ(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 = (delctx *) malloc (sizeof(delctx));
    memset(dctx,(char)0,sizeof(delctx));
    dctx->norow = 0;
    actx = (amtctx *) malloc (sizeof(amtctx));
    memset(actx,(char)0,sizeof(amtctx));

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

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

  /* open third cursor */

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


  /* bind variables */

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

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

  /* open fourth cursor */

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

  /* bind variables */

    OCIBND(dctx->curd4, dctx->w_id_bp4,errhp,":w_id",dctx->w_id,
         SIZ(int), SQLT_INT);
    OCIBND(dctx->curd4, dctx->d_id_bp4,errhp,":d_id",dctx->d_id,
```

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



  /* open sixth cursor */

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

  /* bind variables */


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

}


void shiftdata(from)
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];
  }
}


tkvcd (int plsqlflag)

{

  int i, j;
  int rpc,rcount,count;
  int invalid;

  if (plsqlflag)
  {

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

    pldctx->retry=0;

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

```
  {


retry:

    invalid = 0;

  /* initialization for array operations */

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

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

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

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

  /* array select from new_order and orders tables */

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

    execstatus=OCIStmtExecute(tpcsvc,dctx->curd3,errhp,rpc,0,
            NULLP(CONST OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
    if(execstatus != OCI_SUCCESS)
    {
     DISCARD OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
     errcode = OCIERROR(errhp,execstatus);
     if(errcode == NOT_SERIALIZABLE)
     {
```

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

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

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

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

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

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

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

    if (rcount != rpc) {
#ifdef TUX
      userlog ("Error in TPC-C server %d: %d rows selected, %d cust updated\n",
            proc_no, rpc, rcount);
#else
      DISCARD fprintf (stderr,
            "Error in TPC-C server %d: %d rows selected, %d cust updated\n",
            proc_no, rpc, rcount);
#endif
      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);

}

void tkvcddone (int plsqlflag)

{

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

}
```

plnew.c

```
#ifdef RCSID
static char *RCSid =
```

```
   "$Header: tkvcnew.c 21-apr-98.18:32:59 rdecker Exp $ Copyr (c) 1994 Oracle";
#endif /* RCSID */

/*============================================================
======+
|       Copyright (c) 1996 , 1997, 1998 Oracle Corp, Redwood Shores, CA       |
|                  OPEN SYSTEMS PERFORMANCE GROUP                |
|                      All Rights Reserved                       |

+============================================================
======+
| FILENAME
|    plnew.c
| DESCRIPTION
|    OCI version (using PL/SQL stored procedure) of
|    NEW ORDER transaction in TPC-C benchmark.

+============================================================
======*/

#include "tpcc.h"

#ifdef TUX
#include <userlog.h>
#endif

#define SQLTXT2 "BEGIN inittpcc.init_no(:idx1arr); END;"

#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];
  OCIStmt *curn1;
  OCIBind *ol_i_id_bp;
  OCIBind *ol_supply_w_id_bp;
  OCIBind *i_price_bp;
  OCIBind *i_name_bp;
  OCIBind *s_bg_bp;
  ub4 nol_i_count;
  ub4 nol_s_count;
  ub4 nol_q_count;
  ub4 nol_item_count;
  ub4 nol_name_count;
  ub4 nol_qty_count;
  ub4 nol_bg_count;
  ub4 nol_am_count;
  ub4 s_remote_count;
  OCIStmt *curn2;
  OCIBind *ol_quantity_bp;
  OCIBind *s_remote_bp;
  OCIBind *s_quantity_bp;
  OCIBind *w_id_bp;
  OCIBind *d_id_bp;
  OCIBind *c_id_bp;
  OCIBind *o_all_local_bp;
  OCIBind *o_all_cnt_bp;
  OCIBind *w_tax_bp;
  OCIBind *d_tax_bp;
  OCIBind *o_id_bp;
  OCIBind *c_discount_bp;
```

```
  OCIBind *c_credit_bp;
  OCIBind *c_last_bp;
  OCIBind *retries_bp;
  OCIBind *cr_date_bp;
  OCIBind *ol_o_id_bp;
  OCIBind *ol_amount_bp;

  sb2 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;

static newctx *nctx;

tkvcninit ()
{

  int i;
  text stmbuf[32*1024];

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

  /* open first cursor */
  DISCARD OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&nctx->curn1),
        OCI_HTYPE_STMT, 0, (dvoid**)0));
#if defined(ISO)
  sqlfile("../blocks/tkvcpnew_iso.sql",stmbuf);
#else
#if defined(ISO7)
  sqlfile("../blocks/tkvcpnew_iso7.sql",stmbuf);
#else
  sqlfile("../blocks/tkvcpnew.sql",stmbuf);
#endif
#endif

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

  /* bind variables */

  OCIBNDPL(nctx->curn1, nctx->w_id_bp, errhp, ":w_id",ADR(w_id),SIZ(w_id),
        SQLT_INT, &nctx->w_id_len);
  OCIBNDPL(nctx->curn1, nctx->d_id_bp, errhp, ":d_id",ADR(d_id),SIZ(d_id),
        SQLT_INT, &nctx->d_id_len);
  OCIBNDPL(nctx->curn1, nctx->c_id_bp, errhp, ":c_id",ADR(c_id),SIZ(c_id),
        SQLT_INT, &nctx->c_id_len);
  OCIBNDPL(nctx->curn1, nctx->o_all_local_bp, errhp, ":o_all_local",
        ADR(o_all_local), SIZ(o_all_local),SQLT_INT, &nctx->o_all_local_len);
  OCIBNDPL(nctx->curn1, nctx->o_all_cnt_bp, errhp, ":o_ol_cnt",ADR(o_ol_cnt),
        SIZ(o_ol_cnt),SQLT_INT, &nctx->o_ol_cnt_len);
  OCIBNDPL(nctx->curn1, nctx->w_tax_bp, errhp,
":w_tax",ADR(w_tax),SIZ(w_tax),
        SQLT_FLT, &nctx->w_tax_len);
  OCIBNDPL(nctx->curn1, nctx->d_tax_bp, errhp, ":d_tax",ADR(d_tax),SIZ(d_tax),
        SQLT_FLT, &nctx->d_tax_len);
  OCIBNDPL(nctx->curn1, nctx->o_id_bp, errhp, ":o_id",ADR(o_id),SIZ(o_id),
        SQLT_INT, &nctx->o_id_len);
  OCIBNDPL(nctx->curn1, nctx->c_discount_bp, errhp, ":c_discount",
        ADR(c_discount), SIZ(c_discount),SQLT_FLT, &nctx->c_discount_len);
  OCIBNDPL(nctx->curn1, nctx->c_credit_bp, errhp, ":c_credit",c_credit,
```

```
        SIZ(c_credit),SQLT_CHR, &nctx->c_credit_len);
  OCIBNDPL(nctx->curn1, nctx->c_last_bp, errhp, ":c_last",c_last,SIZ(c_last),
        SQLT_STR, &nctx->c_last_len);
  OCIBNDPL(nctx->curn1, nctx->retries_bp, errhp, ":retry",ADR(retries),
        SIZ(retries),SQLT_INT, &nctx->retries_len);
  OCIBNDPL(nctx->curn1, nctx->cr_date_bp, errhp, ":cr_date",&cr_date,
        SIZ(OCIDate), SQLT_ODT, &nctx->cr_date_len);

  OCIBNDPLA(nctx->curn1, nctx->ol_i_id_bp,errhp,":ol_i_id",nol_i_id,
         SIZ(int), SQLT_INT, nctx->nol_i_id_len,NITEMS,&nctx->nol_i_count);
  OCIBNDPLA(nctx->curn1, nctx->ol_supply_w_id_bp, errhp, ":ol_supply_w_id",
         nol_supply_w_id,SIZ(int),SQLT_INT, nctx->nol_supply_w_id_len,
         NITEMS, &nctx->nol_s_count);
#ifdef USE_IEEE_NUMBER
  OCIBNDPLA(nctx->curn1, nctx->ol_quantity_bp,errhp,":ol_quantity",
         nol_quantity, SIZ(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, SQLTXT2);
  DISCARD OCIERROR(errhp,OCIStmtPrepare(nctx->curn2, errhp, stmbuf,
              strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));


  /* execute second cursor to init newinit package */
  {
    int idx1arr[NITEMS];
    OCIBind *idx1arr_bp;
    ub2 idx1arr_len[NITEMS];
    ub2 idx1arr_rcode[NITEMS];
    sb2 idx1arr_ind[NITEMS];
    ub4 idx1arr_count;
    ub2 idx;

    for (idx = 0; idx < NITEMS; idx++) {
      idx1arr[idx] = idx + 1;
      idx1arr_ind[idx] = TRUE;
      idx1arr_len[idx] = sizeof(int);
    }
    idx1arr_count = NITEMS;
          o_ol_cnt = NITEMS;


    /* Bind array */
```

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

    execstatus = OCIStmtExecute(tpcsvc,nctx->curn2,errhp,1,0,
            NULLP(CONST OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
    if(execstatus != OCI_SUCCESS) {
     OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
     errcode = OCIERROR(errhp,execstatus);
     return -1;
    }
  }

 return (0);

}



tkvcn ()
{

  int i;
  int rcount;

retry:

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

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

  o_ol_cnt = NITEMS;
  o_all_local = 1;
  for (i = 0; i < NITEMS; i++) {
    if (nol_i_id[i] == 0) {
      o_ol_cnt = i;
      break;
    }
    if (nol_supply_w_id[i] != w_id) {
#ifdef USE_IEEE_NUMBER
      nctx->s_remote[i] = 1.0;
#else
      nctx->s_remote[i] = 1;
#endif /* USE_IEEE_NUMBER */
      o_all_local = 0;
    }
    else
      nctx->s_remote[i] = 0;
  }

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

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

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

```
    nctx->s_quant_len[i] = sizeof(int);
    nctx->i_name_len[i]=0;
    nctx->s_bg_len[i] = 0;
  }
  for (i = o_ol_cnt; i < NITEMS; i++) {

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

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


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

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

  #ifdef DEBUG
    printf("w_id = %d, d_id = %d, c_id = %d\n",w_id, d_id, c_id);
  #endif

    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);
  }



  void tkvcndone ()

  {

    int i;

    if (nctx)
    {
     DISCARD OCIHandleFree((dvoid *)nctx->curn1,OCI_HTYPE_STMT);
     DISCARD OCIHandleFree((dvoid *)nctx->curn2,OCI_HTYPE_STMT);
     free (nctx);
    }
  }
```

plord.c

```
/* Copyright (c) 2002, Oracle Corporation.  All rights reserved.  */

/*

   NAME
     tkvcordq.c - OCI version  using queues of ORDER STATUS
                  transaction in TPC-C benchmark.
```

```
  DESCRIPTION
    <short description of facility this file declares/defines>

  EXPORT FUNCTION(S)

  INTERNAL FUNCTION(S)
    <other external functions defined - one-line descriptions>

  STATIC FUNCTION(S)
    <static functions defined - one-line descriptions>

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  xnie       06/25/02 - queue open cluster join.
  heri       05/07/02 - Fix error in cursor.
  heri       02/01/02 - Cleanup, remove indicator values and return codes.
  lwang      07/25/01 - Merged lwang_tpccitrc
  lwang      07/23/01 - fix include
  lwang      07/23/01 - Creation

*/


#include "tpcc.h"

//#define USE_IEEE_NUMBER
/*---------------------------------------------------------------------------
                PRIVATE TYPES AND CONSTANTS
  ---------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------
                STATIC FUNCTION DECLARATIONS
  ---------------------------------------------------------------------------*/



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

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


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

#define SQLCUR3 "SELECT  /*+ INDEX(ordl) */ \
           ol_i_id, ol_supply_w_id, ol_quantity, ol_amount, ol_delivery_d \
           FROM ordl \
           WHERE ol_o_id = :o_id AND  ol_d_id = :d_id AND ol_w_id = :w_id"

#define SQLCUR4 "SELECT count(c_last) FROM cust \
            WHERE c_d_id = :d_id AND c_w_id = :w_id AND c_last = :c_last "


struct ordctx {

  ub2 c_rowid_len[100];
  ub2 ol_supply_w_id_len[NITEMS];
  ub2 ol_i_id_len[NITEMS];
  ub2 ol_quantity_len[NITEMS];
  ub2 ol_amount_len[NITEMS];
  ub2 ol_delivery_d_len[NITEMS];
  ub2 ol_w_id_len;
  ub2 ol_d_id_len;
  ub2 ol_o_id_len;


  ub4 ol_supply_w_id_csize;
  ub4 ol_i_id_csize;
  ub4 ol_quantity_csize;
  ub4 ol_amount_csize;
```

```
  ub4 ol_delivery_d_csize;
  ub4 ol_w_id_csize;
  ub4 ol_d_id_csize;
  ub4 ol_o_id_csize;

  OCIStmt *curo0;
  OCIStmt *curo1;
  OCIStmt *curo2;
  OCIStmt *curo3;
  OCIStmt *curo4;
  OCIBind *c_id_bp;
  OCIBind *w_id_bp[4];
  OCIBind *d_id_bp[4];
  OCIBind *c_last_bp[2];
  OCIBind *o_id_bp;
  OCIBind *c_rowid_bp;
  OCIDefine *c_rowid_dp;
  OCIDefine *c_last_dp[2];
  OCIDefine *c_id_dp;
  OCIDefine *c_first_dp[2];
  OCIDefine *c_middle_dp[2];
  OCIDefine *c_balance_dp[2];
  OCIDefine *o_id_dp[2];
  OCIDefine *o_entry_d_dp[2];
  OCIDefine *o_cr_id_dp[2];
  OCIDefine *o_ol_cnt_dp[2];
  OCIDefine *ol_d_d_dp;
  OCIDefine *ol_i_id_dp;
  OCIDefine *ol_supply_w_id_dp;
  OCIDefine *ol_quantity_dp;
  OCIDefine *ol_amount_dp;
  OCIDefine *ol_d_base_dp;
  OCIDefine *c_count_dp;
  OCIRowid *c_rowid_ptr[100];
  OCIRowid *c_rowid_cust;
  int cs;
  int cust_idx;
  int norow;
  int rcount;
  int somerows;
};

typedef struct ordctx ordctx;

struct defctx
{
 boolean reexec;
 ub4 count;
};
typedef struct defctx defctx;

static ordctx *octx;

static defctx cbctx;


ub2 rlenp_c_last;
ub2 rlenp_c_first;



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

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


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

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

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

/*  c_id == 0, use lastname to find customer */
  DISCARD sprintf((char *) stmbuf, SQLCUR2);
  DISCARD OCIERROR(errhp,
    OCIStmtPrepare(octx->curo2,errhp,stmbuf,(ub4)strlen((char *)stmbuf),
              OCI_NTV_SYNTAX,OCI_DEFAULT));
  DISCARD OCIERROR(errhp,
    OCIAttrSet(octx->curo2,OCI_HTYPE_STMT,&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,errhp));

  DISCARD sprintf((char *) stmbuf, SQLCUR3);
  DISCARD OCIERROR(errhp,
    OCIStmtPrepare(octx->curo3,errhp,stmbuf,(ub4)strlen((char *)stmbuf),
              OCI_NTV_SYNTAX,OCI_DEFAULT));
  DISCARD OCIERROR(errhp,
    OCIAttrSet(octx->curo3,OCI_HTYPE_STMT,&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,errhp));
  DISCARD sprintf((char *) stmbuf, SQLCUR4);
  DISCARD OCIERROR(errhp,
    OCIStmtPrepare(octx->curo4,errhp,stmbuf,(ub4)strlen((char *)stmbuf),
              OCI_NTV_SYNTAX,OCI_DEFAULT));
  DISCARD OCIERROR(errhp,
    OCIAttrSet(octx->curo4,OCI_HTYPE_STMT,&octx->norow,0,
            OCI_ATTR_PREFETCH_ROWS,errhp));

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

    octx->ol_supply_w_id_len[i] = sizeof(int);
    octx->ol_i_id_len[i] = sizeof(int);
    octx->ol_quantity_len[i] = sizeof(int);
    octx->ol_amount_len[i] = sizeof(int);
    octx->ol_delivery_d_len[i] = sizeof(ol_d_base[0]);
  }
  octx->ol_supply_w_id_csize = NITEMS;
  octx->ol_i_id_csize = NITEMS;
  octx->ol_quantity_csize = NITEMS;
  octx->ol_amount_csize = NITEMS;
  octx->ol_delivery_d_csize = NITEMS;
  octx->ol_w_id_csize = NITEMS;
  octx->ol_o_id_csize = NITEMS;
  octx->ol_d_id_csize = NITEMS;
  octx->ol_w_id_len = sizeof(int);
  octx->ol_d_id_len = sizeof(int);
  octx->ol_o_id_len = sizeof(int);

  /* bind variables */

  /* c_id (customer id) is not known */
  OCIBND(octx->curo0,octx->w_id_bp[0],errhp,":w_id",ADR(w_id),
      SIZ(int),SQLT_INT);
  OCIBND(octx->curo0,octx->d_id_bp[0],errhp,":d_id",ADR(d_id),
      SIZ(int),SQLT_INT);
  OCIBND(octx->curo0,octx->c_last_bp[0],errhp,":c_last",c_last,
      SIZ(c_last), SQLT_STR);
  OCIDFNRA(octx->curo0,octx->c_rowid_dp,errhp,1,octx->c_rowid_ptr,
       SIZ(OCIRowid*), SQLT_RDD, NULL, octx->c_rowid_len, NULL);

  OCIBND(octx->curo1,octx->c_rowid_bp,errhp,":cust_rowid", &octx-
>c_rowid_cust,
```

```
        sizeof( octx->c_rowid_ptr[0]),SQLT_RDD);
  OCIDEF(octx->curo1,octx->c_id_dp,errhp,1,ADR(c_id),SIZ(int),SQLT_INT);
#ifdef USE_IEEE_NUMBER
  OCIDEF(octx->curo1,octx->c_balance_dp[0],errhp,2,ADR(c_balance),
        SIZ(double),SQLT_BDOUBLE);
#else
  OCIDEF(octx->curo1,octx->c_balance_dp[0],errhp,2,ADR(c_balance),
        SIZ(double),SQLT_FLT);
#endif /* USE_IEEE_NUMBER */
/*   OCIDEF(octx->curo1,octx->c_first_dp[0],errhp,3,c_first,SIZ(c_first)-1,
        SQLT_CHR); */

  OCIHandleAlloc(octx->curo1, (dvoid**)&(octx->c_first_dp[0]),
       OCI_HTYPE_DEFINE,0, (dvoid**)0);
  OCIDefineByPos(octx->curo1,&(octx->c_first_dp[0]), errhp, 3, c_first,
      SIZ(c_first)-1, SQLT_CHR, NULL, &rlenp_c_first, NULL, OCI_DEFAULT);

  OCIDEF(octx->curo1,octx->c_middle_dp[0],errhp,4,c_middle,
        SIZ(c_middle)-1,SQLT_AFC);
 /*  OCIDEF(octx->curo1,octx->c_last_dp[0],errhp,5,c_last,SIZ(c_last)-1,
        SQLT_CHR); */

  OCIHandleAlloc(octx->curo1, (dvoid**)&(octx->c_last_dp[0]),
       OCI_HTYPE_DEFINE,0, (dvoid**)0);
  OCIDefineByPos(octx->curo1,&(octx->c_last_dp[0]), errhp, 5, c_last,
      SIZ(c_last)-1, SQLT_CHR, NULL, &rlenp_c_last, NULL, OCI_DEFAULT);



  OCIDEF(octx->curo1,octx->o_id_dp[0],errhp,6,ADR(o_id),SIZ(int),SQLT_INT);
  OCIDEF(octx->curo1,octx->o_entry_d_dp[0],errhp,7,
        &o_entry_d_base,SIZ(OCIDate),SQLT_ODT);
  OCIDEF(octx->curo1,octx->o_cr_id_dp[0],errhp,8,ADR(o_carrier_id),
        SIZ(int),SQLT_INT);
  OCIDEF(octx->curo1,octx->o_ol_cnt_dp[0],errhp,9,ADR(o_ol_cnt),
        SIZ(int),SQLT_INT);

/* Bind for third cursor , no-zero customer id */
  OCIBND(octx->curo2,octx->w_id_bp[1],errhp,":w_id",ADR(w_id),
       SIZ(int),SQLT_INT);
  OCIBND(octx->curo2,octx->d_id_bp[1],errhp,":d_id",ADR(d_id),
            SIZ(int),SQLT_INT);
  OCIBND(octx->curo2,octx->c_id_bp,errhp,":c_id",ADR(c_id),
            SIZ(int),SQLT_INT);
#ifdef USE_IEEE_NUMBER
  OCIDEF(octx->curo2,octx->c_balance_dp[1],errhp,1,ADR(c_balance),
        SIZ(double),SQLT_BDOUBLE);
#else
  OCIDEF(octx->curo2,octx->c_balance_dp[1],errhp,1,ADR(c_balance),
        SIZ(double),SQLT_FLT);
#endif /* USE_IEEE_NUMBER */
/***
  OCIDEF(octx->curo2,octx->c_first_dp[1],errhp,2,c_first,SIZ(c_first)-1,
        SQLT_CHR);
***/

  OCIHandleAlloc(octx->curo2, (dvoid**)&(octx->c_first_dp[1]),
       OCI_HTYPE_DEFINE,0, (dvoid**)0);
  OCIDefineByPos(octx->curo2,&(octx->c_first_dp[1]), errhp, 2, c_first,
      SIZ(c_first)-1, SQLT_CHR, NULL, &rlenp_c_first, NULL, OCI_DEFAULT);



  OCIDEF(octx->curo2,octx->c_middle_dp[1],errhp,3,c_middle,
        SIZ(c_middle)-1,SQLT_AFC);
/***
  OCIDEF(octx->curo2,octx->c_last_dp[1],errhp,4,c_last,SIZ(c_last)-1,
        SQLT_CHR);
***/

  OCIHandleAlloc(octx->curo2, (dvoid**)&(octx->c_last_dp[1]),
       OCI_HTYPE_DEFINE,0, (dvoid**)0);
  OCIDefineByPos(octx->curo2,&(octx->c_last_dp[1]), errhp, 4, c_last,
      SIZ(c_last)-1, SQLT_CHR, NULL, &rlenp_c_last, NULL, OCI_DEFAULT);



  OCIDEF(octx->curo2,octx->o_id_dp[1],errhp,5,ADR(o_id),SIZ(int),SQLT_INT);
  OCIDEF(octx->curo2,octx->o_entry_d_dp[1],errhp,6, &o_entry_d_base,
        SIZ(OCIDate),SQLT_ODT);
  OCIDEF(octx->curo2, octx->o_cr_id_dp[1],errhp,7,ADR(o_carrier_id),
        SIZ(int), SQLT_INT);
  OCIDEF(octx->curo2,octx->o_ol_cnt_dp[1],errhp,8,ADR(o_ol_cnt),
        SIZ(int),SQLT_INT);

/* Bind for last cursor */

  OCIBND(octx->curo3,octx->w_id_bp[2],errhp,":w_id",ADR(w_id),
```

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

  OCIDFNRA(octx->curo3, octx->ol_i_id_dp, errhp, 1, ol_i_id,SIZ(int),SQLT_INT,
            NULL,octx->ol_i_id_len, NULL);
  OCIDFNRA(octx->curo3,octx->ol_supply_w_id_dp,errhp,2, ol_supply_w_id,
        SIZ(int),SQLT_INT, NULL,
        octx->ol_supply_w_id_len, NULL);
#ifdef USE_IEEE_NUMBER
  OCIDFNRA(octx->curo3, octx->ol_quantity_dp,errhp,3, ol_quantity,SIZ(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);
  OCIDEF(octx->curo4,octx->c_count_dp,errhp,1,ADR(octx->rcount),SIZ(int),
       SQLT_INT);

  return (0);
}
```

```
tkvco ()

{

  int i;
  int rcount;

#if defined(ISO9)
  int secondread = 0;
  char sdate[30];
  ub4  datelen;
  sysdate(sdate);
  printf("Order Status started at: %s\n", sdate);
#endif

  for (i = 0; i < NITEMS; i++) {
    octx->ol_supply_w_id_len[i] = sizeof(int);
    octx->ol_i_id_len[i] = sizeof(int);
    octx->ol_quantity_len[i] = sizeof(int);
    octx->ol_amount_len[i] = sizeof(int);
    octx->ol_delivery_d_len[i] = sizeof(OCIDate);
  }
  octx->ol_supply_w_id_csize = NITEMS;
  octx->ol_i_id_csize = NITEMS;
  octx->ol_quantity_csize = NITEMS;
  octx->ol_amount_csize = NITEMS;
  octx->ol_delivery_d_csize = NITEMS;
retry:
  if(bylastname)
  {
   cbctx.reexec = FALSE;
   execstatus=OCIStmtExecute(tpcsvc,octx->curo0,errhp,100,0,
           NULLP(CONST OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
   /* will get OCI_NO_DATA if <100 found */
   if ((execstatus != OCI_NO_DATA) && (execstatus != OCI_SUCCESS))
   {
    errcode=OCIERROR(errhp, execstatus);
    if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR))
    {
     DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
     retries++;
     goto retry;
    } else {
```

```
      return -1;
     }
    }
    if (execstatus == OCI_NO_DATA) /* there are no more rows */
    {
     /* get rowcount, find middle one */
     DISCARD OCIAttrGet(octx->curo0,OCI_HTYPE_STMT,&rcount,NULL,
                        OCI_ATTR_ROW_COUNT,errhp);
     if (rcount <1)
     {
      userlog("ORDERSTATUS  rcount=%d\n",rcount);
      return (-1);
     }
     octx->cust_idx=(rcount)/2 ;
    }
    else
    {
     /* count the number of rows */
     execstatus=OCIStmtExecute(tpcsvc,octx->curo4,errhp,1,0,
              NULLP(CONST
OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
     if ((execstatus != OCI_NO_DATA) && (execstatus != OCI_SUCCESS))
     {
      errcode=OCIERROR(errhp, execstatus);
           if ((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR))
      {
       DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
       retries++;
       goto retry;
      } else {
       return -1;
      }
     }
     if (octx->rcount+1 < 2*10 )
       octx->cust_idx=(octx->rcount+1)/2 ;
     else          /* */
     {
       cbctx.reexec = TRUE;
       cbctx.count = (octx->rcount+1)/2 ;
       execstatus=OCIStmtExecute(tpcsvc,octx->curo0,errhp,cbctx.count,
                          0,NULLP(CONST OCISnapshot),
                          NULLP(OCISnapshot),OCI_DEFAULT);
       /* will get OCI_NO_DATA if <100 found */
       if (cbctx.count > 0)
       {
         userlog ("did not get all rows ");
         return (-1);
       }

       if ((execstatus != OCI_NO_DATA) && (execstatus != OCI_SUCCESS))
       {
        errcode=OCIERROR(errhp, execstatus);
        if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR))
        {
         DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
         retries++;
         goto retry;
        } else {
         return -1;
        }
       }
       octx->cust_idx=0 ;
     }
    }

    octx->c_rowid_cust = octx->c_rowid_ptr[octx->cust_idx];
    execstatus=OCIStmtExecute(tpcsvc,octx->curo1,errhp,1,0,
                NULLP(CONST
OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
    if (execstatus != OCI_SUCCESS)
    {
      errcode=OCIERROR(errhp,execstatus);
      DISCARD OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
      if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR)
        || (errcode == SNAPSHOT_TOO_OLD))
      {
        retries++;
        goto retry;
        } else {
        return -1;
      }
     }
    }
    else
    {
     execstatus=OCIStmtExecute(tpcsvc,octx->curo2,errhp,1,0,
```

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

    printf ("c_id = %d\n", c_id);
    printf ("c_last = %s\n", c_last);
    printf ("c_first = %s\n", c_first);
    printf ("c_middle = %s\n", c_middle);
    printf ("c_balance = %7.2f\n", (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*)FULLDATE,SIZ(
FULLDATE),(text*
)0,0,&datelen,o_entry_d));
    printf ("o_entry_d = %s\n", o_entry_d);
    printf ("o_carrier_id = %d\n", o_carrier_id);
    printf ("o_ol_cnt = %d\n", o_ol_cnt);
    printf ("-----------------------------------------------\n\n", sdate);

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

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

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

 return (0);
```

```
}


void tkvcodone ()

{

  if (octx)
    free (octx);

}

/* end of file tkvcord.c */
```

plpay.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: plpay.c 7030100.1 95/07/19 14:44:59 plai Generic<base> $ Copyr (c)
1994 Oracle";
#endif /* RCSID */

/*=============================================================
======+
|      Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
|                OPEN SYSTEMS PERFORMANCE GROUP                 |
|                    All Rights Reserved                        |

+=============================================================
======+
| FILENAME
|   plpay.c
| DESCRIPTION
|   OCI version (using PL/SQL stored procedure) of
|   PAYMENT transaction in TPC-C benchmark.

+=============================================================
======*/

#include "tpcc.h"

#ifdef TUX
#include <userlog.h>
#endif


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

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;

payctx *pctx;

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

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

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

  /*  build the init statement  and execute it */

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

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

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

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

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

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


  /* bind variables */


  OCIBNDPL(pctx->curp0, pctx->w_id_bp[0], errhp,":w_id",ADR(w_id),SIZ(int),
        SQLT_INT, NULL);
  OCIBNDPL(pctx->curp0, pctx->d_id_bp[0], errhp,":d_id",ADR(d_id),SIZ(int),
        SQLT_INT, NULL);
  OCIBND(pctx->curp0, pctx->c_w_id_bp[0],
errhp,":c_w_id",ADR(c_w_id),SIZ(int),
        SQLT_INT);
  OCIBND(pctx->curp0, pctx->c_d_id_bp[0], errhp,":c_d_id",ADR(c_d_id),SIZ(int),
        SQLT_INT);
  OCIBND(pctx->curp0, pctx->c_id_bp[0], errhp,":c_id",ADR(c_id),SIZ(int),
        SQLT_INT);
#ifdef USE_IEEE_NUMBER
  OCIBNDPL(pctx->curp0, pctx->h_amount_bp[0],
errhp,":h_amount",ADR(h_amount),
```

```
        SIZ(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);
#ifndef USE_IEEE_NUMBER
  OCIBNDPL(pctx->curp0, pctx->c_balance_bp[0], errhp,":c_balance",
        ADR(c_balance), SIZ(double),SQLT_BDOUBLE, &pctx->c_balance_len);
#else
  OCIBNDPL(pctx->curp0, pctx->c_balance_bp[0], errhp,":c_balance",
        ADR(c_balance), SIZ(double),SQLT_FLT, &pctx->c_balance_len);
#endif /* USE_IEEE_NUMBER */
  OCIBNDPL(pctx->curp0, pctx->c_data_bp[0], errhp,":c_data",c_data,SIZ(c_data),
        SQLT_STR, &pctx->c_data_len);
/*
  OCIBNDR(pctx->curp0, pctx->h_date_bp, errhp,":h_date",h_date,SIZ(h_date),
        SQLT_STR, &pctx->h_date_ind, &pctx->h_date_len, &pctx->h_date_rc);
*/
  OCIBNDPL(pctx->curp0, pctx->retries_bp[0], errhp,":retry",ADR(retries),
        SIZ(int), SQLT_INT, &pctx->retries_len);
  OCIBNDPL(pctx->curp0, pctx->cr_date_bp[0], errhp,":cr_date",ADR(cr_date),
        SIZ(OCIDate),SQLT_ODT, &pctx->cr_date_len);

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


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

```
errhp,":h_amount",ADR(h_amount),
        SIZ(float),SQLT_BFLOAT, &pctx->h_amount_len);
#else
  OCIBNDPL(pctx->curp1, pctx->h_amount_bp[1],
errhp,":h_amount",ADR(h_amount),
        SIZ(int),SQLT_INT, &pctx->h_amount_len);
#endif /* USE_IEEE_NUMBER */
  OCIBND(pctx->curp1, pctx->c_last_bp[1], errhp,":c_last",c_last,SIZ(c_last),
        SQLT_STR);
  OCIBNDPL(pctx->curp1, pctx->w_street_1_bp[1], errhp,":w_street_1",w_street_1,
        SIZ(w_street_1),SQLT_STR, &pctx->w_street_1_len);
  OCIBNDPL(pctx->curp1, pctx->w_street_2_bp[1], errhp,":w_street_2",w_street_2,
        SIZ(w_street_2),SQLT_STR, &pctx->w_street_2_len);
  OCIBNDPL(pctx->curp1, pctx->w_city_bp[1], errhp,":w_city",w_city,SIZ(w_city),
        SQLT_STR, &pctx->w_city_len);
  OCIBNDPL(pctx->curp1, pctx->w_state_bp[1], errhp,":w_state",w_state,
        SIZ(w_state), SQLT_STR, &pctx->w_state_len);
  OCIBNDPL(pctx->curp1, pctx->w_zip_bp[1], errhp,":w_zip",w_zip,SIZ(w_zip),
        SQLT_STR, &pctx->w_zip_len);
  OCIBNDPL(pctx->curp1, pctx->d_street_1_bp[1], errhp,":d_street_1",d_street_1,
        SIZ(d_street_1),SQLT_STR, &pctx->d_street_1_len);
  OCIBNDPL(pctx->curp1, pctx->d_street_2_bp[1], errhp,":d_street_2",d_street_2,
        SIZ(d_street_2),SQLT_STR, &pctx->d_street_2_len);
  OCIBNDPL(pctx->curp1, pctx->d_city_bp[1], errhp,":d_city",d_city,SIZ(d_city),
        SQLT_STR, &pctx->d_city_len);
  OCIBNDPL(pctx->curp1, pctx->d_state_bp[1], errhp,":d_state",d_state,
        SIZ(d_state), SQLT_STR, &pctx->d_state_len);
  OCIBNDPL(pctx->curp1, pctx->d_zip_bp[1], errhp,":d_zip",d_zip,SIZ(d_zip),
        SQLT_STR, &pctx->d_zip_len);
  OCIBNDPL(pctx->curp1, pctx->c_first_bp[1], errhp,":c_first",c_first,
        SIZ(c_first), SQLT_STR, &pctx->c_first_len);
  OCIBNDPL(pctx->curp1, pctx->c_middle_bp[1], errhp,":c_middle",c_middle,2,
        SQLT_AFC, &pctx->c_middle_len);

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

  return (0);
}


tkvcp ()
{

retry:

  pctx->w_id_len = SIZ(w_id);
  pctx->d_id_len = SIZ(d_id);
  pctx->c_w_id_len = 0;
  pctx->c_d_id_len = 0;
  pctx->c_id_len = 0;
```

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

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


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



void tkvcpdone ()

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

plsto.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: plsto.c 7010000.3 95/02/14 12:48:03 plai Generic<base> $ Copyr (c)
1994 Oracle";
#endif /* RCSID */

/*=========================================================
======+
|      Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
|              OPEN SYSTEMS PERFORMANCE GROUP                   |
|                  All Rights Reserved                          |

+=========================================================
```

```
======+
| FILENAME
|    plsto.c
| DESCRIPTION
|    OCI version of STOCK LEVEL transaction in TPC-C benchmark.

+=============================================================
======*/

#include "tpcc.h"

#ifdef PLSQLSTO
#define SQLTXT "BEGIN stocklevel.getstocklevel (:w_id, :d_id, :threshold, \
  :low_stock); END;"
#else
/*+ USE_NL(ordl) nocache (stok) */
#define SQLTXT "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

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;

stoctx *sctx;

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

  /* bind variables */

  OCIBND(sctx->curs,sctx->w_id_bp,errhp, ":w_id", ADR(w_id),sizeof(int),
       SQLT_INT);
  OCIBND(sctx->curs,sctx->d_id_bp,errhp, ":d_id", ADR(d_id),sizeof(int),
    SQLT_INT);
#ifdef USE_IEEE_NUMBER
  OCIBND(sctx->curs,sctx->threshold_bp,errhp, ":threshold", ADR(threshold),
    sizeof(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);
```

```
}

tkvcs ()
{

retry:
   execstatus= OCIStmtExecute(tpcsvc,sctx->curs,errhp,1,0,0,0,
                        OCI_COMMIT_ON_SUCCESS | OCI_DEFAULT);
   if (execstatus != OCI_SUCCESS)
   {
     errcode=OCIERROR(errhp,execstatus);
     OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
     if((errcode == NOT_SERIALIZABLE) || (errcode == RECOVERR)
        || (errcode == SNAPSHOT_TOO_OLD))
     {
       retries++;
       goto retry;
     } else {
       return -1;
     }
    }

  return (0);
}

void tkvcsdone ()
{
  if(sctx) free(sctx);
}
```

tpccpl.c

```
#ifdef RCSID
static char *RCSid =
   "$Header: tpccpl.c 7030100.2 96/04/02 17:51:34 plai Generic<base> $ Copyr (c)
1994 Oracle";
#endif /* RCSID */


/*=========================================================
======+
|      Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
|                OPEN SYSTEMS PERFORMANCE GROUP                 |
|                   All Rights Reserved                         |

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

+=========================================================
======*/

#include <stdio.h>
#include <time.h>
#include <sys/timeb.h>
#include "tpcc.h"
#ifdef TUX
#include <userlog.h>
#else
#include <stdarg.h>
#endif

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

FILE *lfp;
FILE *fopen ();
#ifdef ORA_NT
#undef boolean
//#include "dpbcore.h"
//#define gettime dpbtimef
#else
extern double gettime ();
#endif
int proc_no = 0;
static int logon = 0;
static int new_init = 0;
static int pay_init = 0;
static int ord_init = 0;
static int del_init_oci = 0;
static int del_init_plsql = 0;
```

```
static int sto_init = 0;
static int res_init = 0;

int execstatus;
int errcode;

struct timeb  timebuffer;//change

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


/* for stock-level transaction */

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

/* for delivery transaction */

int del_o_id[10];
int retries;

/* for order-status transaction */

int bylastname;
char c_last[17];
char c_first[17];
char c_middle[3];
double c_balance;
int o_id;
text o_entry_d[20];
ub4  datelen;
int o_carrier_id;
int o_ol_cnt;
int ol_supply_w_id[15];
int ol_i_id[15];
#ifdef USE_IEEE_NUMBER
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_quanti10[15];
int nol_quanti91[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 status;
int tracelevel = 0;

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

#ifndef AVOID_DEADLOCK
int indx[NITEMS], ordl_cnt;
void swap(struct newstruct *str, int i, int j);
void q_sort(int *arr, struct newstruct *str, int left, int right);
#endif

/*
extern char oracle_home[256];
*/

/* NewOrder Binding stuff */

void debuglog(char *filename, char *format, ...)
{
        FILE *fp;
        va_list args;
        char buf[4096];
        int len;

        struct tm* newtime;
    time_t now;

        va_start( args, format );
        //_strtime( buf );

    time(&now);
    newtime=localtime(&now);
    strcpy(buf, asctime(newtime));

        strcat( buf," ");
        len = strlen( buf );
        (void)vsnprintf( buf+ len, sizeof( buf) - len - 1, format, args);
        buf[sizeof( buf )- 1]= '\0';
        va_end( args );

        if (fp = fopen(filename, "a"))
          {
                        fprintf(fp,"%s.\n", buf);
                        fflush(fp);
                        fclose(fp);
        }
}


#ifndef TUX
void userlog (char* fmtp, ...)
```

```
{
  va_list va;
  va_start(va,fmtp);
  vfprintf(stderr,fmtp,va);
  va_end(va);
}
#endif

/* vmm313 void ocierror(fname, lineno, errhp, status)  */
/* vmm313 void ocierror(fname, lineno, errhp, status)  */
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:
          debuglog("/tmp/OCIErrorLog","2 Error -
OCI_SUCCESS_WITH_INFO\n");
    //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);
          debuglog("/tmp/OCIErrorLog","2 Error - %s\n", errbuf);
    //fprintf(stderr,"Error - %s\n", errbuf);
    break;
  case OCI_NEED_DATA:
          debuglog("/tmp/OCIErrorLog","3 Module %s Line %d\n", fname, lineno);
    debuglog("/tmp/OCIErrorLog","3 Error - OCI_NEED_DATA\n");
    //fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    //fprintf(stderr,"Error - OCI_NEED_DATA\n");
    return (IRRECERR);
  case OCI_NO_DATA:
          debuglog("/tmp/OCIErrorLog","4 Module %s Line %d\n", fname, lineno);
    debuglog("/tmp/OCIErrorLog","4 Error - OCI_NO_DATA\n");
    //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)
          {
                    debuglog("/tmp/OCIErrorLog","5 ERROR");
                    return (errcode);
          }
    if (errcode == SNAPSHOT_TOO_OLD)
          {
                    debuglog("/tmp/OCIErrorLog","6 snapshot_too_old");
                    return (errcode);
          }
    while (lstat != OCI_NO_DATA)
    {
            debuglog("/tmp/OCIErrorLog","7 Module %s Line %d\n", fname, lineno);
      debuglog("/tmp/OCIErrorLog","7 Error - %s\n", errbuf);
      //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);
  /* vmm313    TPCexit(1); */
  /* vmm313    exit(1); */
    case OCI_INVALID_HANDLE:
          debuglog("/tmp/OCIErrorLog","8 Module %s Line %d\n", fname, lineno);
    debuglog("/tmp/OCIErrorLog","8 Error - OCI_INVALID_HANDLE\n");
    //fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    //fprintf(stderr,"Error - OCI_INVALID_HANDLE\n");
    TPCexit(1);
    exit(-1);
  case OCI_STILL_EXECUTING:
          debuglog("/tmp/OCIErrorLog","9 Module %s Line %d\n", fname, lineno);
    debuglog("/tmp/OCIErrorLog","9 Error - OCI_STILL_EXECUTE\n");
    //fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    //fprintf(stderr,"Error - OCI_STILL_EXECUTE\n");
    return (IRRECERR);
```

```
  case OCI_CONTINUE:
          debuglog("/tmp/OCIErrorLog","10 Module %s Line %d\n", fname, lineno);
    debuglog("/tmp/OCIErrorLog","10 Error - OCI_CONTINUE\n");
    //fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    //fprintf(stderr,"Error - OCI_CONTINUE\n");
    return (IRRECERR);
  default:
          debuglog("/tmp/OCIErrorLog","11 Module %s Line %d\n", fname, lineno);
    debuglog("/tmp/OCIErrorLog","11 Status - %s\n", status);
    //fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    //fprintf(stderr,"Status - %s\n", status);
    return (IRRECERR);
  }
  return (RECOVERR);
}


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

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

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

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

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

/*
    sprintf(realfile,"%s/bench/tpc/tpcc/blocks/%s",oracle_home,fnam);
*/
    sprintf(realfile,"%s",fnam);
    fd = vopen(realfile,"r");
    while (fgets((char *)linebuf+nulpt, SQL_BUF_SIZE,fd))
    {
        nulpt = strlen((char *)linebuf);
    }
    return(nulpt);
}

#ifdef NOT
void vgetdate (unsigned char *oradt)
{
 struct tm *loctime;
 time_t  int_time;

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

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

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

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

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

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


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

 return;

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

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


      int day,month,year;

      memcpy(&Date,oradt,7);

      year = (Date.century-100)*100 + Date.year-100;
      month = Date.month;
      day = Date.day;
      sprintf(outdate,"%02d-%02d-%4d\0",day,month,year);

      return;
}

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

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


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

      memcpy(&Date,oradt,7);

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

      sprintf(outdate,"%02d-%02d-%4d %02d:%02d:%02d\0",
                  day,month,year,hour,min,sec);

      return;
}
#endif
```

```
void TPCexit (void)

{

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



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

  if (lfp) {
    fclose (lfp);
    lfp = NULL;
  }

}
```

```
TPCinit (id, uid, pwd)

int id;
char *uid;
char *pwd;

{

  char filename[40];
  text stmbuf[100];

  proc_no = id;
  sprintf (filename, "tpcc_%d.del", proc_no);
  if ((lfp = fopen (filename, "w")) == NULL) {
#ifdef TUX
    userlog ("Error in TPC-C server %d: Failed to open %s\n",
          proc_no, filename);
#else
    fprintf (stderr, "Error in TPC-C server %d: Failed to open %s\n",
          proc_no, filename);
#endif

    return (-1);
  }



  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);
  OCIServerAttach(tpcsrv, errhp, (text *)0,0,OCI_DEFAULT); //change
  OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsvc, OCI_HTYPE_SVCCTX, 0 ,
(dvoid **)0);
  OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX, (dvoid *)tpcsrv,
(ub4)0,OCI_ATTR_SERVER, errhp);
  OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr, OCI_HTYPE_SESSION, 0 ,
(dvoid **)0);
  OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)uid,
(ub4)strlen(uid),OCI_ATTR_USERNAME, errhp);
```

```
  OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)pwd,
(ub4)strlen(pwd),
                OCI_ATTR_PASSWORD, errhp);
  OCIERROR(errhp, OCISessionBegin(tpcsvc, errhp, tpcusr, OCI_CRED_RDBMS,
OCI_DEFAULT));
  OCIAttrSet(tpcsvc, OCI_HTYPE_SVCCTX, tpcusr, 0, OCI_ATTR_SESSION,
errhp);
  /* run all transaction in serializable mode */

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

/*
This is done in cvdrv.c
  if (tracelevel == 2) {
    OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
    memset(stmbuf,0,100);
    sprintf ((char *) stmbuf, SQLTXTTRC);
    OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
                              OCI_NTV_SYNTAX, OCI_DEFAULT);
    OCIERROR(errhp, OCIStmtExecute(tpcsvc, curi,
errhp,1,0,0,0,OCI_DEFAULT));
    OCIHandleFree((dvoid *)curi, OCI_HTYPE_STMT);
  }
*/
  if (tracelevel == 3) {
    OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid**)0);
    memset(stmbuf,0,100);
    sprintf ((char *) stmbuf, SQLTXTTIM);
    OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
                              OCI_NTV_SYNTAX, OCI_DEFAULT);
    OCIERROR(errhp, OCIStmtExecute(tpcsvc, curi,
errhp,1,0,0,0,OCI_DEFAULT));
    OCIHandleFree((dvoid *)curi, OCI_HTYPE_STMT);
  }

  logon = 1;

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

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

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

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


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

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

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

```
  return (0);

}



TPCnew (str)

struct newstruct *str;

{

  int i;

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

#ifndef AVOID_DEADLOCK

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

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

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

#endif

/*
  vgetdate(cr_date); */

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

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

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

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

  str->newout.terror = NOERR;
  str->newout.o_id = o_id;
  str->newout.o_ol_cnt = o_ol_cnt;
  strncpy (str->newout.c_last, c_last, 17);
  strncpy (str->newout.c_credit, c_credit, 3);
  str->newout.c_discount = c_discount;
  str->newout.w_tax = (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++) {
          str->newout.ol_i_id[i] = nol_i_id[i];//add
          str->newout.ol_supply_w_id[i] = nol_supply_w_id[i];//add
    str->newout.ol_quantity[i] = nol_quantity[i];//add
    strncpy (str->newout.i_name[i], i_name[i], 25);
    str->newout.brand_generic[i] = brand_generic[i][0];
#ifndef 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 */
```

```
  }

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

  if (status){
    strcpy (str->newout.status, "Item number is not valid");
    return(0); //add for abort transaction
  }
  else
    str->newout.status[0] = '\0';
  str->newout.retry = retries;
#if defined(TOP)  || defined(TUX)    /* changed mjb 17 feb for tuxedo */
  return(1);
#else
  return (0);
#endif

}



TPCpay (str)

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

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

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



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

```
  strncpy (str->payout.d_zip, d_zip, 10);
  str->payout.c_id = c_id;
  strncpy (str->payout.c_first, c_first, 17);
  strncpy (str->payout.c_middle, c_middle, 3);
  strncpy (str->payout.c_last, c_last, 17);
  strncpy (str->payout.c_street_1, c_street_1, 21);
  strncpy (str->payout.c_street_2, c_street_2, 21);
  strncpy (str->payout.c_city, c_city, 21);
  strncpy (str->payout.c_state, c_state, 3);
  strncpy (str->payout.c_zip, c_zip, 10);
  strncpy (str->payout.c_phone, c_phone, 17);
  strncpy (str->payout.c_since, (char*)c_since_d, 11);
  strncpy (str->payout.c_credit, c_credit, 3);
  str->payout.c_credit_lim = (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;
#if defined(TOP)  || defined(TUX)   /* changed mjb 17 Feb */
  return(1);
#else
  return (0);
#endif

}



TPCord (str)

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);
    debuglog("ord.log",c_last);
  }
  else {
    c_id = str->ordin.c_id;
    memset (c_last, ' ', sizeof(c_last)-1);
    c_last[16] = '\0';
  }
  retries = 0;

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

  datelen = sizeof(o_entry_d);
  OCIERROR(errhp,

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

  debuglog("ord.log","after c_last=%s",c_last);


  str->ordout.terror = NOERR;
  str->ordout.c_id = c_id;

  strcpy (str->ordout.c_last,"");
  strcpy (str->ordout.c_first,"");
  strcpy (str->ordout.c_middle, "");

  c_last[rlenp_c_last] = '\0';
  c_first[rlenp_c_first] = '\0';

  strncpy (str->ordout.c_last, c_last, 17);
  strncpy (str->ordout.c_first, c_first, 17);
  strncpy (str->ordout.c_middle, c_middle, 3);

  str->ordout.c_balance = c_balance/100;
  str->ordout.o_id = o_id;
  strncpy (str->ordout.o_entry_d, (char*)o_entry_d, 20);
  if ( o_carrier_id == 11 )
    str->ordout.o_carrier_id = 0;
```

```
  else
     str->ordout.o_carrier_id = o_carrier_id;
  str->ordout.o_ol_cnt = o_ol_cnt;
  for (i = 0; i < o_ol_cnt; i++) {
    ol_delivery_d[i][10] = '\0';
    if ( !strcmp((char*)ol_delivery_d[i],"15-09-1911") )
            strncpy((char*)ol_delivery_d[i],"NOT DELIVR",10);
    str->ordout.ol_supply_w_id[i] = ol_supply_w_id[i];
    str->ordout.ol_i_id[i] = ol_i_id[i];
#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;
#if defined(TOP)  || defined(TUX)
  return(1);
#else
  return (0);
#endif

}



TPCdel (str)

struct delstruct *str;

{

  double tr_end;
  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));

  str->delin.plsqlflag = 1; // 1:PLSQLDEL, 0:OCIDEL
  //We suspect PLSQLDEL has bugs.

  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);
  }

  //tr_end = gettime ();
  ftime(&timebuffer);
  tr_end = timebuffer.time + timebuffer.millitm/1000.0;
/* old type
  fprintf (lfp, "%d %d %f %f %d %d", str->delin.in_timing_int,
         (tr_end - str->delin.qtime) <= DELRT ? 1 : 0,
         str->delin.qtime, tr_end, w_id, o_carrier_id);
  for (i = 0; i < 10; i++) {
    fprintf (lfp, " %d %d", i + 1, del_o_id[i]);
    if (del_o_id[i] <= 0) {
#ifdef TUX
      userlog ("DELIVERY: no new order for w_id: %d, d_id %d\n",
            w_id, i + 1);
#else
      fprintf (stderr, "DELIVERY: no new order for w_id: %d, d_id %d\n",
            w_id, i + 1);
#endif
    }
  }
  fprintf (lfp, " %d\n", retries);
  str->delout.terror = NOERR;
  str->delout.retry = retries;
*/
fprintf (lfp, "%f %f %d %d",str->delin.qtime, tr_end, w_id, o_carrier_id);
  for (i = 0; i < 10; i++) {
    fprintf (lfp, " %d",del_o_id[i]);
    if (del_o_id[i] <= 0) {
#ifdef TUX
      userlog ("DELIVERY: no new order for w_id: %d, d_id %d\n",
            w_id, i + 1);
```

```
#else
        fprintf (stderr, "DELIVERY: no new order for w_id: %d, d_id %d\n",
                w_id, i + 1);
#endif
      }
    }
    fprintf (lfp, "\n");
    str->delout.terror = NOERR;
    str->delout.retry = retries;

#if defined(TOP) || defined(TUX)   /* changed mjb 17 feb */
    return(1);
#else
    return (0);
#endif

}



TPCsto (str)

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;
#if defined(TOP) || defined(TUX) /* changed mjb 17 feb */
    return(1);
#else
    return (0);
#endif

}

#ifndef AVOID_DEADLOCK

void 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 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 = str->newout.s_quantity[i];
  str->newout.s_quantity[i] = str->newout.s_quantity[j];
  str->newout.s_quantity[j] = 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
```

tpccsvr.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: tpccsvr.c 7030100.1 95/07/19 15:39:28 plai Generic<base> $ Copyr (c)
1995 Oracle";
#endif /* RCSID */

/*=============================================================
======+
 |      Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
 |                OPEN SYSTEMS PERFORMANCE GROUP                 |
 |                    All Rights Reserved                        |

+=============================================================
======+
 | FILENAME
 |   tpccsvr.c
 | DESCRIPTION
 |   Tuxedo server for TPC-C. use a #define TUX
 |   TOPEND server for TPC-C. use a #define TOP

+=============================================================
======*/



#include <stdio.h>
#include <math.h>
#ifdef WINDOWS
#include <windows.h>
#include <process.h>
#endif
```

```
#ifdef TUX
#include <atmi.h>          // must occur prior to include of tpccapi.h
#include <stdlib.h>        // for generation of random seed for server id
#include <time.h>          // for generation of random seed for server id
#endif

#define TPCSVR
#include "tpcc.h"
#include "tpcc_info.h"
#ifdef WINDOWS
#include "httpext.h"       //ISAPI DDL information header
#include "tpccapi.h"       //this dlls specific structure, value e.t. header
#endif


#ifdef TUX


  #include <tmenv.h>
  #include <xa.h>
  #include <userlog.h>

/* set up pointers for type casting */
struct newstruct *newinfo;
struct paystruct *payinfo;
struct ordstruct *ordinfo;
struct delstruct *delinfo;
struct stostruct *stoinfo;

//extern void TMlog();

#endif


void TMlog( char *format, ...)
{
            va_list args;
            char buf[4096];
            int len;
            struct tm* newtime;
        time_t now;
            va_start( args, format );
#ifdef WINDOWS
            _strtime( buf );
#else
        time(&now);
        newtime=localtime(&now);
        strcpy(buf, asctime(newtime));
#endif
            strcat( buf," ");
            len = strlen( buf );
#ifdef WINDOWS
            (void)_vsnprintf( buf+ len, sizeof( buf) - len - 1, format, args);
#else
            (void)vsnprintf( buf+ len, sizeof( buf) - len - 1, format, args);
#endif
            buf[sizeof( buf )- 1]= '\0';
            va_end( args );
            userlog( buf );
}


/* FUNCTION: int tpsvrinit (int argc, char *argv[]);
 *
 * PURPOSE:     Connects into database
 * ARGUMENTS:   parameters passed in as  int svrid, char *uid, char *pwd, int
txntype
 *              do not check ordering, assume correct
 *              svrid:   an id number for server running
 *                uid:   the userid for the database
 *                pwd:   the password for the userid
 *            txntype:   transaction type the server will be running
 * RETURNS:     None
 *
 * COMMENTS:    None
 *
 */


int tpsvrinit (int argc, char *argv[])

{

 int svrid, txntype;
 char *uid, *pwd;
```

```
 int svrcnt;

    /* pull out the values from argv */
    svrid = atoi(argv[0]);
    uid = argv[1];
    pwd = argv[2];
    txntype = atoi(argv[3]);

#ifdef TUX

            //srand ( (unsigned)time( NULL ) );
     //svrcnt = rand();
     svrcnt = getpid();

    /* send 6 for all txns to be inited */
    /* fix uid an pwd for now, pull out later */
    /* not passing parameters through TUX  yet */
    if (TPCinit (svrcnt, "tpcc", "tpcc", 6)) {
            TMlog( " FAILED to init all txns types");

            return (-1);
    }

    return 0;

#else                                   // for topend
    if (TPCinit (svrid, uid, pwd, txntype)) {
             fprintf(stderr, "Failed in TPCinit (probably connecting).");
             exit (1);
    }

   return (1);
#endif

}



void tpsvrdone ()

{
  TPCexit (0);

}

/* FUNCTION: int NEWORDER(CLIENTDATA *jobData, NewOrderData *neword,
int deadlock)
 *
 * PURPOSE:     This function handles the new order transaction.
 *
 * ARGUMENTS:   deadlock :  count of deadlocks encountered during txn
 *              jobData:    pointer to entire block of user data
 *              neword:     pointer to datastructure in jobData that contains the new order
data
 * RETURNS:             int     TRUE    transaction committed
 *                              FALSE   item number not valid
 *                              -1              deadlock max retry reached
 *
 *
 * COMMENTS:    None
 *
 */

#ifdef TOP
int NEWORDER(CLIENTDATA *jobData, NewOrderData *neword, int deadlock)
#else
void NEWORDER (TPSVCINFO *msg)
#endif

{

#ifdef TOP
  int result;

  result =  TPCnew(neword);

  return result;

#else                           // for Tuxedo



  newinfo = (struct newstruct *) msg->data;
  newinfo->retval = TPCnew (newinfo);  // set return value to 0 or -1
```

```
  // always return tpreturn success - let client side poll retval for actual error
  tpreturn (TPSUCCESS, 0, (char *) newinfo, sizeof (struct newstruct), 0);

#endif

}

/* FUNCTION: int PAYMENT(CLIENTDATA *jobData, PaymentData *paydata,
int deadlock)
 *
 * PURPOSE:     This function handles the new order transaction.
 *
 * ARGUMENTS:   deadlock :   count of deadlocks encountered during txn
 *              jobData:     pointer to entire block of user data
 *              paydata:     pointer to datastructure in jobData that contains the new order
data
 * RETURNS:           int     TRUE    transaction committed
 *                            FALSE   item number not valid
 *                            -1              deadlock max retry reached
 *
 *
 * COMMENTS:    None
 *
 */

#ifdef TOP
int PAYMENT(CLIENTDATA *jobData, PaymentData *paydata, int deadlock)
#else
void PAYMENT (TPSVCINFO *msg)
#endif

{

#ifdef TOP

      int result;

      result = TPCpay(paydata);

      return result;
#else
  payinfo = (struct paystruct *) msg->data;
  payinfo->retval = TPCpay (payinfo);  // set return value to 1 or 0 or -1

  // always return tpreturn success - let client side poll retval for actual error
  tpreturn (TPSUCCESS, 0, (char *) payinfo, sizeof (struct paystruct), 0);

#endif

}

/* FUNCTION: int ORDERSTATUS(CLIENTDATA *jobData, OrderStatusData
*orddata, int deadlock)
 *
 * PURPOSE:     This function handles the new order transaction.
 *
 * ARGUMENTS:   deadlock :   count of deadlocks encountered during txn
 *              jobData:     pointer to entire block of user data
 *              orddata:     pointer to datastructure in jobData that contains the new order
data
 * RETURNS:           int     TRUE    transaction committed
 *                            FALSE   item number not valid
 *                            -1              deadlock max retry reached
 *
 *
 * COMMENTS:    None
 *
 */



#ifdef TOP
int ORDERSTATUS(CLIENTDATA *jobData, OrderStatusData *orddata, int
deadlock)
#else
void ORDERSTATUS (TPSVCINFO *msg)
#endif

{

#ifdef TOP
            int result;
```

```
        result = TPCord(orddata);

        return result;

#else


  ordinfo = (struct ordstruct *) msg->data;
  ordinfo->retval = TPCord (ordinfo);  // set return value to 0 or -1

  // always return tpreturn success - let client side poll retval for actual error
  tpreturn (TPSUCCESS, 0, (char *) ordinfo, sizeof (struct ordstruct), 0);


#endif

}

/* FUNCTION: int DELIVERY(CLIENTDATA *jobData, DeliveryData *deldata,
int deadlock)
 *
 * PURPOSE:     This function handles the new order transaction.
 *
 * ARGUMENTS:   deadlock :   count of deadlocks encountered during txn
 *              jobData:     pointer to entire block of user data
 *              stodata:     pointer to datastructure in jobData that contains the new order
data
 * RETURNS:           int     TRUE    transaction committed
 *                            FALSE   item number not valid
 *                            -1              deadlock max retry reached
 *
 *
 * COMMENTS:    None
 *
 */

#ifdef TOP
int DELIVERY(CLIENTDATA *jobData, DeliveryData *deldata, int deadlock)
#else
void DELIVERY (TPSVCINFO *msg)
#endif

{
#ifdef TOP
        int result;

        result = TPCdel(deldata);

        return result;

#else


  delinfo = (struct delstruct *) msg->data;
  delinfo->retval = TPCdel (delinfo);  // set return value to 0 or -1

  // always return tpreturn success - let client side poll retval for actual error
  tpreturn (TPSUCCESS, 0, (char *) delinfo, sizeof (struct delstruct), 0);

#endif

}



/* FUNCTION: int STOCKLEVEL(CLIENTDATA *jobData, StockLevelData
*stodata, int deadlock)
 *
 * PURPOSE:     This function handles the new order transaction.
 *
 * ARGUMENTS:   deadlock :   count of deadlocks encountered during txn
 *              jobData:     pointer to entire block of user data
 *              stodata:     pointer to datastructure in jobData that contains the new order
data
 * RETURNS:           int     TRUE    transaction committed
 *                            FALSE   item number not valid
 *                            -1              deadlock max retry reached
 *
 *
 * COMMENTS:    None
 *
 */

#ifdef TOP
int STOCKLEVEL(CLIENTDATA *jobData, StockLevelData *stodata, int
```

```
deadlock)
#else
void STOCKLEVEL (TPSVCINFO *msg)
#endif

{

#ifdef TOP

        int result;

        result = TPCsto(stodata);

        return result;

#else


  stoinfo = (struct stostruct *) msg->data;
  stoinfo->retval = TPCsto (stoinfo);  // set return value to 0 or -1

  // always return tpreturn success - let client side poll retval for actual error
  tpreturn (TPSUCCESS, 0, (char *) stoinfo, sizeof (struct stostruct), 0);


#endif
}

/* FUNCTION: int OPSTUXSERVER(CLIENTDATA *jobData, NewOrderData
*neword, int deadlock)
 *
 * PURPOSE:     This function handles all transactions.
 *
 * ARGUMENTS:   deadlock :  count of deadlocks encountered during txn
 *          jobData:    pointer to entire block of user data
 *          neword:     pointer to datastructure in jobData that contains the new order
data
 * RETURNS:          int     TRUE    transaction committed
 *                           FALSE   item number not valid
 *                           -1              deadlock max retry reached
 *
 *
 * COMMENTS:    None
 *
 */

#ifdef TOP
int OPSTUXSERVER(CLIENTDATA *jobData, NewOrderData *neword, int
deadlock)
#else
void OPSTUXSERVER (TPSVCINFO *msg)
#endif

{

#ifdef TOP
  int result;

  result =  TPCnew(neword);

  return result;

#else                           // for Tuxedo

  if (msg->len == sizeof(struct newstruct)) {  // len for neworder
     //if (msg->len == 928) {  // len for neworder for old
     newinfo = (struct newstruct *) msg->data;
     newinfo->retval = TPCnew (newinfo);  // set return value to 0 or -1

     // always return tpreturn success - let client side poll retval for actual error
     tpreturn (TPSUCCESS, 0, (char *) newinfo, sizeof (struct newstruct), 0);
  }
  else if (msg->len == sizeof(struct paystruct)) {
           //if (msg->len == 616) {  // len for payment for old
     payinfo = (struct paystruct *) msg->data;
           payinfo->retval = TPCpay (payinfo);  // set return value to 1 or 0 or -1

           // always return tpreturn success - let client side poll retval for actual error
           tpreturn (TPSUCCESS, 0, (char *) payinfo, sizeof (struct paystruct), 0);
  }
  else if (msg->len == sizeof(struct ordstruct)) {
     //if (msg->len == 544) {  // len for order status
     ordinfo = (struct ordstruct *) msg->data;
     ordinfo->retval = TPCord (ordinfo);  // set return value to 0 or -1

     // always return tpreturn success - let client side poll retval for actual error
```

```
    tpreturn (TPSUCCESS, 0, (char *) ordinfo, sizeof (struct ordstruct), 0);
  }
  else if (msg->len == sizeof(struct delstruct)) {
    //if (msg->len == 40) {  // len for delivery
    delinfo = (struct delstruct *) msg->data;
    delinfo->retval = TPCdel (delinfo);  // set return value to 0 or -1


    // always return tpreturn success - let client side poll retval for actual error
    tpreturn (TPSUCCESS, 0, (char *) delinfo, sizeof (struct delstruct), 0);
  }
  else {  // assume rest is stock level

    stoinfo = (struct stostruct *) msg->data;
    stoinfo->retval = TPCsto (stoinfo);  // set return value to 0 or -1

    // always return tpreturn success - let client side poll retval for actual error
    tpreturn (TPSUCCESS, 0, (char *) stoinfo, sizeof (struct stostruct), 0);
  }

#endif

}


/* FUNCTION: int OPSTUXSERVER2(CLIENTDATA *jobData, NewOrderData
*neword, int deadlock)
 *
 * PURPOSE:     This function handles derivery transactions.
 *
 * ARGUMENTS:   deadlock :  count of deadlocks encountered during txn
 *              jobData:    pointer to entire block of user data
 *              neword:     pointer to datastructure in jobData that contains the new order
data
 * RETURNS:           int     TRUE    transaction committed
 *                            FALSE   item number not valid
 *                            -1              deadlock max retry reached
 *
 *
 * COMMENTS:    None
 *
 */
#ifdef TOP
int OPSTUXSERVER2(CLIENTDATA *jobData, NewOrderData *neword, int
deadlock)
#else
void OPSTUXSERVER2 (TPSVCINFO *msg)
#endif

{

#ifdef TOP
  int result;

  result =  TPCnew(neword);

  return result;

#else                           // for Tuxedo

  if (msg->len == sizeof(struct newstruct)) {  // len for neworder
    //if (msg->len == 928) {  // len for neworder for old
    newinfo = (struct newstruct *) msg->data;
    newinfo->retval = TPCnew (newinfo);  // set return value to 0 or -1

    // always return tpreturn success - let client side poll retval for actual error
    tpreturn (TPSUCCESS, 0, (char *) newinfo, sizeof (struct newstruct), 0);
  }
  else if (msg->len == sizeof(struct paystruct)) {
            //if (msg->len == 616) {  // len for payment for old
    payinfo = (struct paystruct *) msg->data;
            payinfo->retval = TPCpay (payinfo);  // set return value to 1 or 0 or -1

            // always return tpreturn success - let client side poll retval for actual error
            tpreturn (TPSUCCESS, 0, (char *) payinfo, sizeof (struct paystruct), 0);
  }
  else if (msg->len == sizeof(struct ordstruct)) {
    //if (msg->len == 544) {  // len for order status
    ordinfo = (struct ordstruct *) msg->data;
    ordinfo->retval = TPCord (ordinfo);  // set return value to 0 or -1

    // always return tpreturn success - let client side poll retval for actual error
    tpreturn (TPSUCCESS, 0, (char *) ordinfo, sizeof (struct ordstruct), 0);
  }
  else if (msg->len == sizeof(struct delstruct)) {
```

```
    //if (msg->len == 40) {  // len for delivery
    delinfo = (struct delstruct *) msg->data;
    delinfo->retval = TPCdel (delinfo);  // set return value to 0 or -1


    // always return tpreturn success - let client side poll retval for actual error
    tpreturn (TPSUCCESS, 0, (char *) delinfo, sizeof (struct delstruct), 0);
  }
  else {  // assume rest is stock level

    stoinfo = (struct stostruct *) msg->data;
    stoinfo->retval = TPCsto (stoinfo);  // set return value to 0 or -1

    // always return tpreturn success - let client side poll retval for actual error
    tpreturn (TPSUCCESS, 0, (char *) stoinfo, sizeof (struct stostruct), 0);
  }

#endif

}
```

bs-mb.c

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

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

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

#ifndef _TMDLLIMPORT
#define _TMDLLIMPORT
#endif

_TMDLLIMPORT extern struct xa_switch_t tmnull_switch;

struct tmsvrargs_t tmsvrargs = {
        NULL,
        &_tmdsptchtbl[0],
        0,
        tpsvrinit,
        tpsvrdone,
        _tmrunserver,    /* PRIVATE  */
        NULL,                   /* RESERVED */
        NULL,                   /* RESERVED */
        NULL,                   /* RESERVED */
        NULL                    /* RESERVED */
};

struct tmsvrargs_t *
#ifdef _TMPROTOTYPES
_tmgetsvrargs(void)
#else
_tmgetsvrargs()
#endif
{
        tmsvrargs.xa_switch = &tmnull_switch;
        return(&tmsvrargs);
}

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

```
        return( _tmstartserver( argc, argv, _tmgetsvrargs()));
}
```

tpcc_info.h

```
/*
 * $Header: tpcc_info.h 7030100.1 95/07/19 15:11:37 plai Generic<base> $ Copyr (c)
1995 Oracle
 */
/*=============================================================
======+
|       Copyright (c) 1995  Oracle Corp, Redwood Shores, CA     |
|                OPEN SYSTEMS PERFORMANCE GROUP                 |
|                    All Rights Reserved                        |

+=============================================================
======+
| FILENAME
|    tpcc_info.h
| DESCRIPTION
|    Include file for TPC-C benchmark programs.

+=============================================================
======*/

#ifndef TPCC_INFO_H
#define TPCC_INFO_H

/* this set is duplicated in c_Defs.h, c_Defs.h is used for batch driver */
#define MENTXN    0         /* menu txn */
#define NEWTXN    1         /* new order transaction */
#define PAYTXN    2         /* payment transaction */
#define ORDTXN    3         /* order status transaction */
#define DELTXN    4         /* delivery transaction */
#define STOTXN    5         /* stock level transaction */
#define ALLTXN    6         /* for processing all txns */
#define ALLTXNNODEL 7       /* for processing all txns except delivery */
/* 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;
   int ol_i_id[15]; //add
   int ol_supply_w_id[15];//add
   int ol_quantity[15];//add
};

struct newstruct {
   int    retval;
   int old_quantity[15];
   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;
  float 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 {
  int retval;
  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 {
  int retval;
  struct ordinstruct ordin;
  struct ordoutstruct ordout;
};


/* Delivery */

struct delinstruct {
  int w_id;
```

```
  int o_carrier_id;
  double qtime;
  int in_timing_int;
  int plsqlflag;
};

struct deloutstruct {
  int terror;
  int retry;
};

struct delstruct {
  int retval;
  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 {
  int retval;
  struct stoinstruct stoin;
  struct stooutstruct stoout;
};


/* used these definitions in client code only */
typedef struct delstruct  DeliveryData, *pDeliveryData;
typedef struct newstruct  NewOrderData, *pNewOrderData;
typedef struct paystruct  PaymentData, *pPaymentData;
typedef struct ordstruct  OrderStatusData, *pOrderStatusData;
typedef struct stostruct  StockLevelData, *pStockLevelData;


#endif
```

tpcc.h

```
/*
 * $Header: tpcc.h 7030100.1 95/07/19 15:10:55 plai Generic<base> $ Copyr (c)
1993 Oracle
 */
/*=============================================================
======+
|       Copyright (c) 1995  Oracle Corp, Redwood Shores, CA     |
|                 OPEN SYSTEMS PERFORMANCE GROUP                |
|                     All Rights Reserved                       |

+=============================================================
======+
| FILENAME
|   tpcc.h
| DESCRIPTION
|   Include file for TPC-C benchmark programs.

+=============================================================
======*/

#ifndef TPCC_H
#define TPCC_H

#ifndef FALSE
# define FALSE 0
#endif

#ifndef TRUE
# define TRUE 1
#endif

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
```

```
#include <string.h>

#ifndef boolean
#define boolean int
#endif

#include "tpccflags.h"

#include <oratypes.h>
#include <oci.h>
#include <ocidfn.h>
/*
#ifdef __STDC__
#include "ociapr.h"
#else
#include "ocikpr.h"
#endif
*/

typedef struct cda_def csrdef;
typedef struct cda_def ldadef;


/* TPC-C transaction functions */

extern int TPCinit ();
extern int TPCnew ();
extern int TPCpay ();
extern int TPCord ();
extern int TPCdel ();
extern int TPCsto ();
extern void TPCexit ();
extern int TPCdumpinit ();
extern void TPCdumpnew ();
extern void TPCdumppay ();
extern void TPCdumpord ();
extern void TPCdumpdel ();
extern void TPCdumpsto ();
extern void TPCdumpexit ();
//extern void userlog(char* fmtp, ...);


/* Error codes */

#define RECOVERR -10
#define IRRECERR -20
#define NOERR    111
#define DEL_ERROR -666
#define DEL_DATE_LEN 7
#define NDISTS 10
#define NITEMS 15
#define SQL_BUF_SIZE 8192

#define FULLDATE "dd-mon-yy.hh24:mi:ss"
#define SHORTDATE "dd-mm-yyyy"



#define DELRT 80.0

extern int tkvcninit ();
extern int tkvcpinit ();
extern int tkvcoinit ();
extern int tkvcdinit ();
extern int tkvcsinit ();

extern int tkvcn ();
extern int tkvcp ();
extern int tkvco ();
extern int tkvcd ();
extern int tkvcs ();

extern void tkvcndone ();
extern void tkvcpdone ();
extern void tkvcodone ();
extern void tkvcddone ();
extern void tkvcsdone ();

extern int tkvcss (); /* for alter session to get memory size and trace */
extern boolean multitranx;
extern int ord_init;


extern void errrpt ();
extern int ocierror(char *fname, int lineno,OCIError *errhp, sword status);
extern int sqlfile(char *fname, text *linebuf);
```

```
extern FILE *lfp;
extern FILE *fopen ();
extern int proc_no;
extern int doid[];

extern int execstatus;
extern int errcode;

extern OCIEnv *tpcenv;
extern OCIServer *tpcsrv;
extern OCIError *errhp;
extern OCISvcCtx *tpcsvc;
extern OCISession *tpcusr;
extern OCIStmt *curntest;
/* The bind and define handles for each transaction are
   included in their respective header files. */



/* for stock-level transaction */

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

/* for delivery transaction */

extern int del_o_id[10];
extern int carrier_id;
extern int retries;

/* for order-status transaction */

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

/* for payment transaction */

extern int c_w_id;
extern int c_d_id;
#ifdef USE_IEEE_NUMBER
extern float h_amount;
#else
extern int h_amount;
#endif /* USE_IEEE_NUMBER */
extern char w_street_1[21];
extern char w_street_2[21];
extern char w_city[21];
extern char w_state[3];
extern char w_zip[10];
extern char d_street_1[21];
extern char d_street_2[21];
extern char d_city[21];
extern char d_state[3];
extern char d_zip[10];
extern char c_street_1[21];
extern char c_street_2[21];
extern char c_city[21];
```

```
extern char c_state[3];
extern char c_zip[10];
extern char c_phone[17];
extern text c_since_d[11];
extern char c_credit[3];
extern int c_credit_lim;
extern float c_discount;
extern char c_data[201];
extern text h_date[20];

/* for new order transaction */

extern int nol_i_id[15];
extern int nol_supply_w_id[15];
#ifdef USE_IEEE_NUMBER
extern float nol_quantity[15];
extern float nol_amount[15];
extern float s_quantity[15];
extern float i_price[15];
#else
extern int nol_quantity[15];
extern int nol_amount[15];
extern int s_quantity[15];
extern int i_price[15];
#endif /* USE_IEEE_NUMBER */
extern int nol_quanti10[15];
extern int nol_quanti91[15];
extern int nol_ytdqty[15];
extern int o_all_local;
extern float w_tax;
extern float d_tax;
extern float total_amount;
extern char i_name[15][25];
extern int i_name_strlen[15];
extern ub2 i_name_strlen_len[15];
extern ub2 i_name_strlen_rcode[15];
extern ub4 i_name_strlen_csize;
extern char brand_gen[15];
extern ub2 brand_gen_len[15];
extern ub2 brand_gen_rcode[15];
extern ub4 brand_gen_csize;
extern char brand_generic[15][1];
extern int status;
extern int tracelevel;

extern ub2 rlenp_c_last;
extern ub2 rlenp_c_first;

/* Miscellaneous */
extern OCIDate cr_date;
extern OCIDate c_since;
extern OCIDate o_entry_d_base;
extern OCIDate ol_d_base[15];

#ifndef DISCARD
# define DISCARD (void)
#endif

#ifndef sword
# define sword int
#endif

#define VER7          2

#define NA            -1     /* ANSI SQL NULL */
#define NLT           1      /* length for string null terminator */
#define DEADLOCK      60     /* ORA-00060: deadlock */
#define NO_DATA_FOUND  1403   /* ORA-01403: no data found */
#define NOT_SERIALIZABLE  8177  /* ORA-08177: transaction not serializable */
#define SNAPSHOT_TOO_OLD  1555  /* ORA-01555: snapshot too old */

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

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

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

#define min(x,y)  (((x) < (y)) ? (x) : (y))

#define OCIERROR(errp,function)\
      ocierror(__FILE__,__LINE__,(errp),(function));
```

```
#define OCIBND(stmp, bndp, errp, sqlvar, progv, progvl, ftype)\
      ocierror(__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
      ocierror(__FILE__,__LINE__, (errp), \
           OCIBindByName((stmp), &(bndp), (errp), \
                        (text *)(sqlvar), strlen((sqlvar)),\
                        (progv), (progvl), (ftype),0,0,0,0,0,OCI_DEFAULT));



/* bind arrays for sql */
#define OCIBNDRA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
      DISCARD ocierror(__FILE__,__LINE__,(errp), \
       OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0));
\
      DISCARD ocierror(__FILE__,__LINE__,(errp), \
        OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\

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


/* use with callback data */
#define OCIBNDRAD(stmp,bndp,errp,sqlvar,progvl,ftype,indp,ctxp,\
               cbf_nodata,cbf_data) \
      DISCARD ocierror(__FILE__,__LINE__,(errp), \
       OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0));
\
      DISCARD ocierror(__FILE__,__LINE__,(errp), \
          OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar), \
                    strlen((sqlvar)),0,(progvl),(ftype), \
                    indp,0,0,0,0,OCI_DATA_AT_EXEC)); \
      DISCARD ocierror(__FILE__,__LINE__,(errp), \
        OCIBindDynamic((bndp),(errp),(ctxp),(cbf_nodata),(ctxp),(cbf_data)));



/* bind in/out for plsql without indicator and rcode */
#define OCIBNDPL(stmp,bndp,errp,sqlvar,progv,progvl,ftype,alen) \
    DISCARD ocierror(__FILE__,__LINE__,(errp), \
      OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0)); \
    DISCARD ocierror(__FILE__,__LINE__,(errp), \
      OCIBindByName((stmp),&(bndp),(errp),(CONST text *)(sqlvar), \
        (sb4)strlen((CONST char *)(sqlvar)), (dvoid*)(progv),(progvl),(ftype),\
            NULLP(dvoid),(alen), NULLP(ub2), 0,NULLP(ub4),OCI_DEFAULT));

/* bind  in values for plsql with indicator and rcode */
#define OCIBNDR(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode) \
      DISCARD ocierror(__FILE__,__LINE__,(errp), \
       OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0));
\
      DISCARD ocierror(__FILE__,__LINE__,(errp), \
       OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
                    (progv),(progvl),(ftype),(indp),(alen),(arcode),0,0, \
                OCI_DEFAULT));

/* bind in/out for plsql arrays witout indicator and rcode */
#define OCIBNDPLA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,alen,ms,cu) \
      DISCARD ocierror(__FILE__,__LINE__, (errp), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0));\
      DISCARD ocierror(__FILE__,__LINE__,(errp),\
        OCIBindByName((stmp),&(bndp),(errp),(CONST text *)(sqlvar), \
            (sb4)strlen((CONST char *) (sqlvar)),(void *)(progv), \
            (progvl),(ftype),NULL,(alen),NULL,(ms),(cu),OCI_DEFAULT));

/* bind in/out values for plsql with indicator and rcode */
#define OCIBNDRAA(stmp,bndp,errp,sqlvar,progv,progvl,ftype,indp,alen,arcode,\
              ms,cu) \
      ocierror(__FILE__,__LINE__, (errp), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoid**)0));\
      ocierror(__FILE__,__LINE__,(errp),\
        OCIBindByName((stmp),&(bndp),(errp),(text *)(sqlvar),strlen((sqlvar)),\
      (progv),(progvl),(ftype),(indp),(alen),(arcode),(ms),(cu),OCI_DEFAULT));

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


#define OCIDEF(stmp,dfnp,errp,pos,progv,progvl,ftype) \
      OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
                                          (dvoid**)0);\
      OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),(progvl),\
                  (ftype),NULL,NULL,NULL,OCI_DEFAULT); \
```

```
#define OCIDFNRA(stmp,dfnp,errp,pos,progv,progvl,ftype,indp,alen,arcode) \
      OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
                                          (dvoid**)0);\
      OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),\
                                          (progvl),(ftype),(indp),(alen),\
                                          (arcode),OCI_DEFAULT);

#define OCIDFNDYN(stmp,dfnp,errp,pos,progv,progvl,ftype,indp,ctxp,cbf_data) \
      ocierror(__FILE__,__LINE__,(errp), \
      OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
                                          (dvoid**)0));\
      ocierror(__FILE__,__LINE__,(errp), \
      OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv), (progvl),(ftype),\
                        (indp),NULL,NULL, OCI_DYNAMIC_FETCH));\
      ocierror(__FILE__,__LINE__,(errp), \
      OCIDefineDynamic((dfnp),(errp),(ctxp),(cbf_data)));


#ifndef TPCSVR

/* 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;
  int ol_i_id[15]; //add
  int ol_supply_w_id[15];//add
  int ol_quantity[15];//add
};

struct newstruct {
  int    retval;
  int old_quantity[15];
  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;
  float 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 {
  int retval;
  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 {
  int retval;
  struct ordinstruct ordin;
  struct ordoutstruct ordout;
};


/* Delivery */

struct delinstruct {
  int w_id;
  int o_carrier_id;
  double qtime;
  int in_timing_int;
  int plsqlflag;
};

struct deloutstruct {
  int terror;
  int retry;
};

struct delstruct {
  int retval;
  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 {
  int retval;
  struct stoinstruct stoin;
  struct stooutstruct stoout;
};


#endif

#endif
```

Makefile

```
#gcc=/usr/local/bin/gcc
#============================================================
======+
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                  |
#                   All Rights Reserved                          |
#============================================================
======+
# FILENAME
#      Makefile
# DESCRIPTION
#      Makefile for batch driver, load program and tx testing.
#============================================================
======
#
#  Programs:
#
#      tpcc.exe       : OCI TPC-C generator
#      tpccload.exe   : Database loader for TPC-C
#      single_txn.exe : OCI program to test the TPC-C transactions
#      getrand.exe    :
#      90per.exe      :
#      runtpb.exe     :
#      sleep.exe      :
#      press_return.exe     :
#      runid.exe      :
#

#all: compile load
all: generic

#include $(ORACLE_HOME)/bench/buildtools/prefix.mk

#I_SYM=-I
#include $(ORACLE_HOME)/rdbms/lib/env_rdbms.mk

LINK=gcc
#/opt/SunProd/SUNWspro6.1/bin/../WS6U1/bin/cc
CC=gcc
#/opt/SunProd/SUNWspro6.1/bin/../WS6U1/bin/cc

#LDFLAGS=-L$(ORACLE_HOME)/rdbms/lib/ -L$(ORACLE_HOME)/lib/ -dy \
#-L$(TUXDIR)/lib/ -L$(ORACLE_HOME)/rdbms/lib/ -L$(ORACLE_HOME)/lib/ \
#$(ORACLE_HOME)/rdbms/lib/defopt.o -lclntsh \
#`cat $(ORACLE_HOME)/lib/sysliblist` \
#-o $@

BENCHRUN_SRC_DIR=/home/oracle/tpcc-kit/benchrun/source
CFLAGS=-I$(TUXDIR)/include -DTUX -g -O2 -mcpu=nocona -march=nocona

LDFLAGS=-L$(ORACLE_HOME)/rdbms/lib/ -L$(ORACLE_HOME)/lib/ -dy \
-L$(TUXDIR)/lib/ -L$(ORACLE_HOME)/rdbms/lib/ -L$(ORACLE_HOME)/lib/ \
$(ORACLE_HOME)/rdbms/lib/defopt.o -lclntsh \
`cat $(ORACLE_HOME)/lib/sysliblist` \
-L$(ORACLE_HOME)/lib -laio -lm -o $@
#-R$(ORACLE_HOME)/lib -laio -lthread -lposix4 -lkstat -lm -o $@
```

```
I_SYM=-I

TARGS=compile cleanup
REMOVE=rm

#DPB_LIB_DIR=${BENCHRUN_SRC_DIR}/lib
#DPB_LIB=$(DPB_LIB_DIR)/dpblibunix.o

TPCBIN=../../bin
INCLUDE=$(I_SYM). $(I_SYM)$(ORACLE_HOME)/rdbms/demo \
 $(I_SYM)$(ORACLE_HOME)/rdbms/public \
 $(I_SYM)$(ORACLE_HOME)/rdbms/include \
 $(I_SYM)$(ORACLE_HOME)/plsql/public \
 $(I_SYM)$(ORACLE_HOME)/network/public
# $(I_SYM)$(DPB_LIB_DIR)
ITUX=$(I_SYM)$(ROOTDIR)/include

MEMBS=
OBJS=tpccload.o c_trans.o c_drv_o7.o c_dump.o tpccpl.o getrand.o 90per.o report.o
errrpt.o sleep.o press_return.o runid.o
CTRAN_OBJS=plnew.o plpay.o plord.o pldel.o plsto.o
CTRANPOCI_OBJS=plnew.o plpay_oci.o plord.o pldel.o plsto.o
CTRANTUX_OBJS=plnew_tux.o plpay.o plord.o pldel_tux.o plsto.o
OTHER_OBJS=c_drv_val.o test_drv.o test_sample.o test_tran.o single_txn_ran.o
TUX_OBJS=c_drv_tux.o tpccpl_tux.o tpccsvr.o tpccpl.o bs-mb.o

files:

compile: $(OBJS) $(DPB_LIB)
        @-$(DOTARGS)

load: $(TPCBIN)/tpcc.exe $(TPCBIN)/tpccload.exe \
        $(TPCBIN)/single_txn.exe $(TPCBIN)/90per.exe \
        $(TPCBIN)/runtpb.exe $(TPCBIN)/getrand.exe \
        $(TPCBIN)/sleep.exe $(TPCBIN)/runid.exe \
        $(TPCBIN)/single_txn_ran.exe \
        $(TPCBIN)/press_return.exe
        @-$(DOTARGS)

cleanup:
        $(REMOVE) $(OBJS) $(CTRAN_OBJS) $(CTRANTUX_OBJS)
$(OTHER_OBJS) \
        $(TPCBIN)/tpcc.exe $(TPCBIN)/tpccload.exe \
        $(TPCBIN)/single_txn.exe $(TPCBIN)/90per.exe \
        $(TPCBIN)/runtpb.exe $(TPCBIN)/getrand.exe \
        $(TPCBIN)/sleep.exe $(TPCBIN)/runid.exe \
        $(TPCBIN)/single_txn_ran.exe \
        $(TPCBIN)/press_return.exe \
        $(TUX_OBJS)
        @-$(DOTARGS)

#$(DPB_LIB):
#       ( cd $(DPB_LIB_DIR); $(MAKE) -f Makefile.unix )

report.o: report.c results.h
        $(CC) $(CFLAGS) $(INCLUDE) -c report.c

errrpt.o: errrpt.c results.h
        $(CC) $(CFLAGS) $(INCLUDE) -c errrpt.c

sleep.o: sleep.c
        $(CC) $(CFLAGS) $(INCLUDE) -c sleep.c

press_return.o: press_return.c
        $(CC) $(CFLAGS) $(INCLUDE) -c press_return.c

runid.o: runid.c
        $(CC) $(CFLAGS) $(INCLUDE) -c runid.c

tpccload.o: tpccload.c tpcc.h
        $(CC) $(CFLAGS) $(INCLUDE) -c tpccload.c

c_drv_o7.o: c_drv_o7.c tpcc.h
        $(CC) $(CFLAGS) $(INCLUDE) -c c_drv_o7.c

c_drv_val.o: c_drv.c tpcc.h
        $(CP) c_drv.c c_drv_val.c
        -$(CC) $(CFLAGS) -DVALIDATE $(INCLUDE) -c c_drv_val.c
        $(REMOVE) c_drv_val.c

c_drv_tux.o: c_drv.c tpcc.h
        $(CP) c_drv.c c_drv_tux.c
        -$(CC) $(CFLAGS) -DTUX $(INCLUDE) $(ITUX) -c c_drv_tux.c
        $(REMOVE) c_drv_tux.c
```

```
c_dump.o: c_dump.c tpcc.h
	$(CC) $(CFLAGS) $(INCLUDE) -c c_dump.c

single_txn.o: single_txn.c tpcc.h
	$(CC) $(CFLAGS) $(INCLUDE) -c single_txn.c

single_txn_ran.o: single_txn_ran.c tpcc.h
	$(CC) $(CFLAGS) $(INCLUDE) -c single_txn_ran.c

runtpb.o: runtpb.c
	$(CC) $(CFLAGS) -DORA_AUX $(INCLUDE) -c runtpb.c

c_trans.o: $(CTRAN_OBJS)
	$(LD) -r -o $@ $(CTRAN_OBJS)

c_trans_tux.o: $(CTRANTUX_OBJS)
	$(LD) -r -o $@ $(CTRANTUX_OBJS)

tpccpl.o: tpccpl.c tpcc.h
	$(CC) $(CFLAGS) $(INCLUDE) -c tpccpl.c

tpccpl_tux.o: tpccpl.c tpcc.h
	$(CP) tpccpl.c tpccpl_tux.c
	-$(CC) $(CFLAGS) -DTUX $(INCLUDE) $(ITUX) -c tpccpl_tux.c
	$(REMOVE) tpccpl_tux.c

plnew_tux.o: plnew.c tpcc.h
	$(CP) plnew.c plnew_tux.c
	-$(CC) $(CFLAGS) -DTUX $(INCLUDE) $(ITUX) -c plnew_tux.c
	$(REMOVE) plnew_tux.c

plnew.o: plnew.c tpcc.h
	$(CC) $(CFLAGS) $(INCLUDE) -c plnew.c

plpay.o: plpay.c tpcc.h
	$(CC) $(CFLAGS) $(INCLUDE) -c plpay.c

plord.o: plord.c tpcc.h
	$(CC) $(CFLAGS) $(INCLUDE) -c plord.c

pldel_tux.o: pldel.c tpcc.h
	$(CP) pldel.c pldel_tux.c
	-$(CC) $(CFLAGS) -DTUX $(INCLUDE) $(ITUX) -c pldel_tux.c
	$(REMOVE) pldel_tux.c

pldel.o: pldel.c tpcc.h
	$(CC) $(CFLAGS) $(INCLUDE) -c pldel.c

plsto.o: plsto.c tpcc.h
	$(CC) $(CFLAGS) $(INCLUDE) -c plsto.c

tpccsvr.o: tpccsvr.c tpcc.h
	$(CC) $(CFLAGS) $(INCLUDE) $(ITUX) -c tpccsvr.c

bs-mb.o:  bs-mb.c
	$(CC) $(CFLAGS) $(INCLUDE) $(ITUX) -c bs-mb.c

generic : $(CTRAN_OBJS) tpccsvr.o bs-mb.o tpccpl.o
	$(LINK) $(LDFLAGS) \
    -ltux -lbuft -lfml -lfml32 -lengine -ldl -lpthread /usr/lib/libcrypt.a \
    tpccsvr.o bs-mb.o tpccpl.o plnew.o pldel.o plsto.o plord.o plpay.o \
	$(SSABED) $(DEF_OPT) $(TTLIBS) -lc


getrand.o: getrand.c
	$(CC) $(CFLAGS) $(INCLUDE) -c getrand.c

90per.o: 90per.c
	$(CC) $(CFLAGS) $(INCLUDE) -c 90per.c

$(TPCBIN)/getrand.exe: getrand.o
	$(LINK) $(LDFLAGS) \
	getrand.o -lc

$(TPCBIN)/sleep.exe: sleep.o
	$(LINK) $(LDFLAGS) \
	sleep.o -lc

$(TPCBIN)/press_return.exe: press_return.o
	$(LINK) $(LDFLAGS) \
	press_return.o -lc

$(TPCBIN)/runid.exe: runid.o
	$(LINK) $(LDFLAGS) \
	runid.o $(DPB_LIB) -lc
```

```
$(TPCBIN)/90per.exe: 90per.o
        $(LINK) $(LDFLAGS) \
        90per.o -lc

$(TPCBIN)/tpccload.exe: tpccload.o $(DPB_LIB)
        $(LINK) $(LDFLAGS) \
        tpccload.o $(DPB_LIB) \
        $(SSABED) $(DEF_OPT) $(TTLIBS) -lc

$(TPCBIN)/runtpb.exe: runtpb.o $(DPB_LIB)
        $(LINK) $(LDFLAGS) \
        runtpb.o $(DPB_LIB) \
        $(SSABED) $(DEF_OPT) $(TTLIBS) -lc

$(TPCBIN)/tpcc.exe: c_drv_o7.o c_trans.o tpccpl.o c_dump.o report.o errrpt.o
$(DPB_LIB)
        $(LINK) $(LDFLAGS) \
        c_drv_o7.o c_trans.o tpccpl.o c_dump.o errrpt.o report.o $(DPB_LIB)\
    $(SSABED) $(DEF_OPT) $(TTLIBS) -lc

$(TPCBIN)/single_txn.exe: single_txn.o $(DPB_LIB) c_trans.o tpccpl.o c_dump.o
        $(LINK) $(LDFLAGS) \
        single_txn.o c_trans.o tpccpl.o c_dump.o $(DPB_LIB) \
        $(SSABED) $(DEF_OPT) $(TTLIBS) -lc

$(TPCBIN)/single_txn_ran.exe: single_txn_ran.o $(DPB_LIB) c_trans.o tpccpl.o
c_dump.o
        $(LINK) $(LDFLAGS) \
        single_txn_ran.o c_trans.o tpccpl.o c_dump.o $(DPB_LIB) \
        $(SSABED) $(DEF_OPT) $(TTLIBS) –lc
```

paynz.sql

```
DECLARE /* paynz */
    not_serializable        EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_serializable,-8177);
    deadlock                EXCEPTION;
    PRAGMA EXCEPTION_INIT(deadlock,-60);
    snapshot_too_old        EXCEPTION;
    PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
  BEGIN
    LOOP BEGIN
```

```
  UPDATE ware
     SET w_ytd = w_ytd + :h_amount
   WHERE w_id = :w_id
RETURNING w_name, w_street_1, w_street_2, w_city, w_state, w_zip
    INTO inittpcc.ware_name, :w_street_1, :w_street_2, :w_city,
        :w_state, :w_zip;

 UPDATE  cust
    SET  c_balance = c_balance - :h_amount,
        c_ytd_payment = c_ytd_payment + :h_amount,
        c_payment_cnt = c_payment_cnt+1
 WHERE    c_id = :c_id AND c_d_id = :c_d_id AND
        c_w_id = :c_w_id
RETURNING rowid, c_first, c_middle, c_last, c_street_1,
        c_street_2, c_city, c_state, c_zip, c_phone,
        c_since, c_credit, c_credit_lim,
        c_discount, c_balance
    INTO inittpcc.cust_rowid,:c_first, :c_middle, :c_last, :c_street_1,
        :c_street_2, :c_city, :c_state, :c_zip, :c_phone,
        :c_since, :c_credit, :c_credit_lim,
        :c_discount, :c_balance;
  IF SQL%NOTFOUND THEN
    raise NO_DATA_FOUND;
  END IF;


IF :c_credit = 'BC' THEN
  UPDATE cust
         SET c_data = substr ((to_char (:c_id) || ' ' ||
                    to_char (:c_d_id) || ' ' ||
                    to_char (:c_w_id) || ' ' ||
                    to_char (:d_id) || ' ' ||
                    to_char (:w_id) || ' ' ||
                    to_char (:h_amount/100, '9999.99') || ' | ')
                    || c_data, 1, 500)
    WHERE rowid = inittpcc.cust_rowid
RETURNING substr(c_data,1, 200)
    INTO :c_data;

END IF;

  UPDATE dist
     SET d_ytd = d_ytd + :h_amount
```

```
      WHERE d_id = :d_id
        AND d_w_id = :w_id
    RETURNING d_name, d_street_1, d_street_2, d_city,d_state, d_zip
      INTO inittpcc.dist_name,:d_street_1,:d_street_2,:d_city,:d_state,
           :d_zip;
    IF SQL%NOTFOUND THEN
     raise NO_DATA_FOUND;
    END IF;


    INSERT INTO hist  (h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
                       h_amount, h_date, h_data)
    VALUES
      (:c_id, :c_d_id, :c_w_id, :d_id, :w_id, :h_amount,
       :cr_date, inittpcc.ware_name || '    ' || inittpcc.dist_name);
    EXIT;

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

  END LOOP;
END;
```

payz.sql

```
DECLARE /* payz */
    not_serializable        EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_serializable,-8177);
    deadlock                EXCEPTION;
    PRAGMA EXCEPTION_INIT(deadlock,-60);
    snapshot_too_old        EXCEPTION;
    PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
  BEGIN
    LOOP BEGIN
      UPDATE ware
        SET w_ytd = w_ytd+:h_amount
        WHERE w_id = :w_id
        RETURNING w_name,
              w_street_1, w_street_2, w_city, w_state, w_zip
      INTO inittpcc.ware_name,
          :w_street_1, :w_street_2, :w_city, :w_state, :w_zip;


 SELECT rowid
 BULK COLLECT INTO inittpcc.row_id
 FROM cust
 WHERE c_d_id = :c_d_id AND c_w_id = :c_w_id AND c_last = :c_last
 ORDER BY c_last, c_d_id, c_w_id, c_first;

inittpcc.c_num := sql%rowcount;
inittpcc.cust_rowid := inittpcc.row_id((inittpcc.c_num) / 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;
```

tkvcinin.sql

```
-- The initnew package for storing variables used in the
-- New Order anonymous block

CREATE OR REPLACE PACKAGE inittpcc
AS
 TYPE intarray IS TABLE OF INTEGER INDEX BY BINARY_INTEGER;
 TYPE distarray IS TABLE OF VARCHAR(24) INDEX BY BINARY_INTEGER;
 nulldate       DATE;
 TYPE rowidarray IS TABLE OF ROWID INDEX BY PLS_INTEGER;
 s_dist                    distarray;
 idx1arr            intarray;
 s_remote                  intarray;
 dist               intarray;
 row_id              rowidarray;
 cust_rowid           rowid;
 dist_name           VARCHAR2(11);
 ware_name            VARCHAR2(11);
 c_num              PLS_INTEGER;

 PROCEDURE init_no(idxarr intarray);
 PROCEDURE init_del;
 PROCEDURE init_pay;
END inittpcc;
/
show errors;

CREATE OR REPLACE PACKAGE BODY inittpcc AS
  PROCEDURE init_no (idxarr  intarray)
  IS
  BEGIN
       -- initialize null date
   nulldate := TO_DATE('01-01-1811', 'MM-DD-YYYY');
    idx1arr := idxarr;
  END init_no;

  PROCEDURE init_del
  IS
  BEGIN
    FOR i IN 1 .. 10 LOOP
      dist(i) := i;
    END LOOP;
  END init_del;

  PROCEDURE init_pay IS
  BEGIN
    NULL;
  END init_pay;

END inittpcc;
/
show errors
exit
```

tkvcpdel.sql

```
declare
  TYPE numarray IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
  TYPE numlist is varray (10) of number;
  dist numarray;
  amt numarray ;
  cnt pls_integer;

  not_serializable EXCEPTION;
  PRAGMA EXCEPTION_INIT(not_serializable, -8177);
  deadlock        EXCEPTION;
  PRAGMA EXCEPTION_INIT(deadlock, -60);
  snapshot_too_old EXCEPTION;
  PRAGMA EXCEPTION_INIT(snapshot_too_old, -1555);

BEGIN
  LOOP BEGIN
    FORALL d IN 1..10
      DELETE FROM nord N
        WHERE no_d_id = inittpcc.dist(d)
          AND no_w_id = :w_id
          AND no_o_id = (select min(no_o_id)
                         from nord
                         where no_d_id = N.no_d_id
                         and no_w_id = N.no_w_id)
      RETURNING no_d_id, no_o_id BULK COLLECT INTO :d_id, :order_id;

    :ordcnt := SQL%ROWCOUNT;

    FORALL o in 1.. :ordcnt
      UPDATE ordr SET o_carrier_id = :carrier_id
      WHERE o_id = :order_id (o)
        AND o_d_id = :d_id(o)
        AND o_w_id = :w_id
      RETURNING o_c_id BULK COLLECT INTO :o_c_id;

    FORALL o in 1.. :ordcnt
      UPDATE ordl SET ol_delivery_d = :now
      WHERE ol_w_id = :w_id
        AND ol_d_id = :d_id(o)
        AND ol_o_id = :order_id(o)
      RETURNING sum(ol_amount) BULK COLLECT INTO  :sums;

    FORALL c IN 1.. :ordcnt
      UPDATE cust
        SET c_balance = c_balance + :sums(c),
                    c_delivery_cnt = c_delivery_cnt + 1
       WHERE c_w_id = :w_id
         AND c_d_id = :d_id(c)
         AND c_id = :o_c_id(c);
    COMMIT;
    EXIT;
    EXCEPTION
      WHEN not_serializable OR deadlock OR snapshot_too_old
      THEN
        ROLLBACK;
        :retry := :retry + 1;
    END;

  END LOOP; -- for retry
END;
```

tkvcpnew.sql

```
-- New Order Anonymous block

 DECLARE
    idx                 PLS_INTEGER;
    dummy_local           PLS_INTEGER;
    cache_ol_cnt          PLS_INTEGER;
    not_serializable      EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_serializable,-8177);
    deadlock              EXCEPTION;
    PRAGMA EXCEPTION_INIT(deadlock,-60);
    snapshot_too_old      EXCEPTION;
    PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);

  PROCEDURE u1 IS
  BEGIN
     FORALL idx IN 1 .. cache_ol_cnt
       UPDATE  stock_item
       SET s_order_cnt = s_order_cnt + 1,
```

```
    s_ytd = s_ytd + :ol_quantity(idx),
    s_remote_cnt = s_remote_cnt + :s_remote(idx),
    s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                  THEN s_quantity +91
                  ELSE s_quantity
                END) - :ol_quantity(idx)
    WHERE i_id = :ol_i_id(idx)
    AND s_w_id = :ol_supply_w_id(idx)
    RETURNING i_price, i_name, s_quantity, s_dist_01,
          i_price*:ol_quantity(idx),
     CASE WHEN i_data NOT LIKE '%ORIGINAL%'
         THEN 'G'
          ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
               THEN 'G'
               ELSE 'B'
               END)
      END
   BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u1;

PROCEDURE u2 IS
BEGIN
   FORALL idx IN 1 .. cache_ol_cnt
    UPDATE  stock_item
    SET s_order_cnt = s_order_cnt + 1,
    s_ytd = s_ytd + :ol_quantity(idx),
    s_remote_cnt = s_remote_cnt + :s_remote(idx),
    s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                  THEN s_quantity +91
                  ELSE s_quantity
                END) - :ol_quantity(idx)
    WHERE i_id = :ol_i_id(idx)
    AND s_w_id = :ol_supply_w_id(idx)
    RETURNING i_price, i_name, s_quantity, s_dist_02,
          i_price*:ol_quantity(idx),
     CASE WHEN i_data NOT LIKE '%ORIGINAL%'
         THEN 'G'
          ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
               THEN 'G'
               ELSE 'B'
               END)
      END
   BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u2;

PROCEDURE u3 IS
BEGIN
   FORALL idx IN 1 .. cache_ol_cnt
    UPDATE  stock_item
    SET s_order_cnt = s_order_cnt + 1,
    s_ytd = s_ytd + :ol_quantity(idx),
    s_remote_cnt = s_remote_cnt + :s_remote(idx),
    s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                  THEN s_quantity +91
                  ELSE s_quantity
                END) - :ol_quantity(idx)
    WHERE i_id = :ol_i_id(idx)
    AND s_w_id = :ol_supply_w_id(idx)
    RETURNING i_price, i_name, s_quantity, s_dist_03,
          i_price*:ol_quantity(idx),
     CASE WHEN i_data NOT LIKE '%ORIGINAL%'
         THEN 'G'
          ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
               THEN 'G'
               ELSE 'B'
               END)
      END
   BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u3;

PROCEDURE u4 IS
BEGIN
   FORALL idx IN 1 .. cache_ol_cnt
    UPDATE  stock_item
    SET s_order_cnt = s_order_cnt + 1,
    s_ytd = s_ytd + :ol_quantity(idx),
    s_remote_cnt = s_remote_cnt + :s_remote(idx),
    s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                  THEN s_quantity +91
                  ELSE s_quantity
                END) - :ol_quantity(idx)
    WHERE i_id = :ol_i_id(idx)
    AND s_w_id = :ol_supply_w_id(idx)
```

```
      RETURNING i_price, i_name, s_quantity, s_dist_04,
              i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                 THEN 'G'
                 ELSE 'B'
                 END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                  :ol_amount,:brand_generic;
END u4;

PROCEDURE u5 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE  stock_item
      SET s_order_cnt = s_order_cnt + 1,
      s_ytd = s_ytd + :ol_quantity(idx),
      s_remote_cnt = s_remote_cnt + :s_remote(idx),
      s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      WHERE i_id = :ol_i_id(idx)
      AND s_w_id = :ol_supply_w_id(idx)
      RETURNING i_price, i_name, s_quantity, s_dist_05,
              i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                 THEN 'G'
                 ELSE 'B'
                 END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                  :ol_amount,:brand_generic;
END u5;

PROCEDURE u6 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE  stock_item
      SET s_order_cnt = s_order_cnt + 1,
      s_ytd = s_ytd + :ol_quantity(idx),
      s_remote_cnt = s_remote_cnt + :s_remote(idx),
      s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      WHERE i_id = :ol_i_id(idx)
      AND s_w_id = :ol_supply_w_id(idx)
      RETURNING i_price, i_name, s_quantity, s_dist_06,
              i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                 THEN 'G'
                 ELSE 'B'
                 END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                  :ol_amount,:brand_generic;
END u6;

PROCEDURE u7 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE  stock_item
      SET s_order_cnt = s_order_cnt + 1,
      s_ytd = s_ytd + :ol_quantity(idx),
      s_remote_cnt = s_remote_cnt + :s_remote(idx),
      s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      WHERE i_id = :ol_i_id(idx)
      AND s_w_id = :ol_supply_w_id(idx)
      RETURNING i_price, i_name, s_quantity, s_dist_07,
              i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                 THEN 'G'
                 ELSE 'B'
                 END)
```

```
      END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u7;

PROCEDURE u8 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE  stock_item
      SET s_order_cnt = s_order_cnt + 1,
      s_ytd = s_ytd + :ol_quantity(idx),
      s_remote_cnt = s_remote_cnt + :s_remote(idx),
      s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      WHERE i_id = :ol_i_id(idx)
      AND s_w_id = :ol_supply_w_id(idx)
      RETURNING i_price, i_name, s_quantity, s_dist_08,
             i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                 THEN 'G'
                 ELSE 'B'
                 END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u8;

PROCEDURE u9 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE  stock_item
      SET s_order_cnt = s_order_cnt + 1,
      s_ytd = s_ytd + :ol_quantity(idx),
      s_remote_cnt = s_remote_cnt + :s_remote(idx),
      s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      WHERE i_id = :ol_i_id(idx)
      AND s_w_id = :ol_supply_w_id(idx)
      RETURNING i_price, i_name, s_quantity, s_dist_09,
             i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                 THEN 'G'
                 ELSE 'B'
                 END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u9;

PROCEDURE u10 IS
BEGIN
    FORALL idx IN 1 .. cache_ol_cnt
      UPDATE  stock_item
      SET s_order_cnt = s_order_cnt + 1,
      s_ytd = s_ytd + :ol_quantity(idx),
      s_remote_cnt = s_remote_cnt + :s_remote(idx),
      s_quantity = (CASE WHEN s_quantity < :ol_quantity (idx) + 10
                    THEN s_quantity +91
                    ELSE s_quantity
                END) - :ol_quantity(idx)
      WHERE i_id = :ol_i_id(idx)
      AND s_w_id = :ol_supply_w_id(idx)
      RETURNING i_price, i_name, s_quantity, s_dist_10,
             i_price*:ol_quantity(idx),
       CASE WHEN i_data NOT LIKE '%ORIGINAL%'
            THEN 'G'
            ELSE (CASE WHEN s_data NOT LIKE '%ORIGINAL%'
                 THEN 'G'
                 ELSE 'B'
                 END)
       END
    BULK COLLECT INTO :i_price, :i_name, :s_quantity, inittpcc.s_dist,
                :ol_amount,:brand_generic;
END u10;

PROCEDURE fix_items IS
  rows_lost                PLS_INTEGER;
  max_index                PLS_INTEGER;
```

```
  temp_index                PLS_INTEGER;
BEGIN
  idx := 1;
  rows_lost := 0;
  max_index := dummy_local;

  WHILE (max_index != cache_ol_cnt) LOOP

   WHILE (idx <= sql%rowcount AND
              sql%bulk_rowcount(idx + rows_lost) = 1)
   LOOP
     idx := idx + 1;
   END LOOP;

   temp_index := max_index;
   WHILE (temp_index >= idx + rows_lost) LOOP
     :ol_amount(temp_index + 1)      :=  :ol_amount(temp_index);
     :i_price(temp_index + 1)        :=  :i_price(temp_index);
     :i_name(temp_index + 1)         :=  :i_name(temp_index);
     :s_quantity(temp_index + 1)     :=  :s_quantity(temp_index);
     inittpcc.s_dist(temp_index + 1) :=  inittpcc.s_dist(temp_index);
     :brand_generic(temp_index + 1)  :=  :brand_generic(temp_index);
     temp_index := temp_index - 1;
   END LOOP;

   IF (idx + rows_lost <= cache_ol_cnt) THEN
     :i_price(idx + rows_lost)       :=   0;
     :i_name(idx + rows_lost)        :=   'NO ITEM';
     :s_quantity(idx + rows_lost)    :=   0;
     inittpcc.s_dist(idx + rows_lost) := NULL;
     :brand_generic(idx + rows_lost)  := ' ';
     :ol_amount(idx + rows_lost)     := 0;
     rows_lost := rows_lost + 1;
     max_index := max_index + 1;
   END IF;

 END LOOP;
END fix_items;

BEGIN
   LOOP BEGIN
      cache_ol_cnt := :o_ol_cnt;
```

```
UPDATE dist SET d_next_o_id = d_next_o_id + 1
 WHERE d_id = :d_id AND  d_w_id = :w_id
 RETURNING d_tax, d_next_o_id-1
 INTO :d_tax, :o_id;

SELECT c_discount, c_last, c_credit, w_tax
 INTO :c_discount, :c_last, :c_credit , :w_tax
 FROM cust , ware
 WHERE c_id = :c_id AND c_d_id = :d_id AND c_w_id = w_id
 AND  w_id = :w_id;


INSERT INTO nord (no_o_id, no_d_id, no_w_id)
 VALUES (:o_id, :d_id, :w_id);

INSERT INTO ordr  (o_id,o_d_id, o_w_id, o_c_id, o_entry_d,
                   o_carrier_id, o_ol_cnt, o_all_local)
 VALUES (:o_id, :d_id, :w_id, :c_id,
        :cr_date, 11, :o_ol_cnt, :o_all_local);


dummy_local :=  :d_id;

IF (dummy_local < 6) THEN
  IF (dummy_local < 3) THEN
    IF (dummy_local = 1) THEN
      u1;
    ELSE
      u2;
    END IF;
  ELSE
    IF (dummy_local = 3) THEN
      u3;
    ELSIF (dummy_local = 4) then
      u4;
    ELSE
      u5;
    END IF;
  END IF;
ELSE
  IF (dummy_local < 8) THEN
    IF (dummy_local = 6) THEN
      u6;
```

```
      ELSE
        u7;
      END IF;
     ELSE
      IF (dummy_local = 8) THEN
        u8;
      ELSIF (dummy_local = 9) then
        u9;
      ELSE
        u10;
      END IF;
    END IF;
  END IF;

  dummy_local := sql%rowcount;

  IF (dummy_local != cache_ol_cnt )  THEN fix_items; END IF;

  FORALL idx IN 1..dummy_local
   INSERT INTO ordl
     (ol_o_id, ol_d_id, ol_w_id, ol_number, ol_delivery_d, ol_i_id,
          ol_supply_w_id, ol_quantity,ol_amount,ol_dist_info)
   VALUES (:o_id, :d_id, :w_id, inittpcc.idx1arr(idx), inittpcc.nulldate,
         :ol_i_id(idx), :ol_supply_w_id(idx),
         :ol_quantity(idx), :ol_amount(idx), inittpcc.s_dist(idx));

  IF (dummy_local != :o_ol_cnt) THEN
    :o_ol_cnt := dummy_local;
    ROLLBACK;
  END IF;

EXIT;

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

Appendix B : Database Design

B1. Database build scripts

rawmap.sh

```
raw /dev/raw/raw28 /dev/sdai1
raw /dev/raw/raw62 /dev/sdai2
raw /dev/raw/raw96 /dev/sdai3
raw /dev/raw/raw130 /dev/sdai5
raw /dev/raw/raw164 /dev/sdai6
raw /dev/raw/raw198 /dev/sdai7
raw /dev/raw/raw232 /dev/sdai8
raw /dev/raw/raw266 /dev/sdai9
raw /dev/raw/raw300 /dev/sdai10
raw /dev/raw/raw334 /dev/sdai11
raw /dev/raw/raw368 /dev/sdai12
raw /dev/raw/raw402 /dev/sdai13
raw /dev/raw/raw29 /dev/sdaj1
raw /dev/raw/raw63 /dev/sdaj2
raw /dev/raw/raw97 /dev/sdaj3
raw /dev/raw/raw131 /dev/sdaj5
raw /dev/raw/raw165 /dev/sdaj6
raw /dev/raw/raw199 /dev/sdaj7
raw /dev/raw/raw233 /dev/sdaj8
raw /dev/raw/raw267 /dev/sdaj9
raw /dev/raw/raw301 /dev/sdaj10
raw /dev/raw/raw335 /dev/sdaj11
raw /dev/raw/raw369 /dev/sdaj12
raw /dev/raw/raw403 /dev/sdaj13
raw /dev/raw/raw30 /dev/sdak1
raw /dev/raw/raw64 /dev/sdak2
raw /dev/raw/raw98 /dev/sdak3
raw /dev/raw/raw132 /dev/sdak5
raw /dev/raw/raw166 /dev/sdak6
raw /dev/raw/raw200 /dev/sdak7
raw /dev/raw/raw234 /dev/sdak8
raw /dev/raw/raw268 /dev/sdak9
raw /dev/raw/raw302 /dev/sdak10
raw /dev/raw/raw336 /dev/sdak11
```

raw /dev/raw/raw370 /dev/sdak12
raw /dev/raw/raw404 /dev/sdak13
raw /dev/raw/raw31 /dev/sdam1
raw /dev/raw/raw65 /dev/sdam2
raw /dev/raw/raw99 /dev/sdam3
raw /dev/raw/raw133 /dev/sdam5
raw /dev/raw/raw167 /dev/sdam6
raw /dev/raw/raw201 /dev/sdam7
raw /dev/raw/raw235 /dev/sdam8
raw /dev/raw/raw269 /dev/sdam9
raw /dev/raw/raw303 /dev/sdam10
raw /dev/raw/raw337 /dev/sdam11
raw /dev/raw/raw371 /dev/sdam12
raw /dev/raw/raw405 /dev/sdam13
raw /dev/raw/raw32 /dev/sdan1
raw /dev/raw/raw66 /dev/sdan2
raw /dev/raw/raw100 /dev/sdan3
raw /dev/raw/raw134 /dev/sdan5
raw /dev/raw/raw168 /dev/sdan6
raw /dev/raw/raw202 /dev/sdan7
raw /dev/raw/raw236 /dev/sdan8
raw /dev/raw/raw270 /dev/sdan9
raw /dev/raw/raw304 /dev/sdan10
raw /dev/raw/raw338 /dev/sdan11
raw /dev/raw/raw372 /dev/sdan12
raw /dev/raw/raw406 /dev/sdan13
raw /dev/raw/raw33 /dev/sdap1
raw /dev/raw/raw67 /dev/sdap2
raw /dev/raw/raw101 /dev/sdap3
raw /dev/raw/raw135 /dev/sdap5
raw /dev/raw/raw169 /dev/sdap6
raw /dev/raw/raw203 /dev/sdap7
raw /dev/raw/raw237 /dev/sdap8
raw /dev/raw/raw271 /dev/sdap9
raw /dev/raw/raw305 /dev/sdap10
raw /dev/raw/raw339 /dev/sdap11
raw /dev/raw/raw373 /dev/sdap12
raw /dev/raw/raw407 /dev/sdap13
raw /dev/raw/raw34 /dev/sdaq1
raw /dev/raw/raw68 /dev/sdaq2
raw /dev/raw/raw102 /dev/sdaq3
raw /dev/raw/raw136 /dev/sdaq5
raw /dev/raw/raw170 /dev/sdaq6
raw /dev/raw/raw204 /dev/sdaq7
raw /dev/raw/raw238 /dev/sdaq8
raw /dev/raw/raw272 /dev/sdaq9
raw /dev/raw/raw306 /dev/sdaq10
raw /dev/raw/raw340 /dev/sdaq11
raw /dev/raw/raw374 /dev/sdaq12
raw /dev/raw/raw408 /dev/sdaq13
raw /dev/raw/raw35 /dev/sdar1
raw /dev/raw/raw69 /dev/sdar2
raw /dev/raw/raw103 /dev/sdar3
raw /dev/raw/raw137 /dev/sdar5
raw /dev/raw/raw171 /dev/sdar6
raw /dev/raw/raw205 /dev/sdar7
raw /dev/raw/raw239 /dev/sdar8
raw /dev/raw/raw273 /dev/sdar9
raw /dev/raw/raw307 /dev/sdar10
raw /dev/raw/raw341 /dev/sdar11
raw /dev/raw/raw375 /dev/sdar12
raw /dev/raw/raw409 /dev/sdar13
raw /dev/raw/raw36 /dev/sdas1
raw /dev/raw/raw70 /dev/sdas2
raw /dev/raw/raw104 /dev/sdas3
raw /dev/raw/raw138 /dev/sdas5
raw /dev/raw/raw172 /dev/sdas6
raw /dev/raw/raw206 /dev/sdas7
raw /dev/raw/raw240 /dev/sdas8
raw /dev/raw/raw274 /dev/sdas9
raw /dev/raw/raw308 /dev/sdas10
raw /dev/raw/raw342 /dev/sdas11
raw /dev/raw/raw376 /dev/sdas12
raw /dev/raw/raw410 /dev/sdas13
raw /dev/raw/raw1 /dev/sdb1
raw /dev/raw/raw2 /dev/sdc1
raw /dev/raw/raw3 /dev/sdd1
raw /dev/raw/raw37 /dev/sdd2
raw /dev/raw/raw71 /dev/sdd3
raw /dev/raw/raw105 /dev/sdd5
raw /dev/raw/raw139 /dev/sdd6
raw /dev/raw/raw173 /dev/sdd7
raw /dev/raw/raw207 /dev/sdd8
raw /dev/raw/raw241 /dev/sdd9
raw /dev/raw/raw275 /dev/sdd10
raw /dev/raw/raw309 /dev/sdd11

raw /dev/raw/raw343 /dev/sdd12
raw /dev/raw/raw377 /dev/sdd13
raw /dev/raw/raw4 /dev/sde1
raw /dev/raw/raw38 /dev/sde2
raw /dev/raw/raw72 /dev/sde3
raw /dev/raw/raw106 /dev/sde5
raw /dev/raw/raw140 /dev/sde6
raw /dev/raw/raw174 /dev/sde7
raw /dev/raw/raw208 /dev/sde8
raw /dev/raw/raw242 /dev/sde9
raw /dev/raw/raw276 /dev/sde10
raw /dev/raw/raw310 /dev/sde11
raw /dev/raw/raw344 /dev/sde12
raw /dev/raw/raw378 /dev/sde13
raw /dev/raw/raw5 /dev/sdf1
raw /dev/raw/raw39 /dev/sdf2
raw /dev/raw/raw73 /dev/sdf3
raw /dev/raw/raw107 /dev/sdf5
raw /dev/raw/raw141 /dev/sdf6
raw /dev/raw/raw175 /dev/sdf7
raw /dev/raw/raw209 /dev/sdf8
raw /dev/raw/raw243 /dev/sdf9
raw /dev/raw/raw277 /dev/sdf10
raw /dev/raw/raw311 /dev/sdf11
raw /dev/raw/raw345 /dev/sdf12
raw /dev/raw/raw379 /dev/sdf13
raw /dev/raw/raw6 /dev/sdg1
raw /dev/raw/raw40 /dev/sdg2
raw /dev/raw/raw74 /dev/sdg3
raw /dev/raw/raw108 /dev/sdg5
raw /dev/raw/raw142 /dev/sdg6
raw /dev/raw/raw176 /dev/sdg7
raw /dev/raw/raw210 /dev/sdg8
raw /dev/raw/raw244 /dev/sdg9
raw /dev/raw/raw278 /dev/sdg10
raw /dev/raw/raw312 /dev/sdg11
raw /dev/raw/raw346 /dev/sdg12
raw /dev/raw/raw380 /dev/sdg13
raw /dev/raw/raw7 /dev/sdi1
raw /dev/raw/raw41 /dev/sdi2
raw /dev/raw/raw75 /dev/sdi3
raw /dev/raw/raw109 /dev/sdi5
raw /dev/raw/raw143 /dev/sdi6
raw /dev/raw/raw177 /dev/sdi7
raw /dev/raw/raw211 /dev/sdi8
raw /dev/raw/raw245 /dev/sdi9
raw /dev/raw/raw279 /dev/sdi10
raw /dev/raw/raw313 /dev/sdi11
raw /dev/raw/raw347 /dev/sdi12
raw /dev/raw/raw381 /dev/sdi13
raw /dev/raw/raw8 /dev/sdj1
raw /dev/raw/raw42 /dev/sdj2
raw /dev/raw/raw76 /dev/sdj3
raw /dev/raw/raw110 /dev/sdj5
raw /dev/raw/raw144 /dev/sdj6
raw /dev/raw/raw178 /dev/sdj7
raw /dev/raw/raw212 /dev/sdj8
raw /dev/raw/raw246 /dev/sdj9
raw /dev/raw/raw280 /dev/sdj10
raw /dev/raw/raw314 /dev/sdj11
raw /dev/raw/raw348 /dev/sdj12
raw /dev/raw/raw382 /dev/sdj13
raw /dev/raw/raw9 /dev/sdk1
raw /dev/raw/raw43 /dev/sdk2
raw /dev/raw/raw77 /dev/sdk3
raw /dev/raw/raw111 /dev/sdk5
raw /dev/raw/raw145 /dev/sdk6
raw /dev/raw/raw179 /dev/sdk7
raw /dev/raw/raw213 /dev/sdk8
raw /dev/raw/raw247 /dev/sdk9
raw /dev/raw/raw281 /dev/sdk10
raw /dev/raw/raw315 /dev/sdk11
raw /dev/raw/raw349 /dev/sdk12
raw /dev/raw/raw383 /dev/sdk13
raw /dev/raw/raw10 /dev/sdl1
raw /dev/raw/raw44 /dev/sdl2
raw /dev/raw/raw78 /dev/sdl3
raw /dev/raw/raw112 /dev/sdl5
raw /dev/raw/raw146 /dev/sdl6
raw /dev/raw/raw180 /dev/sdl7
raw /dev/raw/raw214 /dev/sdl8
raw /dev/raw/raw248 /dev/sdl9
raw /dev/raw/raw282 /dev/sdl10
raw /dev/raw/raw316 /dev/sdl11
raw /dev/raw/raw350 /dev/sdl12
raw /dev/raw/raw384 /dev/sdl13

raw /dev/raw/raw11 /dev/sdn1
raw /dev/raw/raw45 /dev/sdn2
raw /dev/raw/raw79 /dev/sdn3
raw /dev/raw/raw113 /dev/sdn5
raw /dev/raw/raw147 /dev/sdn6
raw /dev/raw/raw181 /dev/sdn7
raw /dev/raw/raw215 /dev/sdn8
raw /dev/raw/raw249 /dev/sdn9
raw /dev/raw/raw283 /dev/sdn10
raw /dev/raw/raw317 /dev/sdn11
raw /dev/raw/raw351 /dev/sdn12
raw /dev/raw/raw385 /dev/sdn13
raw /dev/raw/raw12 /dev/sdo1
raw /dev/raw/raw46 /dev/sdo2
raw /dev/raw/raw80 /dev/sdo3
raw /dev/raw/raw114 /dev/sdo5
raw /dev/raw/raw148 /dev/sdo6
raw /dev/raw/raw182 /dev/sdo7
raw /dev/raw/raw216 /dev/sdo8
raw /dev/raw/raw250 /dev/sdo9
raw /dev/raw/raw284 /dev/sdo10
raw /dev/raw/raw318 /dev/sdo11
raw /dev/raw/raw352 /dev/sdo12
raw /dev/raw/raw386 /dev/sdo13
raw /dev/raw/raw13 /dev/sdp1
raw /dev/raw/raw47 /dev/sdp2
raw /dev/raw/raw81 /dev/sdp3
raw /dev/raw/raw115 /dev/sdp5
raw /dev/raw/raw149 /dev/sdp6
raw /dev/raw/raw183 /dev/sdp7
raw /dev/raw/raw217 /dev/sdp8
raw /dev/raw/raw251 /dev/sdp9
raw /dev/raw/raw285 /dev/sdp10
raw /dev/raw/raw319 /dev/sdp11
raw /dev/raw/raw353 /dev/sdp12
raw /dev/raw/raw387 /dev/sdp13
raw /dev/raw/raw14 /dev/sdq1
raw /dev/raw/raw48 /dev/sdq2
raw /dev/raw/raw82 /dev/sdq3
raw /dev/raw/raw116 /dev/sdq5
raw /dev/raw/raw150 /dev/sdq6
raw /dev/raw/raw184 /dev/sdq7
raw /dev/raw/raw218 /dev/sdq8
raw /dev/raw/raw252 /dev/sdq9
raw /dev/raw/raw286 /dev/sdq10
raw /dev/raw/raw320 /dev/sdq11
raw /dev/raw/raw354 /dev/sdq12
raw /dev/raw/raw388 /dev/sdq13
raw /dev/raw/raw15 /dev/sds1
raw /dev/raw/raw49 /dev/sds2
raw /dev/raw/raw83 /dev/sds3
raw /dev/raw/raw117 /dev/sds5
raw /dev/raw/raw151 /dev/sds6
raw /dev/raw/raw185 /dev/sds7
raw /dev/raw/raw219 /dev/sds8
raw /dev/raw/raw253 /dev/sds9
raw /dev/raw/raw287 /dev/sds10
raw /dev/raw/raw321 /dev/sds11
raw /dev/raw/raw355 /dev/sds12
raw /dev/raw/raw389 /dev/sds13
raw /dev/raw/raw16 /dev/sdt1
raw /dev/raw/raw50 /dev/sdt2
raw /dev/raw/raw84 /dev/sdt3
raw /dev/raw/raw118 /dev/sdt5
raw /dev/raw/raw152 /dev/sdt6
raw /dev/raw/raw186 /dev/sdt7
raw /dev/raw/raw220 /dev/sdt8
raw /dev/raw/raw254 /dev/sdt9
raw /dev/raw/raw288 /dev/sdt10
raw /dev/raw/raw322 /dev/sdt11
raw /dev/raw/raw356 /dev/sdt12
raw /dev/raw/raw390 /dev/sdt13
raw /dev/raw/raw17 /dev/sdu1
raw /dev/raw/raw51 /dev/sdu2
raw /dev/raw/raw85 /dev/sdu3
raw /dev/raw/raw119 /dev/sdu5
raw /dev/raw/raw153 /dev/sdu6
raw /dev/raw/raw187 /dev/sdu7
raw /dev/raw/raw221 /dev/sdu8
raw /dev/raw/raw255 /dev/sdu9
raw /dev/raw/raw289 /dev/sdu10
raw /dev/raw/raw323 /dev/sdu11
raw /dev/raw/raw357 /dev/sdu12
raw /dev/raw/raw391 /dev/sdu13
raw /dev/raw/raw18 /dev/sdv1
raw /dev/raw/raw52 /dev/sdv2

raw /dev/raw/raw86 /dev/sdv3
raw /dev/raw/raw120 /dev/sdv5
raw /dev/raw/raw154 /dev/sdv6
raw /dev/raw/raw188 /dev/sdv7
raw /dev/raw/raw222 /dev/sdv8
raw /dev/raw/raw256 /dev/sdv9
raw /dev/raw/raw290 /dev/sdv10
raw /dev/raw/raw324 /dev/sdv11
raw /dev/raw/raw358 /dev/sdv12
raw /dev/raw/raw392 /dev/sdv13
raw /dev/raw/raw19 /dev/sdx1
raw /dev/raw/raw53 /dev/sdx2
raw /dev/raw/raw87 /dev/sdx3
raw /dev/raw/raw121 /dev/sdx5
raw /dev/raw/raw155 /dev/sdx6
raw /dev/raw/raw189 /dev/sdx7
raw /dev/raw/raw223 /dev/sdx8
raw /dev/raw/raw257 /dev/sdx9
raw /dev/raw/raw291 /dev/sdx10
raw /dev/raw/raw325 /dev/sdx11
raw /dev/raw/raw359 /dev/sdx12
raw /dev/raw/raw393 /dev/sdx13
raw /dev/raw/raw20 /dev/sdy1
raw /dev/raw/raw54 /dev/sdy2
raw /dev/raw/raw88 /dev/sdy3
raw /dev/raw/raw122 /dev/sdy5
raw /dev/raw/raw156 /dev/sdy6
raw /dev/raw/raw190 /dev/sdy7
raw /dev/raw/raw224 /dev/sdy8
raw /dev/raw/raw258 /dev/sdy9
raw /dev/raw/raw292 /dev/sdy10
raw /dev/raw/raw326 /dev/sdy11
raw /dev/raw/raw360 /dev/sdy12
raw /dev/raw/raw394 /dev/sdy13
raw /dev/raw/raw21 /dev/sdz1
raw /dev/raw/raw55 /dev/sdz2
raw /dev/raw/raw89 /dev/sdz3
raw /dev/raw/raw123 /dev/sdz5
raw /dev/raw/raw157 /dev/sdz6
raw /dev/raw/raw191 /dev/sdz7
raw /dev/raw/raw225 /dev/sdz8
raw /dev/raw/raw259 /dev/sdz9
raw /dev/raw/raw293 /dev/sdz10
raw /dev/raw/raw327 /dev/sdz11
raw /dev/raw/raw361 /dev/sdz12
raw /dev/raw/raw395 /dev/sdz13
raw /dev/raw/raw22 /dev/sdaa1
raw /dev/raw/raw56 /dev/sdaa2
raw /dev/raw/raw90 /dev/sdaa3
raw /dev/raw/raw124 /dev/sdaa5
raw /dev/raw/raw158 /dev/sdaa6
raw /dev/raw/raw192 /dev/sdaa7
raw /dev/raw/raw226 /dev/sdaa8
raw /dev/raw/raw260 /dev/sdaa9
raw /dev/raw/raw294 /dev/sdaa10
raw /dev/raw/raw328 /dev/sdaa11
raw /dev/raw/raw362 /dev/sdaa12
raw /dev/raw/raw396 /dev/sdaa13
raw /dev/raw/raw23 /dev/sdac1
raw /dev/raw/raw57 /dev/sdac2
raw /dev/raw/raw91 /dev/sdac3
raw /dev/raw/raw125 /dev/sdac5
raw /dev/raw/raw159 /dev/sdac6
raw /dev/raw/raw193 /dev/sdac7
raw /dev/raw/raw227 /dev/sdac8
raw /dev/raw/raw261 /dev/sdac9
raw /dev/raw/raw295 /dev/sdac10
raw /dev/raw/raw329 /dev/sdac11
raw /dev/raw/raw363 /dev/sdac12
raw /dev/raw/raw397 /dev/sdac13
raw /dev/raw/raw24 /dev/sdad1
raw /dev/raw/raw58 /dev/sdad2
raw /dev/raw/raw92 /dev/sdad3
raw /dev/raw/raw126 /dev/sdad5
raw /dev/raw/raw160 /dev/sdad6
raw /dev/raw/raw194 /dev/sdad7
raw /dev/raw/raw228 /dev/sdad8
raw /dev/raw/raw262 /dev/sdad9
raw /dev/raw/raw296 /dev/sdad10
raw /dev/raw/raw330 /dev/sdad11
raw /dev/raw/raw364 /dev/sdad12
raw /dev/raw/raw398 /dev/sdad13
raw /dev/raw/raw25 /dev/sdae1
raw /dev/raw/raw59 /dev/sdae2
raw /dev/raw/raw93 /dev/sdae3
raw /dev/raw/raw127 /dev/sdae5

```
raw /dev/raw/raw161 /dev/sdae6
raw /dev/raw/raw195 /dev/sdae7
raw /dev/raw/raw229 /dev/sdae8
raw /dev/raw/raw263 /dev/sdae9
raw /dev/raw/raw297 /dev/sdae10
raw /dev/raw/raw331 /dev/sdae11
raw /dev/raw/raw365 /dev/sdae12
raw /dev/raw/raw399 /dev/sdae13
raw /dev/raw/raw26 /dev/sdaf1
raw /dev/raw/raw60 /dev/sdaf2
raw /dev/raw/raw94 /dev/sdaf3
raw /dev/raw/raw128 /dev/sdaf5
raw /dev/raw/raw162 /dev/sdaf6
raw /dev/raw/raw196 /dev/sdaf7
raw /dev/raw/raw230 /dev/sdaf8
raw /dev/raw/raw264 /dev/sdaf9
raw /dev/raw/raw298 /dev/sdaf10
raw /dev/raw/raw332 /dev/sdaf11
raw /dev/raw/raw366 /dev/sdaf12
raw /dev/raw/raw400 /dev/sdaf13
raw /dev/raw/raw27 /dev/sdah1
raw /dev/raw/raw61 /dev/sdah2
raw /dev/raw/raw95 /dev/sdah3
raw /dev/raw/raw129 /dev/sdah5
raw /dev/raw/raw163 /dev/sdah6
raw /dev/raw/raw197 /dev/sdah7
raw /dev/raw/raw231 /dev/sdah8
raw /dev/raw/raw265 /dev/sdah9
raw /dev/raw/raw299 /dev/sdah10
raw /dev/raw/raw333 /dev/sdah11
raw /dev/raw/raw367 /dev/sdah12
raw /dev/raw/raw401 /dev/sdah13
raw /dev/raw/raw411 /dev/sdat1
raw /dev/raw/raw412 /dev/sdat2
raw /dev/raw/raw413 /dev/sdat3
sleep 5
chown oracle:dba /dev/raw/raw*
chmod 660 /dev/raw/raw*
```

link.sh

```
if [ ! -d /home/oracle/tpcc_disks ]
then
  mkdir /home/oracle/tpcc_disks
fi

rm -f /home/oracle/tpcc_disks/*
ln -s /dev/raw/raw28 /home/oracle/tpcc_disks/stok_0_25
ln -s /dev/raw/raw62 /home/oracle/tpcc_disks/stok_0_59
ln -s /dev/raw/raw96 /home/oracle/tpcc_disks/stok_0_93
ln -s /dev/raw/raw130 /home/oracle/tpcc_disks/cust_0_25
ln -s /dev/raw/raw164 /home/oracle/tpcc_disks/cust_0_59
ln -s /dev/raw/raw198 /home/oracle/tpcc_disks/cust_0_93
ln -s /dev/raw/raw232 /home/oracle/tpcc_disks/hist_0_25
ln -s /dev/raw/raw266 /home/oracle/tpcc_disks/ordr_0_25
ln -s /dev/raw/raw300 /home/oracle/tpcc_disks/icust2_0_25
ln -s /dev/raw/raw334 /home/oracle/tpcc_disks/iordr2_0_25
ln -s /dev/raw/raw368 /home/oracle/tpcc_disks/temp_0_25
ln -s /dev/raw/raw402 /home/oracle/tpcc_disks/etc_0_25
ln -s /dev/raw/raw29 /home/oracle/tpcc_disks/stok_0_26
ln -s /dev/raw/raw63 /home/oracle/tpcc_disks/stok_0_60
ln -s /dev/raw/raw97 /home/oracle/tpcc_disks/stok_0_94
ln -s /dev/raw/raw131 /home/oracle/tpcc_disks/cust_0_26
ln -s /dev/raw/raw165 /home/oracle/tpcc_disks/cust_0_60
ln -s /dev/raw/raw199 /home/oracle/tpcc_disks/cust_0_94
ln -s /dev/raw/raw233 /home/oracle/tpcc_disks/hist_0_26
ln -s /dev/raw/raw267 /home/oracle/tpcc_disks/ordr_0_26
ln -s /dev/raw/raw301 /home/oracle/tpcc_disks/icust2_0_26
ln -s /dev/raw/raw335 /home/oracle/tpcc_disks/iordr2_0_26
ln -s /dev/raw/raw369 /home/oracle/tpcc_disks/temp_0_26
ln -s /dev/raw/raw403 /home/oracle/tpcc_disks/etc_0_26
ln -s /dev/raw/raw30 /home/oracle/tpcc_disks/stok_0_27
ln -s /dev/raw/raw64 /home/oracle/tpcc_disks/stok_0_61
ln -s /dev/raw/raw98 /home/oracle/tpcc_disks/stok_0_95
ln -s /dev/raw/raw132 /home/oracle/tpcc_disks/cust_0_27
ln -s /dev/raw/raw166 /home/oracle/tpcc_disks/cust_0_61
ln -s /dev/raw/raw200 /home/oracle/tpcc_disks/cust_0_95
ln -s /dev/raw/raw234 /home/oracle/tpcc_disks/hist_0_27
ln -s /dev/raw/raw268 /home/oracle/tpcc_disks/ordr_0_27
ln -s /dev/raw/raw302 /home/oracle/tpcc_disks/icust2_0_27
ln -s /dev/raw/raw336 /home/oracle/tpcc_disks/iordr2_0_27
ln -s /dev/raw/raw370 /home/oracle/tpcc_disks/temp_0_27
ln -s /dev/raw/raw404 /home/oracle/tpcc_disks/etc_0_27
ln -s /dev/raw/raw31 /home/oracle/tpcc_disks/stok_0_28
ln -s /dev/raw/raw65 /home/oracle/tpcc_disks/stok_0_62
ln -s /dev/raw/raw99 /home/oracle/tpcc_disks/stok_0_96
```

```
ln -s /dev/raw/raw133 /home/oracle/tpcc_disks/cust_0_28
ln -s /dev/raw/raw167 /home/oracle/tpcc_disks/cust_0_62
ln -s /dev/raw/raw201 /home/oracle/tpcc_disks/cust_0_96
ln -s /dev/raw/raw235 /home/oracle/tpcc_disks/hist_0_28
ln -s /dev/raw/raw269 /home/oracle/tpcc_disks/ordr_0_28
ln -s /dev/raw/raw303 /home/oracle/tpcc_disks/icust2_0_28
ln -s /dev/raw/raw337 /home/oracle/tpcc_disks/iordr2_0_28
ln -s /dev/raw/raw371 /home/oracle/tpcc_disks/temp_0_28
ln -s /dev/raw/raw405 /home/oracle/tpcc_disks/etc_0_28
ln -s /dev/raw/raw32 /home/oracle/tpcc_disks/stok_0_29
ln -s /dev/raw/raw66 /home/oracle/tpcc_disks/stok_0_63
ln -s /dev/raw/raw100 /home/oracle/tpcc_disks/stok_0_97
ln -s /dev/raw/raw134 /home/oracle/tpcc_disks/cust_0_29
ln -s /dev/raw/raw168 /home/oracle/tpcc_disks/cust_0_63
ln -s /dev/raw/raw202 /home/oracle/tpcc_disks/cust_0_97
ln -s /dev/raw/raw236 /home/oracle/tpcc_disks/hist_0_29
ln -s /dev/raw/raw270 /home/oracle/tpcc_disks/ordr_0_29
ln -s /dev/raw/raw304 /home/oracle/tpcc_disks/icust2_0_29
ln -s /dev/raw/raw338 /home/oracle/tpcc_disks/iordr2_0_29
ln -s /dev/raw/raw372 /home/oracle/tpcc_disks/temp_0_29
ln -s /dev/raw/raw406 /home/oracle/tpcc_disks/etc_0_29
ln -s /dev/raw/raw33 /home/oracle/tpcc_disks/stok_0_30
ln -s /dev/raw/raw67 /home/oracle/tpcc_disks/stok_0_64
ln -s /dev/raw/raw101 /home/oracle/tpcc_disks/stok_0_98
ln -s /dev/raw/raw135 /home/oracle/tpcc_disks/cust_0_30
ln -s /dev/raw/raw169 /home/oracle/tpcc_disks/cust_0_64
ln -s /dev/raw/raw203 /home/oracle/tpcc_disks/cust_0_98
ln -s /dev/raw/raw237 /home/oracle/tpcc_disks/hist_0_30
ln -s /dev/raw/raw271 /home/oracle/tpcc_disks/ordr_0_30
ln -s /dev/raw/raw305 /home/oracle/tpcc_disks/icust2_0_30
ln -s /dev/raw/raw339 /home/oracle/tpcc_disks/iordr2_0_30
ln -s /dev/raw/raw373 /home/oracle/tpcc_disks/temp_0_30
ln -s /dev/raw/raw407 /home/oracle/tpcc_disks/etc_0_30
ln -s /dev/raw/raw34 /home/oracle/tpcc_disks/stok_0_31
ln -s /dev/raw/raw68 /home/oracle/tpcc_disks/stok_0_65
ln -s /dev/raw/raw102 /home/oracle/tpcc_disks/stok_0_99
ln -s /dev/raw/raw136 /home/oracle/tpcc_disks/cust_0_31
ln -s /dev/raw/raw170 /home/oracle/tpcc_disks/cust_0_65
ln -s /dev/raw/raw204 /home/oracle/tpcc_disks/cust_0_99
ln -s /dev/raw/raw238 /home/oracle/tpcc_disks/hist_0_31
ln -s /dev/raw/raw272 /home/oracle/tpcc_disks/ordr_0_31
ln -s /dev/raw/raw306 /home/oracle/tpcc_disks/icust2_0_31
ln -s /dev/raw/raw340 /home/oracle/tpcc_disks/iordr2_0_31
ln -s /dev/raw/raw374 /home/oracle/tpcc_disks/temp_0_31
ln -s /dev/raw/raw408 /home/oracle/tpcc_disks/etc_0_31
ln -s /dev/raw/raw35 /home/oracle/tpcc_disks/stok_0_32
ln -s /dev/raw/raw69 /home/oracle/tpcc_disks/stok_0_66
ln -s /dev/raw/raw103 /home/oracle/tpcc_disks/stok_0_100
ln -s /dev/raw/raw137 /home/oracle/tpcc_disks/cust_0_32
ln -s /dev/raw/raw171 /home/oracle/tpcc_disks/cust_0_66
ln -s /dev/raw/raw205 /home/oracle/tpcc_disks/cust_0_100
ln -s /dev/raw/raw239 /home/oracle/tpcc_disks/hist_0_32
ln -s /dev/raw/raw273 /home/oracle/tpcc_disks/ordr_0_32
ln -s /dev/raw/raw307 /home/oracle/tpcc_disks/icust2_0_32
ln -s /dev/raw/raw341 /home/oracle/tpcc_disks/iordr2_0_32
ln -s /dev/raw/raw375 /home/oracle/tpcc_disks/temp_0_32
ln -s /dev/raw/raw409 /home/oracle/tpcc_disks/etc_0_32
ln -s /dev/raw/raw36 /home/oracle/tpcc_disks/stok_0_33
ln -s /dev/raw/raw70 /home/oracle/tpcc_disks/stok_0_67
ln -s /dev/raw/raw104 /home/oracle/tpcc_disks/stok_0_101
ln -s /dev/raw/raw138 /home/oracle/tpcc_disks/cust_0_33
ln -s /dev/raw/raw172 /home/oracle/tpcc_disks/cust_0_67
ln -s /dev/raw/raw206 /home/oracle/tpcc_disks/cust_0_101
ln -s /dev/raw/raw240 /home/oracle/tpcc_disks/hist_0_33
ln -s /dev/raw/raw274 /home/oracle/tpcc_disks/ordr_0_33
ln -s /dev/raw/raw308 /home/oracle/tpcc_disks/icust2_0_33
ln -s /dev/raw/raw342 /home/oracle/tpcc_disks/iordr2_0_33
ln -s /dev/raw/raw376 /home/oracle/tpcc_disks/temp_0_33
ln -s /dev/raw/raw410 /home/oracle/tpcc_disks/etc_0_33
ln -s /dev/raw/raw1 /home/oracle/tpcc_disks/log_1_1
ln -s /dev/raw/raw2 /home/oracle/tpcc_disks/log_1_2
ln -s /dev/raw/raw3 /home/oracle/tpcc_disks/stok_0_0
ln -s /dev/raw/raw37 /home/oracle/tpcc_disks/stok_0_34
ln -s /dev/raw/raw71 /home/oracle/tpcc_disks/stok_0_68
ln -s /dev/raw/raw105 /home/oracle/tpcc_disks/cust_0_0
ln -s /dev/raw/raw139 /home/oracle/tpcc_disks/cust_0_34
ln -s /dev/raw/raw173 /home/oracle/tpcc_disks/cust_0_68
ln -s /dev/raw/raw207 /home/oracle/tpcc_disks/hist_0_0
ln -s /dev/raw/raw241 /home/oracle/tpcc_disks/ordr_0_0
ln -s /dev/raw/raw275 /home/oracle/tpcc_disks/icust2_0_0
ln -s /dev/raw/raw309 /home/oracle/tpcc_disks/iordr2_0_0
ln -s /dev/raw/raw343 /home/oracle/tpcc_disks/temp_0_0
ln -s /dev/raw/raw377 /home/oracle/tpcc_disks/control_001
ln -s /dev/raw/raw4 /home/oracle/tpcc_disks/stok_0_1
ln -s /dev/raw/raw38 /home/oracle/tpcc_disks/stok_0_35
ln -s /dev/raw/raw72 /home/oracle/tpcc_disks/stok_0_69
```

```
ln -s /dev/raw/raw106 /home/oracle/tpcc_disks/cust_0_1
ln -s /dev/raw/raw140 /home/oracle/tpcc_disks/cust_0_35
ln -s /dev/raw/raw174 /home/oracle/tpcc_disks/cust_0_69
ln -s /dev/raw/raw208 /home/oracle/tpcc_disks/hist_0_1
ln -s /dev/raw/raw242 /home/oracle/tpcc_disks/ordr_0_1
ln -s /dev/raw/raw276 /home/oracle/tpcc_disks/icust2_0_1
ln -s /dev/raw/raw310 /home/oracle/tpcc_disks/iordr2_0_1
ln -s /dev/raw/raw344 /home/oracle/tpcc_disks/temp_0_1
ln -s /dev/raw/raw378 /home/oracle/tpcc_disks/etc_0_1
ln -s /dev/raw/raw5 /home/oracle/tpcc_disks/stok_0_2
ln -s /dev/raw/raw39 /home/oracle/tpcc_disks/stok_0_36
ln -s /dev/raw/raw73 /home/oracle/tpcc_disks/stok_0_70
ln -s /dev/raw/raw107 /home/oracle/tpcc_disks/cust_0_2
ln -s /dev/raw/raw141 /home/oracle/tpcc_disks/cust_0_36
ln -s /dev/raw/raw175 /home/oracle/tpcc_disks/cust_0_70
ln -s /dev/raw/raw209 /home/oracle/tpcc_disks/hist_0_2
ln -s /dev/raw/raw243 /home/oracle/tpcc_disks/ordr_0_2
ln -s /dev/raw/raw277 /home/oracle/tpcc_disks/icust2_0_2
ln -s /dev/raw/raw311 /home/oracle/tpcc_disks/iordr2_0_2
ln -s /dev/raw/raw345 /home/oracle/tpcc_disks/temp_0_2
ln -s /dev/raw/raw379 /home/oracle/tpcc_disks/ware_0_0
ln -s /dev/raw/raw6 /home/oracle/tpcc_disks/stok_0_3
ln -s /dev/raw/raw40 /home/oracle/tpcc_disks/stok_0_37
ln -s /dev/raw/raw74 /home/oracle/tpcc_disks/stok_0_71
ln -s /dev/raw/raw108 /home/oracle/tpcc_disks/cust_0_3
ln -s /dev/raw/raw142 /home/oracle/tpcc_disks/cust_0_37
ln -s /dev/raw/raw176 /home/oracle/tpcc_disks/cust_0_71
ln -s /dev/raw/raw210 /home/oracle/tpcc_disks/hist_0_3
ln -s /dev/raw/raw244 /home/oracle/tpcc_disks/ordr_0_3
ln -s /dev/raw/raw278 /home/oracle/tpcc_disks/icust2_0_3
ln -s /dev/raw/raw312 /home/oracle/tpcc_disks/iordr2_0_3
ln -s /dev/raw/raw346 /home/oracle/tpcc_disks/temp_0_3
ln -s /dev/raw/raw380 /home/oracle/tpcc_disks/etc_0_3
ln -s /dev/raw/raw7 /home/oracle/tpcc_disks/stok_0_4
ln -s /dev/raw/raw41 /home/oracle/tpcc_disks/stok_0_38
ln -s /dev/raw/raw75 /home/oracle/tpcc_disks/stok_0_72
ln -s /dev/raw/raw109 /home/oracle/tpcc_disks/cust_0_4
ln -s /dev/raw/raw143 /home/oracle/tpcc_disks/cust_0_38
ln -s /dev/raw/raw177 /home/oracle/tpcc_disks/cust_0_72
ln -s /dev/raw/raw211 /home/oracle/tpcc_disks/hist_0_4
ln -s /dev/raw/raw245 /home/oracle/tpcc_disks/ordr_0_4
ln -s /dev/raw/raw279 /home/oracle/tpcc_disks/icust2_0_4
ln -s /dev/raw/raw313 /home/oracle/tpcc_disks/iordr2_0_4
ln -s /dev/raw/raw347 /home/oracle/tpcc_disks/temp_0_4
ln -s /dev/raw/raw381 /home/oracle/tpcc_disks/control_002
ln -s /dev/raw/raw8 /home/oracle/tpcc_disks/stok_0_5
ln -s /dev/raw/raw42 /home/oracle/tpcc_disks/stok_0_39
ln -s /dev/raw/raw76 /home/oracle/tpcc_disks/stok_0_73
ln -s /dev/raw/raw110 /home/oracle/tpcc_disks/cust_0_5
ln -s /dev/raw/raw144 /home/oracle/tpcc_disks/cust_0_39
ln -s /dev/raw/raw178 /home/oracle/tpcc_disks/cust_0_73
ln -s /dev/raw/raw212 /home/oracle/tpcc_disks/hist_0_5
ln -s /dev/raw/raw246 /home/oracle/tpcc_disks/ordr_0_5
ln -s /dev/raw/raw280 /home/oracle/tpcc_disks/icust2_0_5
ln -s /dev/raw/raw314 /home/oracle/tpcc_disks/iordr2_0_5
ln -s /dev/raw/raw348 /home/oracle/tpcc_disks/temp_0_5
ln -s /dev/raw/raw382 /home/oracle/tpcc_disks/etc_0_5
ln -s /dev/raw/raw9 /home/oracle/tpcc_disks/stok_0_6
ln -s /dev/raw/raw43 /home/oracle/tpcc_disks/stok_0_40
ln -s /dev/raw/raw77 /home/oracle/tpcc_disks/stok_0_74
ln -s /dev/raw/raw111 /home/oracle/tpcc_disks/cust_0_6
ln -s /dev/raw/raw145 /home/oracle/tpcc_disks/cust_0_40
ln -s /dev/raw/raw179 /home/oracle/tpcc_disks/cust_0_74
ln -s /dev/raw/raw213 /home/oracle/tpcc_disks/hist_0_6
ln -s /dev/raw/raw247 /home/oracle/tpcc_disks/ordr_0_6
ln -s /dev/raw/raw281 /home/oracle/tpcc_disks/icust2_0_6
ln -s /dev/raw/raw315 /home/oracle/tpcc_disks/iordr2_0_6
ln -s /dev/raw/raw349 /home/oracle/tpcc_disks/temp_0_6
ln -s /dev/raw/raw383 /home/oracle/tpcc_disks/dist_0_0
ln -s /dev/raw/raw10 /home/oracle/tpcc_disks/stok_0_7
ln -s /dev/raw/raw44 /home/oracle/tpcc_disks/stok_0_41
ln -s /dev/raw/raw78 /home/oracle/tpcc_disks/stok_0_75
ln -s /dev/raw/raw112 /home/oracle/tpcc_disks/cust_0_7
ln -s /dev/raw/raw146 /home/oracle/tpcc_disks/cust_0_41
ln -s /dev/raw/raw180 /home/oracle/tpcc_disks/cust_0_75
ln -s /dev/raw/raw214 /home/oracle/tpcc_disks/hist_0_7
ln -s /dev/raw/raw248 /home/oracle/tpcc_disks/ordr_0_7
ln -s /dev/raw/raw282 /home/oracle/tpcc_disks/icust2_0_7
ln -s /dev/raw/raw316 /home/oracle/tpcc_disks/iordr2_0_7
ln -s /dev/raw/raw350 /home/oracle/tpcc_disks/temp_0_7
ln -s /dev/raw/raw384 /home/oracle/tpcc_disks/etc_0_7
ln -s /dev/raw/raw11 /home/oracle/tpcc_disks/stok_0_8
ln -s /dev/raw/raw45 /home/oracle/tpcc_disks/stok_0_42
ln -s /dev/raw/raw79 /home/oracle/tpcc_disks/stok_0_76
ln -s /dev/raw/raw113 /home/oracle/tpcc_disks/cust_0_8
ln -s /dev/raw/raw147 /home/oracle/tpcc_disks/cust_0_42
```

```
ln -s /dev/raw/raw181 /home/oracle/tpcc_disks/cust_0_76
ln -s /dev/raw/raw215 /home/oracle/tpcc_disks/hist_0_8
ln -s /dev/raw/raw249 /home/oracle/tpcc_disks/ordr_0_8
ln -s /dev/raw/raw283 /home/oracle/tpcc_disks/icust2_0_8
ln -s /dev/raw/raw317 /home/oracle/tpcc_disks/iordr2_0_8
ln -s /dev/raw/raw351 /home/oracle/tpcc_disks/temp_0_8
ln -s /dev/raw/raw385 /home/oracle/tpcc_disks/system_1
ln -s /dev/raw/raw12 /home/oracle/tpcc_disks/stok_0_9
ln -s /dev/raw/raw46 /home/oracle/tpcc_disks/stok_0_43
ln -s /dev/raw/raw80 /home/oracle/tpcc_disks/stok_0_77
ln -s /dev/raw/raw114 /home/oracle/tpcc_disks/cust_0_9
ln -s /dev/raw/raw148 /home/oracle/tpcc_disks/cust_0_43
ln -s /dev/raw/raw182 /home/oracle/tpcc_disks/cust_0_77
ln -s /dev/raw/raw216 /home/oracle/tpcc_disks/hist_0_9
ln -s /dev/raw/raw250 /home/oracle/tpcc_disks/ordr_0_9
ln -s /dev/raw/raw284 /home/oracle/tpcc_disks/icust2_0_9
ln -s /dev/raw/raw318 /home/oracle/tpcc_disks/iordr2_0_9
ln -s /dev/raw/raw352 /home/oracle/tpcc_disks/temp_0_9
ln -s /dev/raw/raw386 /home/oracle/tpcc_disks/etc_0_9
ln -s /dev/raw/raw13 /home/oracle/tpcc_disks/stok_0_10
ln -s /dev/raw/raw47 /home/oracle/tpcc_disks/stok_0_44
ln -s /dev/raw/raw81 /home/oracle/tpcc_disks/stok_0_78
ln -s /dev/raw/raw115 /home/oracle/tpcc_disks/cust_0_10
ln -s /dev/raw/raw149 /home/oracle/tpcc_disks/cust_0_44
ln -s /dev/raw/raw183 /home/oracle/tpcc_disks/cust_0_78
ln -s /dev/raw/raw217 /home/oracle/tpcc_disks/hist_0_10
ln -s /dev/raw/raw251 /home/oracle/tpcc_disks/ordr_0_10
ln -s /dev/raw/raw285 /home/oracle/tpcc_disks/icust2_0_10
ln -s /dev/raw/raw319 /home/oracle/tpcc_disks/iordr2_0_10
ln -s /dev/raw/raw353 /home/oracle/tpcc_disks/temp_0_10
ln -s /dev/raw/raw387 /home/oracle/tpcc_disks/item_0_0
ln -s /dev/raw/raw14 /home/oracle/tpcc_disks/stok_0_11
ln -s /dev/raw/raw48 /home/oracle/tpcc_disks/stok_0_45
ln -s /dev/raw/raw82 /home/oracle/tpcc_disks/stok_0_79
ln -s /dev/raw/raw116 /home/oracle/tpcc_disks/cust_0_11
ln -s /dev/raw/raw150 /home/oracle/tpcc_disks/cust_0_45
ln -s /dev/raw/raw184 /home/oracle/tpcc_disks/cust_0_79
ln -s /dev/raw/raw218 /home/oracle/tpcc_disks/hist_0_11
ln -s /dev/raw/raw252 /home/oracle/tpcc_disks/ordr_0_11
ln -s /dev/raw/raw286 /home/oracle/tpcc_disks/icust2_0_11
ln -s /dev/raw/raw320 /home/oracle/tpcc_disks/iordr2_0_11
ln -s /dev/raw/raw354 /home/oracle/tpcc_disks/temp_0_11
ln -s /dev/raw/raw388 /home/oracle/tpcc_disks/etc_0_11
ln -s /dev/raw/raw15 /home/oracle/tpcc_disks/stok_0_12
ln -s /dev/raw/raw49 /home/oracle/tpcc_disks/stok_0_46
ln -s /dev/raw/raw83 /home/oracle/tpcc_disks/stok_0_80
ln -s /dev/raw/raw117 /home/oracle/tpcc_disks/cust_0_12
ln -s /dev/raw/raw151 /home/oracle/tpcc_disks/cust_0_46
ln -s /dev/raw/raw185 /home/oracle/tpcc_disks/cust_0_80
ln -s /dev/raw/raw219 /home/oracle/tpcc_disks/hist_0_12
ln -s /dev/raw/raw253 /home/oracle/tpcc_disks/ordr_0_12
ln -s /dev/raw/raw287 /home/oracle/tpcc_disks/icust2_0_12
ln -s /dev/raw/raw321 /home/oracle/tpcc_disks/iordr2_0_12
ln -s /dev/raw/raw355 /home/oracle/tpcc_disks/temp_0_12
ln -s /dev/raw/raw389 /home/oracle/tpcc_disks/tpccaux
ln -s /dev/raw/raw16 /home/oracle/tpcc_disks/stok_0_13
ln -s /dev/raw/raw50 /home/oracle/tpcc_disks/stok_0_47
ln -s /dev/raw/raw84 /home/oracle/tpcc_disks/stok_0_81
ln -s /dev/raw/raw118 /home/oracle/tpcc_disks/cust_0_13
ln -s /dev/raw/raw152 /home/oracle/tpcc_disks/cust_0_47
ln -s /dev/raw/raw186 /home/oracle/tpcc_disks/cust_0_81
ln -s /dev/raw/raw220 /home/oracle/tpcc_disks/hist_0_13
ln -s /dev/raw/raw254 /home/oracle/tpcc_disks/ordr_0_13
ln -s /dev/raw/raw288 /home/oracle/tpcc_disks/icust2_0_13
ln -s /dev/raw/raw322 /home/oracle/tpcc_disks/iordr2_0_13
ln -s /dev/raw/raw356 /home/oracle/tpcc_disks/temp_0_13
ln -s /dev/raw/raw390 /home/oracle/tpcc_disks/etc_0_13
ln -s /dev/raw/raw17 /home/oracle/tpcc_disks/stok_0_14
ln -s /dev/raw/raw51 /home/oracle/tpcc_disks/stok_0_48
ln -s /dev/raw/raw85 /home/oracle/tpcc_disks/stok_0_82
ln -s /dev/raw/raw119 /home/oracle/tpcc_disks/cust_0_14
ln -s /dev/raw/raw153 /home/oracle/tpcc_disks/cust_0_48
ln -s /dev/raw/raw187 /home/oracle/tpcc_disks/cust_0_82
ln -s /dev/raw/raw221 /home/oracle/tpcc_disks/hist_0_14
ln -s /dev/raw/raw255 /home/oracle/tpcc_disks/ordr_0_14
ln -s /dev/raw/raw289 /home/oracle/tpcc_disks/icust2_0_14
ln -s /dev/raw/raw323 /home/oracle/tpcc_disks/iordr2_0_14
ln -s /dev/raw/raw357 /home/oracle/tpcc_disks/temp_0_14
ln -s /dev/raw/raw391 /home/oracle/tpcc_disks/iware_0_0
ln -s /dev/raw/raw18 /home/oracle/tpcc_disks/stok_0_15
ln -s /dev/raw/raw52 /home/oracle/tpcc_disks/stok_0_49
ln -s /dev/raw/raw86 /home/oracle/tpcc_disks/stok_0_83
ln -s /dev/raw/raw120 /home/oracle/tpcc_disks/cust_0_15
ln -s /dev/raw/raw154 /home/oracle/tpcc_disks/cust_0_49
ln -s /dev/raw/raw188 /home/oracle/tpcc_disks/cust_0_83
ln -s /dev/raw/raw222 /home/oracle/tpcc_disks/hist_0_15
```

```
ln -s /dev/raw/raw256 /home/oracle/tpcc_disks/ordr_0_15
ln -s /dev/raw/raw290 /home/oracle/tpcc_disks/icust2_0_15
ln -s /dev/raw/raw324 /home/oracle/tpcc_disks/iordr2_0_15
ln -s /dev/raw/raw358 /home/oracle/tpcc_disks/temp_0_15
ln -s /dev/raw/raw392 /home/oracle/tpcc_disks/etc_0_15
ln -s /dev/raw/raw19 /home/oracle/tpcc_disks/stok_0_16
ln -s /dev/raw/raw53 /home/oracle/tpcc_disks/stok_0_50
ln -s /dev/raw/raw87 /home/oracle/tpcc_disks/stok_0_84
ln -s /dev/raw/raw121 /home/oracle/tpcc_disks/cust_0_16
ln -s /dev/raw/raw155 /home/oracle/tpcc_disks/cust_0_50
ln -s /dev/raw/raw189 /home/oracle/tpcc_disks/cust_0_84
ln -s /dev/raw/raw223 /home/oracle/tpcc_disks/hist_0_16
ln -s /dev/raw/raw257 /home/oracle/tpcc_disks/ordr_0_16
ln -s /dev/raw/raw291 /home/oracle/tpcc_disks/icust2_0_16
ln -s /dev/raw/raw325 /home/oracle/tpcc_disks/iordr2_0_16
ln -s /dev/raw/raw359 /home/oracle/tpcc_disks/temp_0_16
ln -s /dev/raw/raw393 /home/oracle/tpcc_disks/roll1
ln -s /dev/raw/raw20 /home/oracle/tpcc_disks/stok_0_17
ln -s /dev/raw/raw54 /home/oracle/tpcc_disks/stok_0_51
ln -s /dev/raw/raw88 /home/oracle/tpcc_disks/stok_0_85
ln -s /dev/raw/raw122 /home/oracle/tpcc_disks/cust_0_17
ln -s /dev/raw/raw156 /home/oracle/tpcc_disks/cust_0_51
ln -s /dev/raw/raw190 /home/oracle/tpcc_disks/cust_0_85
ln -s /dev/raw/raw224 /home/oracle/tpcc_disks/hist_0_17
ln -s /dev/raw/raw258 /home/oracle/tpcc_disks/ordr_0_17
ln -s /dev/raw/raw292 /home/oracle/tpcc_disks/icust2_0_17
ln -s /dev/raw/raw326 /home/oracle/tpcc_disks/iordr2_0_17
ln -s /dev/raw/raw360 /home/oracle/tpcc_disks/temp_0_17
ln -s /dev/raw/raw394 /home/oracle/tpcc_disks/etc_0_17
ln -s /dev/raw/raw21 /home/oracle/tpcc_disks/stok_0_18
ln -s /dev/raw/raw55 /home/oracle/tpcc_disks/stok_0_52
ln -s /dev/raw/raw89 /home/oracle/tpcc_disks/stok_0_86
ln -s /dev/raw/raw123 /home/oracle/tpcc_disks/cust_0_18
ln -s /dev/raw/raw157 /home/oracle/tpcc_disks/cust_0_52
ln -s /dev/raw/raw191 /home/oracle/tpcc_disks/cust_0_86
ln -s /dev/raw/raw225 /home/oracle/tpcc_disks/hist_0_18
ln -s /dev/raw/raw259 /home/oracle/tpcc_disks/ordr_0_18
ln -s /dev/raw/raw293 /home/oracle/tpcc_disks/icust2_0_18
ln -s /dev/raw/raw327 /home/oracle/tpcc_disks/iordr2_0_18
ln -s /dev/raw/raw361 /home/oracle/tpcc_disks/temp_0_18
ln -s /dev/raw/raw395 /home/oracle/tpcc_disks/idist_0_0
ln -s /dev/raw/raw22 /home/oracle/tpcc_disks/stok_0_19
ln -s /dev/raw/raw56 /home/oracle/tpcc_disks/stok_0_53
ln -s /dev/raw/raw90 /home/oracle/tpcc_disks/stok_0_87
ln -s /dev/raw/raw124 /home/oracle/tpcc_disks/cust_0_19
ln -s /dev/raw/raw158 /home/oracle/tpcc_disks/cust_0_53
ln -s /dev/raw/raw192 /home/oracle/tpcc_disks/cust_0_87
ln -s /dev/raw/raw226 /home/oracle/tpcc_disks/hist_0_19
ln -s /dev/raw/raw260 /home/oracle/tpcc_disks/ordr_0_19
ln -s /dev/raw/raw294 /home/oracle/tpcc_disks/icust2_0_19
ln -s /dev/raw/raw328 /home/oracle/tpcc_disks/iordr2_0_19
ln -s /dev/raw/raw362 /home/oracle/tpcc_disks/temp_0_19
ln -s /dev/raw/raw396 /home/oracle/tpcc_disks/etc_0_19
ln -s /dev/raw/raw23 /home/oracle/tpcc_disks/stok_0_20
ln -s /dev/raw/raw57 /home/oracle/tpcc_disks/stok_0_54
ln -s /dev/raw/raw91 /home/oracle/tpcc_disks/stok_0_88
ln -s /dev/raw/raw125 /home/oracle/tpcc_disks/cust_0_20
ln -s /dev/raw/raw159 /home/oracle/tpcc_disks/cust_0_54
ln -s /dev/raw/raw193 /home/oracle/tpcc_disks/cust_0_88
ln -s /dev/raw/raw227 /home/oracle/tpcc_disks/hist_0_20
ln -s /dev/raw/raw261 /home/oracle/tpcc_disks/ordr_0_20
ln -s /dev/raw/raw295 /home/oracle/tpcc_disks/icust2_0_20
ln -s /dev/raw/raw329 /home/oracle/tpcc_disks/iordr2_0_20
ln -s /dev/raw/raw363 /home/oracle/tpcc_disks/temp_0_20
ln -s /dev/raw/raw397 /home/oracle/tpcc_disks/sp_0
ln -s /dev/raw/raw24 /home/oracle/tpcc_disks/stok_0_21
ln -s /dev/raw/raw58 /home/oracle/tpcc_disks/stok_0_55
ln -s /dev/raw/raw92 /home/oracle/tpcc_disks/stok_0_89
ln -s /dev/raw/raw126 /home/oracle/tpcc_disks/cust_0_21
ln -s /dev/raw/raw160 /home/oracle/tpcc_disks/cust_0_55
ln -s /dev/raw/raw194 /home/oracle/tpcc_disks/cust_0_89
ln -s /dev/raw/raw228 /home/oracle/tpcc_disks/hist_0_21
ln -s /dev/raw/raw262 /home/oracle/tpcc_disks/ordr_0_21
ln -s /dev/raw/raw296 /home/oracle/tpcc_disks/icust2_0_21
ln -s /dev/raw/raw330 /home/oracle/tpcc_disks/iordr2_0_21
ln -s /dev/raw/raw364 /home/oracle/tpcc_disks/temp_0_21
ln -s /dev/raw/raw398 /home/oracle/tpcc_disks/etc_0_21
ln -s /dev/raw/raw25 /home/oracle/tpcc_disks/stok_0_22
ln -s /dev/raw/raw59 /home/oracle/tpcc_disks/stok_0_56
ln -s /dev/raw/raw93 /home/oracle/tpcc_disks/stok_0_90
ln -s /dev/raw/raw127 /home/oracle/tpcc_disks/cust_0_22
ln -s /dev/raw/raw161 /home/oracle/tpcc_disks/cust_0_56
ln -s /dev/raw/raw195 /home/oracle/tpcc_disks/cust_0_90
ln -s /dev/raw/raw229 /home/oracle/tpcc_disks/hist_0_22
ln -s /dev/raw/raw263 /home/oracle/tpcc_disks/ordr_0_22
ln -s /dev/raw/raw297 /home/oracle/tpcc_disks/icust2_0_22
```

```
ln -s /dev/raw/raw331 /home/oracle/tpcc_disks/iordr2_0_22
ln -s /dev/raw/raw365 /home/oracle/tpcc_disks/temp_0_22
ln -s /dev/raw/raw399 /home/oracle/tpcc_disks/iitem_0_0
ln -s /dev/raw/raw26 /home/oracle/tpcc_disks/stok_0_23
ln -s /dev/raw/raw60 /home/oracle/tpcc_disks/stok_0_57
ln -s /dev/raw/raw94 /home/oracle/tpcc_disks/stok_0_91
ln -s /dev/raw/raw128 /home/oracle/tpcc_disks/cust_0_23
ln -s /dev/raw/raw162 /home/oracle/tpcc_disks/cust_0_57
ln -s /dev/raw/raw196 /home/oracle/tpcc_disks/cust_0_91
ln -s /dev/raw/raw230 /home/oracle/tpcc_disks/hist_0_23
ln -s /dev/raw/raw264 /home/oracle/tpcc_disks/ordr_0_23
ln -s /dev/raw/raw298 /home/oracle/tpcc_disks/icust2_0_23
ln -s /dev/raw/raw332 /home/oracle/tpcc_disks/iordr2_0_23
ln -s /dev/raw/raw366 /home/oracle/tpcc_disks/temp_0_23
ln -s /dev/raw/raw400 /home/oracle/tpcc_disks/etc_0_23
ln -s /dev/raw/raw27 /home/oracle/tpcc_disks/stok_0_24
ln -s /dev/raw/raw61 /home/oracle/tpcc_disks/stok_0_58
ln -s /dev/raw/raw95 /home/oracle/tpcc_disks/stok_0_92
ln -s /dev/raw/raw129 /home/oracle/tpcc_disks/cust_0_24
ln -s /dev/raw/raw163 /home/oracle/tpcc_disks/cust_0_58
ln -s /dev/raw/raw197 /home/oracle/tpcc_disks/cust_0_92
ln -s /dev/raw/raw231 /home/oracle/tpcc_disks/hist_0_24
ln -s /dev/raw/raw265 /home/oracle/tpcc_disks/ordr_0_24
ln -s /dev/raw/raw299 /home/oracle/tpcc_disks/icust2_0_24
ln -s /dev/raw/raw333 /home/oracle/tpcc_disks/iordr2_0_24
ln -s /dev/raw/raw367 /home/oracle/tpcc_disks/temp_0_24
ln -s /dev/raw/raw401 /home/oracle/tpcc_disks/etc_0_24
ln -s /dev/raw/raw411 /home/oracle/tpcc_disks/nord_0_0
ln -s /dev/raw/raw412 /home/oracle/tpcc_disks/icust1_0_0
ln -s /dev/raw/raw413 /home/oracle/tpcc_disks/istok_0_0
```

createtable_cust.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreatetable.sh Sat Nov
19 02:35:02 JST 2005 */
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 648000000
  hash is ( (c_id * ( 21600 * 10 ) + c_w_id * 10 + c_d_id) )
  size 180
  pctfree 0  initrans 3
  storage ( buffer_pool recycle ) parallel ( degree 8 )
  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_hist.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreatetable.sh Sat Nov
19 02:35:07 JST 2005 */
set timing on
   set sqlblanklines on
   spool createtable_hist.log
   set echo on
     drop table hist ;

create table hist (
  h_c_id number
, h_c_d_id number
, h_c_w_id number
, h_d_id number
, h_w_id number
, h_date date
, h_amount number
, h_data varchar2(24)
)
  pctfree 5  initrans 4
  storage ( initial 100002k next 100000k maxextents 850 pctincrease 0 freelist groups
16 freelists 21 buffer_pool recycle )
  tablespace hist_0 ;
   set echo off
   spool off
   exit sql.sqlcode;
```

createtable_nord.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreatetable.sh Sat Nov
19 02:35:13 JST 2005 */
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 216000
   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_ordr.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreatetable.sh Sat Nov
19 02:35:11 JST 2005 */
set timing on
   set sqlblanklines on
   spool createtable_ordr.log
   set echo on
      drop cluster ordrcluster_queue including tables ;

  create cluster ordrcluster_queue (
    o_w_id number
, o_d_id number
, o_id number SORT
, o_number number SORT
   )

   hashkeys 216000
   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_ordl.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreatetable.sh Sat Nov
19 02:35:12 JST 2005 */
set timing on
   set sqlblanklines on
   spool createtable_ordl.log
   set echo on
      create table ordl (
   ol_w_id number
, ol_d_id number
, ol_o_id number sort
, ol_number number sort
, ol_i_id number
, ol_delivery_d date
, ol_amount number
, ol_supply_w_id number
, ol_quantity number
, ol_dist_info char(24)
  , constraint ordl_uk primary key (ol_w_id, ol_d_id, ol_o_id, ol_number  ))
CLUSTER ordrcluster_queue(ol_w_id, ol_d_id, ol_o_id, ol_number) ;
   set echo off
   spool off
   exit sql.sqlcode;
```

createtable_stok.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreatetable.sh Sat Nov
19 02:35:08 JST 2005 */
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 2160000000
  hash is ( (s_i_id * 21600 + s_w_id) )
  size 256
  pctfree 0  initrans 2 maxtrans 2
  storage ( buffer_pool keep ) parallel ( degree 8 )
  tablespace stok_0;

create table stok (
  s_i_id number
, s_w_id number
, s_quantity number
, s_ytd number
, s_order_cnt number
, s_remote_cnt number
, s_data varchar2(50)
, s_dist_01 char(24)
, s_dist_02 char(24)
, s_dist_03 char(24)
, s_dist_04 char(24)
, s_dist_05 char(24)
, s_dist_06 char(24)
```

```
, s_dist_07 char(24)
, s_dist_08 char(24)
, s_dist_09 char(24)
, s_dist_10 char(24)
)
cluster stokcluster (
  s_i_id
, s_w_id
);
    set echo off
    spool off
    exit sql.sqlcode;
```

createtable_dist.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreatetable.sh Sat Nov
19 02:35:05 JST 2005 */
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 216000
  hash is ( ((d_w_id * 10) + d_id) )
  size 1448
    initrans 4
  storage ( buffer_pool default )
  tablespace dist_0;

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

createtable_item.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreatetable.sh Sat Nov
19 02:35:10 JST 2005 */
set timing on
    set sqlblanklines on
    spool createtable_item.log
    set echo on
      drop cluster itemcluster including tables ;

create cluster itemcluster (
  i_id number(6,0)
  )
  single table
  hashkeys 100000
  hash is ( (i_id) )
  size 120
  pctfree 0  initrans 3
  storage ( buffer_pool keep )
  tablespace item_0;

create table item (
  i_id number(6,0)
, i_name varchar2(24)
, i_price number
, i_data varchar2(50)
, i_im_id number
)
cluster itemcluster (
  i_id
```

```
);
   set echo off
   spool off
   exit sql.sqlcode;
```

createtable_ware.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreatetable.sh Sat Nov
19 02:35:01 JST 2005 */
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 21600
  hash is ( (w_id - 1) )
  size 1448
    initrans 2
  storage ( buffer_pool default )
  tablespace ware_0;

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

createindex_iware.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreateindex.sh Sat Nov
19 02:35:16 JST 2005 */
set timing on
   set sqlblanklines on
   spool createindex_iware.log ;
   set echo on ;
   drop index iware ;
     create unique index iware on ware ( w_id )
  pctfree 1  initrans 3
  storage ( buffer_pool default )
  parallel 1
  compute statistics
  tablespace iware_0 ;
   set echo off
   spool off
   exit sql.sqlcode;
```

createindex_iitem.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreateindex.sh Sat Nov
19 02:35:19 JST 2005 */
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_idist.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreateindex.sh Sat Nov
```

```
19 02:35:18 JST 2005 */
set timing on
    set sqlblanklines on
    spool createindex_idist.log ;
    set echo on ;
    drop index idist ;
      create unique index idist on dist ( d_w_id
, d_id )
  pctfree 5  initrans 3
  storage ( buffer_pool default )
  parallel 1
  compute statistics
  tablespace idist_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createindex_iordl.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreateindex.sh Sat Nov
19 02:35:21 JST 2005 */
set timing on
  exit 0;
```

createindex_icust1.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreateindex.sh Sat Nov
19 02:35:16 JST 2005 */
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 32
  compute statistics
  tablespace icust1_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createindex_icust2.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreateindex.sh Sat Nov
19 02:35:17 JST 2005 */
set timing on
    set sqlblanklines on
    spool createindex_icust2.log ;
    set echo on ;
    drop index icust2 ;
      create unique index icust2 on cust ( c_last
, c_w_id
, c_d_id
, c_first
, c_id )
  pctfree 1  initrans 3
  storage ( buffer_pool default )
  parallel 32
  compute statistics
  tablespace icust2_0 ;
    set echo off
    spool off
    exit sql.sqlcode;
```

createindex_inord.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreateindex.sh Sat Nov
19 02:35:22 JST 2005 */
set timing on
  exit 0;
```

createindex_iordr1.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreateindex.sh Sat Nov
19 02:35:20 JST 2005 */
set timing on
  exit 0;
```

createindex_iordr2.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreateindex.sh Sat Nov
19 02:35:21 JST 2005 */
set timing on
   set sqlblanklines on
   spool createindex_iordr2.log ;
   set echo on ;
   drop index iordr2 ;
     create unique index iordr2 on ordr ( o_c_id
, o_d_id
, o_w_id
, o_id )
  pctfree 25  initrans 4
  storage ( buffer_pool default )
  parallel 32
  compute statistics
  tablespace iordr2_0 ;
   set echo off
   spool off
   exit sql.sqlcode;
```

createindex_istok.sql

```
/* created automatically by /home/oracle/tpcc-kit/scripts/buildcreateindex.sh Sat Nov
19 02:35:19 JST 2005 */
set timing on
   set sqlblanklines on
   spool createindex_istok.log ;
   set echo on ;
   drop index istok ;
     create unique index istok on stok ( s_i_id
, s_w_id )
  pctfree 1  initrans 3
  storage ( buffer_pool default )
  parallel 32
  compute statistics
  tablespace istok_0 ;
   set echo off
   spool off
   exit sql.sqlcode;
```

dml.sql

```
REM=============================================================
=========+
REM       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
REM                  OPEN SYSTEMS PERFORMANCE GROUP                |
REM                       All Rights Reserved                      |
REM=============================================================
=========+
REM FILENAME
REM     dml.sql
REM DESCRIPTION
REM     Disable table locks for TPC-C tables.
REM USAGE
REM     sqlplus tpcc/tpcc dml.sql
REM=============================================================
==========

connect tpcc/tpcc;
set echo on;

   alter table ware disable table lock;
   alter table dist disable table lock;
   alter table cust disable table lock;
   alter table hist disable table lock;
   alter table item disable table lock;
   alter table stok disable table lock;
   alter table ordr disable table lock;
   alter table nord disable table lock;
   alter table ordl disable table lock;

set echo off;

connect $oracle_dba/$oracle_dba_password;
```

driver.sh

```
#!/bin/sh

. ./stepenv.sh

if expr $# \< 1 > /dev/null; then
  echo "$0 <starting stepname> <optional: only>"
  echo OR use:
  echo "$0 buildcreate  - to build the database creation scripts"
```

```
  echo "$0 create       - to create the database (after buildcreate)"
  echo "$0 steps        - to list individual steps"
  exit 1
fi

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

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

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

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

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

echo Starting from step: $startstep

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

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

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

```
    echo The driver reached the last desired step.  Stopping driver.
    exit 0
   fi
  fi
done

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

stepenv.sh

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


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


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

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

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

#8gb oracle filesize limit (in k)
tpcc_fsize_limit_k=8000000
#2gb - 1k oracle extent limit (in k)
tpcc_extent_limit_k=2097151

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

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

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

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

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

tpcc_undo_bs=8K

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

tpcc_sysaux_size=120M

# fixed table params

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

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

tpcc_item_growth=0
tpcc_iitem_growth=0
tpcc_temp_growth=0

tpcc_cust_growth=regular
tpcc_icust1_growth=regular
tpcc_icust2_growth=regular

tpcc_stok_growth=regular
tpcc_istok_growth=regular

tpcc_ware_growth=regular
tpcc_iware_growth=regular

tpcc_dist_growth=regular
tpcc_idist_growth=regular

# minimum size of temp tablespace
tpcc_tempts_min=10240

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

#  tpcc_stok_tsfileinc=64
#  tpcc_cust_tsfileinc=64
#  tpcc_iordl2_tsfileinc=16
#  tpcc_icust2_tsfileinc=16
#  tpcc_iordl_tsfileinc=16
```

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

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

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

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

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

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

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

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

# remove runscript-loadfixordrordl - shuang, 030626

tpcc_steps="runsqllocal-createdb shutdowndb startupdb-p_build createuser ddview
runscript-createts assigntemp \
  runsql-createtable_ware runsql-createtable_cust runsql-createtable_dist runsql-
createtable_hist runsql-createtable_stok runsql-createtable_item runsql-
createtable_ordr runsql-createtable_ordl runsql-createtable_nord \
runscript-loadware runscript-loaddist runscript-loaditem runscript-loadhist runscript-
loadnord runscript-loadordrordl runscript-loadcust runscript-loadstok \
```

```
analyze runsql-createindex_iware runsql-createindex_icust1 runsql-
createindex_icust2 runsql-createindex_idist runsql-createindex_istok runsql-
createindex_iitem runsql-createindex_iordr1 runsql-createindex_iordr2 runsql-
createindex_iordl runsql-createindex_inord \
createstats createstoredprocs createspacestats createmisc"


tpcc_total_files=524

# no longer automatically exports env variables
set +a

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

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

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

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

createdb.sql

```
/* created automatically by  /home/oracle/tpcc-kit/scripts/buildcreatedb.sh Sat Nov 19
02:35:00 JST 2005 */
spool createdb.log

set echo on

shutdown abort

startup pfile=p_create.ora nomount
create database tpcc
  controlfile reuse
  maxinstances 1
  datafile
   '/home/oracle/tpcc_disks/system_1' size 400M reuse
  logfile '/home/oracle/tpcc_disks/log_1_1' size 39550M reuse,
          '/home/oracle/tpcc_disks/log_1_2' size 39550M reuse
  sysaux datafile '/home/oracle/tpcc_disks/tpccaux' size 120M reuse ;




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

set echo off
exit sql.sqlcode
```

addfile.sh

```
#!/bin/sh
# $1 = tablespace name
# $2 = filename
```

```
# $3 = size
# $4 = temporary ts (1) or not (0)
# global variable $tpcc_listfiles, does not execute sql

if expr x$tpcc_listfiles = xt > /dev/null; then
  echo $2 $3 >> $tpcc_bench/files.dat
  exit 0
fi

if expr $4 = 1 > /dev/null; then
  altersql="alter tablespace $1 add tempfile '$2' size $3 reuse;"
else
  altersql="alter tablespace $1 add datafile '$2' size $3 reuse autoextend on;"
fi

$tpcc_sqlplus $tpcc_user_pass <<!
   spool addfile_$1.log
   set echo on
  $altersql
   set echo off
   spool off
   exit ;
```

addts.sh

```
#!/bin/sh
# $1 = tablespace name
# $2 = filename
# $3 = size
# $4 = uniform size
# $5 = block size
# $6 = temporary ts (1) or not (0)
# $7 = bitmapped manage (t) or not (f) or (d) for dictionary
# global variable $tpcc_listfiles, does not execute sql

if expr x$tpcc_listfiles = xt > /dev/null; then
  echo $2 $3 >> $tpcc_bench/files.dat
  exit 0
fi

if expr $5 = auto > /dev/null; then
  bssql=
else
  bssql="blocksize $5"
fi

if expr $6 = 1 > /dev/null; then
  createsql="create temporary tablespace $1 tempfile '$2' size $3 reuse extent
management local uniform size $4;"
else
  if expr x$7 = xt > /dev/null; then
    createsql="create tablespace $1 datafile '$2' size $3 reuse extent management local
uniform size $4 segment space management auto $bssql nologging ;"
  else
    if expr x$7 = xd > /dev/null; then
      createsql="create tablespace $1 datafile '$2' size $3 reuse extent management
dictionary nologging $bssql;"
    else
      createsql="create tablespace $1 datafile '$2' size $3 reuse extent management local
uniform size $4 segment space management manual $bssql nologging ;"
    fi
  fi
fi

$tpcc_sqlplus $tpcc_user_pass <<!
   spool createts_$1.log
   set echo on
   drop tablespace $1 including contents;
 $createsql
   set echo off
   spool off
   exit ;
!
```

inittpcc.ora

```
ifile=/home/oracle/tpcc-kit/init256r1.ora
```

run.ora

```
control_files                  =
(/home/oracle/tpcc_disks/control_001,/home/oracle/tpcc_disks/control_002)
db_name                                 = tpcc
processes                      = 600
sessions                = 600
```

transactions = 300
db_files = 600
compatible = 10.1.0.0.0
dml_locks = 500
db_block_size = 2048
aq_tm_processes = 0
max_dump_file_size = 1M

db_keep_cache_size = 147500M
db_recycle_cache_size= 40000M
db_cache_size = 42528M
db_8k_cache_size = 500M
db_16k_cache_size = 15360M

log_buffer = 8688608
log_checkpoint_interval = 0
log_checkpoint_timeout = 1700
log_checkpoints_to_alert = true

shared_pool_size = 6500M
java_pool_size = 0
db_writer_processes = 4
db_block_checking = false
db_block_checksum = false
undo_management = auto
undo_retention = 1
undo_tablespace = undo_1
cursor_space_for_time = true
plsql_optimize_level = 2
replication_dependency_tracking = false
fast_start_mttr_target = 0
parallel_max_servers = 0
timed_statistics = false
statistics_level = basic
query_rewrite_enabled = false
aq_tm_processes = 0
trace_enabled = FALSE

p_create.ora

compatible = 10.1.0.0.0
db_name = tpcc
control_files = (/home/oracle/tpcc_disks/control_001,
/home/oracle/tpcc_disks/control_002)
db_block_size = 2048
db_cache_size = 43690M
db_8k_cache_size = 16384M
log_buffer = 1048576
db_16k_cache_size = 43690M
undo_management = manual
statistics_level = basic
shared_pool_size = 8192M
plsql_optimize_level=2
db_4k_cache_size = 20M

p_build.ora

compatible = 10.1.0.0.0
db_name = tpcc
control_files =
(/home/oracle/tpcc_disks/control_001,/home/oracle/tpcc_disks/control_002)
parallel_max_servers = 100
recovery_parallelism = 40
db_files = 483
db_cache_size = 43690M
db_8k_cache_size = 16384M
db_16k_cache_size = 43690M
dml_locks = 500
statistics_level = basic
log_buffer = 1048576
processes = 200
sessions = 200
transactions = 200
shared_pool_size = 8192M
cursor_space_for_time = TRUE
db_block_size = 2048
undo_management = auto
undo_retention = 2
plsql_optimize_level=2

UNDO_TABLESPACE = undo_1
db_4k_cache_size = 20M

loadware.sh

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -w > loadware.log 2>&1
```

loaddist.sh

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -d > loaddist.log 2>&1
```

loaditem.sh

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -i > loaditem.log 2>&1
```

loadhist.sh

```
#created automatically by /home/oracle/tpcc-kit/scripts/evenload.sh Sat Nov 19
02:35:14 JST 2005
rm -f loadhist*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 21600 -h  -b 1 -e 1350 >> loadhist0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -h  -b 1351 -e 2700 >> loadhist1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -h  -b 2701 -e 4050 >> loadhist2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -h  -b 4051 -e 5400 >> loadhist3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -h  -b 5401 -e 6750 >> loadhist4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -h  -b 6751 -e 8100 >> loadhist5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -h  -b 8101 -e 9450 >> loadhist6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -h  -b 9451 -e 10800 >> loadhist7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -h  -b 10801 -e 12150 >> loadhist8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -h  -b 12151 -e 13500 >> loadhist9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -h  -b 13501 -e 14850 >> loadhist10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -h  -b 14851 -e 16200 >> loadhist11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -h  -b 16201 -e 17550 >> loadhist12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -h  -b 17551 -e 18900 >> loadhist13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -h  -b 18901 -e 20250 >> loadhist14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -h  -b 20251 -e 21600 >> loadhist15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

loadnord.sh

```
#created automatically by /home/oracle/tpcc-kit/scripts/evenload.sh Sat Nov 19
02:35:14 JST 2005
rm -f loadnord*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 21600 -n  -b 1 -e 21600 >> loadnord.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

loadordrordl.sh

```
#created automatically by /home/oracle/tpcc-kit/scripts/evenload.sh Sat Nov 19
02:35:15 JST 2005
rm -f loadordrordl*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 21600 -o ${tpcc_disks_location}dummy0.dat -b 1 -e 1350 >>
loadordrordl0.log 2>&1 &
```

```
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -o ${tpcc_disks_location}dummy1.dat -b 1351 -e 2700 >>
loadordrordl1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -o ${tpcc_disks_location}dummy2.dat -b 2701 -e 4050 >>
loadordrordl2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -o ${tpcc_disks_location}dummy3.dat -b 4051 -e 5400 >>
loadordrordl3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -o ${tpcc_disks_location}dummy4.dat -b 5401 -e 6750 >>
loadordrordl4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -o ${tpcc_disks_location}dummy5.dat -b 6751 -e 8100 >>
loadordrordl5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -o ${tpcc_disks_location}dummy6.dat -b 8101 -e 9450 >>
loadordrordl6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -o ${tpcc_disks_location}dummy7.dat -b 9451 -e 10800 >>
loadordrordl7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -o ${tpcc_disks_location}dummy8.dat -b 10801 -e 12150 >>
loadordrordl8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -o ${tpcc_disks_location}dummy9.dat -b 12151 -e 13500 >>
loadordrordl9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -o ${tpcc_disks_location}dummy10.dat -b 13501 -e 14850 >>
loadordrordl10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -o ${tpcc_disks_location}dummy11.dat -b 14851 -e 16200 >>
loadordrordl11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -o ${tpcc_disks_location}dummy12.dat -b 16201 -e 17550 >>
loadordrordl12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -o ${tpcc_disks_location}dummy13.dat -b 17551 -e 18900 >>
loadordrordl13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -o ${tpcc_disks_location}dummy14.dat -b 18901 -e 20250 >>
loadordrordl14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -o ${tpcc_disks_location}dummy15.dat -b 20251 -e 21600 >>
loadordrordl15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

loadcust.sh

```
#created automatically by /home/oracle/tpcc-kit/scripts/evenload.sh Sat Nov 19
02:35:15 JST 2005
rm -f loadcust*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 21600 -C  -l 1 -m 187 >> loadcust0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -C  -l 188 -m 374 >> loadcust1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -C  -l 375 -m 561 >> loadcust2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -C  -l 562 -m 748 >> loadcust3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -C  -l 749 -m 935 >> loadcust4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -C  -l 936 -m 1122 >> loadcust5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -C  -l 1123 -m 1309 >> loadcust6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -C  -l 1310 -m 1496 >> loadcust7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -C  -l 1497 -m 1684 >> loadcust8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -C  -l 1685 -m 1872 >> loadcust9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -C  -l 1873 -m 2060 >> loadcust10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -C  -l 2061 -m 2248 >> loadcust11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -C  -l 2249 -m 2436 >> loadcust12.log 2>&1 &
allprocs="$allprocs ${!}"
```

```
$tpcc_load -M 21600 -C  -l 2437 -m 2624 >> loadcust13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -C  -l 2625 -m 2812 >> loadcust14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -C  -l 2813 -m 3000 >> loadcust15.log 2>&1 &
allprocs="$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-kit/scripts/evenload.sh Sat Nov 19
02:35:15 JST 2005
rm -f loadstok*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 21600 -S  -j 1 -k 6250 >> loadstok0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -S  -j 6251 -k 12500 >> loadstok1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -S  -j 12501 -k 18750 >> loadstok2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -S  -j 18751 -k 25000 >> loadstok3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -S  -j 25001 -k 31250 >> loadstok4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -S  -j 31251 -k 37500 >> loadstok5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -S  -j 37501 -k 43750 >> loadstok6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -S  -j 43751 -k 50000 >> loadstok7.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -S  -j 50001 -k 56250 >> loadstok8.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -S  -j 56251 -k 62500 >> loadstok9.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -S  -j 62501 -k 68750 >> loadstok10.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -S  -j 68751 -k 75000 >> loadstok11.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -S  -j 75001 -k 81250 >> loadstok12.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -S  -j 81251 -k 87500 >> loadstok13.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -S  -j 87501 -k 93750 >> loadstok14.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 21600 -S  -j 93751 -k 100000 >> loadstok15.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
  wait $curproc
  error=`expr $? + $error`
done
exit `expr $error != 0`
```

createts.sh

```
#!/bin/sh
#NOTE - ANY CHANGES MUST BE MADE TO CREATETS.KSH AS WELL.
# createts.sh [name] [no. of file] [no. of partition] [filesize] [ext_size]
#             [unix/nt] [1: temporary ts / 0: others] [filecount] [no of cpu]
#                 [blocksize] [t: bitmapped / f: manual manage / d: dictionary ]

name=$1
fileno=$2
noofts=$3
filesize=$4
extsize=$5
ver=$6
isTemp=$7
filecount=$8
para=`expr $9 \* 2`
#blocksize=${10}  sh bug workaround
blocksize=`echo $@ | cut -d' ' -f10`
#autospace=${11} sh bug workaround
autospace=`echo $@ | cut -d' ' -f11`

addts=$tpcc_scripts/addts.sh
addfile=$tpcc_scripts/addfile.sh

if expr "x$tpcc_createts_print" = "xt" > /dev/null ; then
  createtsout=${tpcc_genscripts_dir}/createts_node${tpcc_rac_node}.sh
```

```
  fileavg=`expr $fileno / $tpcc_np`

  if test $noofts -gt 1 ; then
    avg_ts_node=`expr $noofts / $tpcc_np`
    if test "x$tpcc_rac_createts_phase" = "x1" ; then
      fileavg=$avg_ts_node
    else
      if test "x$tpcc_rac_createts_phase" = "x2" ; then
       fileavg=`expr $fileavg - $avg_ts_node`
      fi
    fi
  fi

  fileend=`expr $fileavg \* $tpcc_rac_node`
  filestart=`expr $fileend - $fileavg`
  if expr $tpcc_rac_node = $tpcc_np > /dev/null; then
    fileend=$fileno
  fi
fi

#if test $ver = unix;
#then
  fileaddr="$tpcc_disks_location/";
#elif test $ver = nt;
#then
#  fileaddr=\\\\.\\
#fi

filecounter=0
i=0
while test $i -lt $noofts; do

  filecount=`expr $filecount + 1`;
  if expr "x$tpcc_createts_print" = "xt" > /dev/null ; then
    if test "x$tpcc_rac_createts_phase" = "x1" ; then
      if test "x$name" = "xitem" -o "x$name" = "xiitem" -o "x$name" = "xtemp" -o
"x$name" = "xrestbl" ; then
        if test $tpcc_rac_node = 1 ; then
          echo $addts $name\_$i $fileaddr$name\_$i\_0 $filesize $extsize $blocksize
$isTemp $autospace \& >> $createtsout
          rac_count=`expr $rac_count + 1`
          if test "$rac_count" = "$para" ; then
            rac_count=0
            echo wait >> $createtsout
          fi
        fi
      else
        if test $filecounter -ge $filestart -a $filecounter -lt $fileend ; then
          echo $addts $name\_$i $fileaddr$name\_$i\_0 $filesize $extsize $blocksize
$isTemp $autospace \& >> $createtsout
          rac_count=`expr $rac_count + 1`
          if test "$rac_count" = "$para" ; then
            rac_count=0
            echo wait >> $createtsout
          fi
        fi
      fi
    fi
  else
    $addts $name\_$i $fileaddr$name\_$i\_0 $filesize $extsize $blocksize $isTemp
$autospace \> junk$filecount 2\>\&1 \&;
  fi
  eval "proc$filecount=$!"
  filecounter=`expr $filecounter + 1`

  p=`expr $filecount % $para`;
  if test $p = 0;
  then
    k=`expr $filecount - $para + 1`;
    if test $k -le $8;
    then
      k=`expr $8 + 1`;
    fi
    while test $k -le $filecount ; do
#      wait `eval echo '$proc'$k`
      wait
      eval "proc$k=$?"
      k=`expr $k + 1`;
    done
  fi

  i=`expr $i + 1`;

done

p=`expr $filecount % $para`
```

```
if test $p != 0;
then
  k=`expr $filecount - $p + 1`;
  if test $k -le $8;
  then
    k=`expr $8 + 1`;
  fi
  while test $k -le $filecount; do
#    wait `eval echo '$proc'$k`
    wait
    eval "proc$k=$?"
    k=`expr $k + 1`
  done
fi

if test "x$tpcc_createts_print" = "xt" -a "x$tpcc_rac_createts_phase" = "x1" ; then
  echo $rac_count
  exit 0
fi

if test "x$tpcc_createts_print" = "xt" -a $noofts -gt 1 -a "x$tpcc_rac_createts_phase"
= "x2" ; then
  filecounter=0
fi

filecount=0
fileperts=`expr $fileno / $noofts - 1`
if test $fileperts -gt 0 ;
then
  i=0
  while test $i -lt $noofts ; do
    j=0;
    while test $j -lt $fileperts ;do

      filecount=`expr $filecount + 1`;
      if expr "x$tpcc_createts_print" = "xt" > /dev/null ; then
        if test "x$tpcc_rac_createts_phase" = "x2" ; then
          if test "x$name" = "xitem" -o "x$name" = "xtemp" -o "x$name" = "xrestbl" ;
then
            if test $tpcc_rac_node = 1 ; then
              echo $addfile $name\_$i $fileaddr$name\_$i\_`expr $j + 1` $filesize
$isTemp \& >> $createtsout
              rac_count=`expr $rac_count + 1`
              if test "$rac_count" = "$para" ; then
                rac_count=0
                echo wait >> $createtsout
              fi
            fi
          else
            if test $filecounter -ge $filestart -a $filecounter -lt $fileend ; then
              echo $addfile $name\_$i $fileaddr$name\_$i\_`expr $j + 1` $filesize
$isTemp \& >> $createtsout
              rac_count=`expr $rac_count + 1`
              if test "$rac_count" = "$para" ; then
                rac_count=0
                echo wait >> $createtsout
              fi
            fi
          fi
        fi
      else
        $addfile $name\_$i $fileaddr$name\_$i\_`expr $j + 1` $filesize $isTemp \>
junk$filecount 2\>\&1 &
      fi
      eval "proc$filecount=$!"

      filecounter=`expr $filecounter + 1`

      p=`expr $filecount % $para`;
      if test $p = 0;
      then
        k=`expr $filecount - $para + 1`;
        if test $k -le $8;
        then
          k=`expr $8 + 1`;
        fi
       while test $k -le $filecount ; do
#          wait `eval echo '$proc'$k`
          wait
          eval "proc$k=$?"
          k=`expr $k + 1`;
        done
      fi

      j=`expr $j + 1`
    done
```

```
  i=`expr $i + 1`
 done

 p=`expr $filecount % $para`
 if test $p != 0;
 then
  k=`expr $filecount - $p + 1`;
  if test $k -le $8;
  then
   k=`expr $8 + 1`;
  fi
  while test $k -le $filecount; do
#    wait `eval echo '$proc'$k`
   wait
   eval "proc$k=$?"
   k=`expr $k + 1`
  done
 fi
fi

if test "x$tpcc_createts_print" = "xt" ; then
 echo $rac_count
fi

i=`expr $8 + 1`
proc=0
while test $i -le $filecount ;do
 eval 'process=$proc"'$i"
 proc=`expr $proc + $process`
 i=`expr $i + 1`
done

out=`expr $proc % 127`
# Added wait here for all tablespaces to be created
wait
if test $out -ne 0
then
 exit 1;
else
 exit 0;
fi
```

extent.sql

```
REM          Copyright (c) 1994  Oracle Corp, Belmont, CA         |
REM               OPEN SYSTEMS PERFORMANCE GROUP                  |
REM                    All Rights Reserved                        |
REM============================================================
==========+
REM FILENAME
REM      extent.sql
REM DESCRIPTION
REM      List all extents in all the TPCC tablespaces.
REM
REM  Usage: sqlplus 'sys/change_on_install as sysdba' @extent
REM============================================================
==========*/
     set space 2
     set pagesize 2000
     set echo off
     set termout off
     set verify off
     set feedback off
     spool extent.rpt
     select substr(e.tablespace_name,1,8) tspace,
       substr(segment_name,1,11) segment, substr(segment_type,1,15) type,
       substr(extent_id,1,5) eid, substr(file_id,1,5) fid, blocks,
       blocks * t.block_size / 1048576 size_MB
     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 = t.tablespace_name
     order by e.tablespace_name, segment_name, extent_id, file_id;

     select substr(e.tablespace_name,1,8) tspace,
       substr(segment_name,1,11) segment,
       sum(blocks) tot_blk, sum(blocks) * t.block_size / 1048576 size_MB
     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 = t.tablespace_name
```

```
group by e.tablespace_name, segment_name, t.block_size
order by e.tablespace_name, segment_name;
spool off;
```

pst_c.sql

```
rem
rem
rem
================================================================
=====+
rem          Copyright (c) 1992  Oracle Corp, Belmont, CA          |
rem               OPEN SYSTEMS PERFORMANCE GROUP                   |
rem                  All Rights Reserved                           |
rem
================================================================
=====+
rem  FILENAME
rem      pst_c.sql
rem  DESCRIPTION
rem      Create Table for OS Specific Process Stats
rem
================================================================
=====*/
rem
rem  Tables for Unix-specific process statistics
rem
rem  Usage: sqlplus internal/internal @pst_c
rem

      connect tpcc/tpcc;
      set echo on;
      DROP TABLE proc_resource;
      DROP TABLE os_stat;

rem
rem  Resource usage for a process.
rem

      CREATE TABLE proc_resource
      (
         config       VARCHAR2(10),
         run          NUMBER,
         proc         NUMBER,
         child        NUMBER,
         user_cpu_ms   NUMBER,
         system_cpu_ms NUMBER,
         maxrss       NUMBER,
         pagein       NUMBER,
         reclaim      NUMBER,
         zerofill     NUMBER,
         pffincr      NUMBER,
         pffdecr      NUMBER,
         swap         NUMBER,
         syscall      NUMBER,
         volcsw       NUMBER,
         involcsw     NUMBER,
         signal       NUMBER,
         lread        NUMBER,
         lwrite       NUMBER,
         bread        NUMBER,
         bwrite       NUMBER,
         phread       NUMBER,
         phwrite      NUMBER
      );

rem
rem  OS statistics.
rem  These results are from the measurement interval only.
rem

      CREATE TABLE os_stat
      (
         config       VARCHAR2(10),
         run          NUMBER,
         hid          NUMBER,
         syscall      NUMBER,
         intr         NUMBER,
         cswitch      NUMBER,
         pagefault    NUMBER,
         usr          NUMBER,
         sys          NUMBER,
         idl          NUMBER,
         wio          NUMBER
      );
```

```
set echo off;
```

space_get.sql

```
REM============================================================
=========+
REM        Copyright (c) 1995  Oracle Corp, Redwood Shores, CA     |
REM                  OPEN SYSTEMS PERFORMANCE GROUP                |
REM                       All Rights Reserved                      |
REM============================================================
=========+
REM FILENAME
REM       space_get.sql
REM DESCRIPTION
REM       Get sizes of tables, indexes and tablespaces.
REM  Usage: sqlplus 'sys/change_on_install as sysdba' @space_get [<tpm> <# of
warehouses>]
REM============================================================
=========*/

  set echo on;
  delete from tpcc_data;
  delete from tpcc_space;
  delete from tpcc_totspace;

  insert into tpcc_data
  select substr(segment_name,1,18), substr(segment_type,1,15),
        sum(blocks), t.block_size,
        round(sum(blocks) * 0.05), 0,
        sum(blocks) + round(sum(blocks) * 0.05)
  from   dba_extents e, dba_tablespaces t
  where  owner = 'TPCC' AND ( segment_type = 'INDEX' OR
        segment_type = 'INDEX PARTITION' OR segment_type = 'CLUSTER'
        OR segment_type = 'TABLE' OR segment_type = 'TABLE PARTITION')
        AND e.tablespace_name <> 'SYSTEM' AND e.tablespace_name <> 'SP_0'
        AND e.tablespace_name = t.tablespace_name
  group by segment_name, segment_type, t.block_size;

  insert into tpcc_data
    select 'SYSTEM', 'SYS', sum(blocks), t.block_size, 0, 0, sum(blocks)
    from   dba_data_files f, dba_tablespaces t
    where  f.tablespace_name = 'SYSTEM' and t.tablespace_name =
f.tablespace_name
    group by t.block_size;

  insert into tpcc_data
    select 'SYSAUX', 'SYS', sum(blocks), t.block_size, 0, 0, sum(blocks)
    from   dba_data_files f, dba_tablespaces t
    where  f.tablespace_name = 'SYSAUX' and t.tablespace_name =
f.tablespace_name
    group by t.block_size;

  insert into tpcc_data
    select 'ROLL_SEG', 'SYS', sum(blocks), t.block_size, 0, 0, sum(blocks)
    from   dba_data_files f, dba_tablespaces t
    where  f.tablespace_name like '%UNDO_TS%' and f.tablespace_name =
t.tablespace_name
    group by f.tablespace_name, t.block_size;

  insert into tpcc_data
    select 'DB_STAT', 'SYS', sum(blocks), t.block_size, 0, 0, sum(blocks)
    from   dba_data_files f, dba_tablespaces t
    where  f.tablespace_name like '%SP_0%' and f.tablespace_name =
t.tablespace_name
    group by f.tablespace_name, t.block_size;

  update tpcc_data
    set five_pct = 0,
       daily_grow = round(blocks * &&1 / 62.5 / &&2),
       total = blocks + round(blocks * &&1 / 62.5 / &&2)
    where segment = 'HIST' OR segment = 'ORDRCLUSTER_QUEUE' OR
         segment = 'IORDL';

  insert into tpcc_space
  select substr(ex$.name,1,18), sum(sp$.sz_blocks), sp$.block_size, 0, 0, 0, 0
  from
   (select f.tablespace_name , sum(blocks) sz_blocks, t.block_size block_size
    from dba_data_files f, dba_tablespaces t
    where f.tablespace_name <> 'SYSTEM' and f.tablespace_name =
t.tablespace_name
    group by f.tablespace_name, t.block_size
   ) sp$,
   (select distinct tablespace_name, segment_name name
    from dba_extents
    where owner = 'TPCC'
     and (segment_type = 'CLUSTER' or segment_type = 'TABLE'
```

```
      or segment_type = 'TABLE PARTITION' or segment_type = 'INDEX'
      or segment_type = 'INDEX PARTITION')
     and tablespace_name <> 'SYSTEM'
  ) ex$
 where sp$.tablespace_name = ex$.tablespace_name
 group by ex$.name, sp$.block_size;

 insert into tpcc_space
 select substr(f.tablespace_name,1,18), sum(blocks), t.block_size, 0, 0, 0, 0
 from dba_data_files f, dba_tablespaces t
 where (f.tablespace_name = 'SYSTEM' or f.tablespace_name = 'SYSAUX')
     and f.tablespace_name = t.tablespace_name
 group by f.tablespace_name, t.block_size;

 insert into tpcc_space
 select 'ROLL_SEG', sum(blocks), t.block_size, 0, 0, 0, 0
 from dba_data_files f, dba_tablespaces t
 where f.tablespace_name = 'UNDO_TS' and f.tablespace_name =
t.tablespace_name
 group by f.tablespace_name, t.block_size;

 insert into tpcc_space
 select 'DB_STAT', sum(blocks), t.block_size, 0, 0, 0, 0
 from dba_data_files f, dba_tablespaces t
 where f.tablespace_name = 'SP_0' and f.tablespace_name = t.tablespace_name
 group by f.tablespace_name, t.block_size;

 update tpcc_space
   set required =
   (
     select sum(total)
     from   tpcc_data
     where  tpcc_data.segment = tpcc_space.segment
   )
   where segment in
   (
     select segment from tpcc_data
   );

 update tpcc_space
   set static =
   (
     select sum(total)
     from   tpcc_data
     where  tpcc_data.segment = tpcc_space.segment
   )
   where segment in
   (
     select segment from tpcc_data
   );

 update tpcc_space
   set static = 0,
     dynamic =
   (
     select sum(blocks)
     from   tpcc_data
     where  tpcc_data.segment = tpcc_space.segment
   )
   where segment in ('HIST', 'ORDRCLUSTER_QUEUE', 'IORDL');

 update tpcc_space
   set oversize = blocks  - required;


 insert into tpcc_totspace
   select &&1, &&2, sum(static * block_size)/1024, sum(dynamic *
block_size)/1024, sum(oversize * block_size)/1024, 0, 0, 0
   from   tpcc_space;

 update tpcc_totspace
   set daily_grow =
   (
     select sum(daily_grow * block_size)/1024
     from   tpcc_data
   );
 update tpcc_totspace
   set space60 = static + 60 * daily_grow;
 set echo off;
```

space_init.sql

```
REM=============================================================
=========+
REM FILENAME
REM       space_init.sql
```

```
REM DESCRIPTION
REM       Create tables for space calculations.
REM  Usage: sqlplus 'sys/change_on_install as sysdba' @space_init.sql
REM=============================================================
=========*/
  set echo on;
  drop table tpcc_data;
  drop table tpcc_space;
  drop table tpcc_totspace;
  create table tpcc_data (
    segment      varchar2(18),
    type         varchar2(15),
    blocks       number,
    block_size   number,
    five_pct     number,
    daily_grow   number,
    total        number
  );
  create table tpcc_space (
    segment       varchar2(18),
    blocks        number,
    block_size    number,
    required      number,
    static        number,
    dynamic       number,
    oversize      number
  );
  create table tpcc_totspace (
    tpm           number,
    nware         number,
    static        number,
    dynamic       number,
    oversize      number,
    daily_grow    number,
    daily_spre    number,
    space60       number
  );
  create unique index itpcc_data on tpcc_data (segment);
  create unique index itpcc_space on tpcc_space (segment);
  set echo off;
```

space_rpt.sql

```
REM=============================================================
=========+
REM       Copyright (c) 1995  Oracle Corp, Redwood Shores, CA     |
REM                 OPEN SYSTEMS PERFORMANCE GROUP                |
REM                    All Rights Reserved                        |
REM=============================================================
=========+
REM FILENAME
REM       space_rpt.sql
REM DESCRIPTION
REM       Generate space report and save it in space.rpt
REM  Usage: sqlplus 'sys/change_on_install as sysdba' @space_rpt.sql
REM=============================================================
=========*/
  set space 2
  set pagesize 2000
  set echo off
  set termout off
  set verify off
  set feedback off
  set pagesize 60 linesize 120
  spool space.rpt
  select tpm, nware from tpcc_totspace;
  select * from tpcc_data order by segment;
  select * from tpcc_space order by segment;
  select static, dynamic, oversize, daily_grow, daily_spre, space60
    from tpcc_totspace;
  spool off;
```

analyze.sh

```
#!/bin/sh
$tpcc_sqlplus $tpcc_user_pass @${tpcc_sql_dir}/analyze > $tpcc_log_dir/junk 2>&1

# this one tends to fail if indices aren't made, which is legal, so
# always exit without error.

exit 0
```

chg_redo2pole.sh

```
sqlplus "/ as sysdba" << EOF
```

```
set echo on
set serveroutput on
spool chg_redo2pole.log

alter database add logfile ('/home/oracle/tpcc_disks/log_1_1') size 470000M;
alter database add logfile ('/home/oracle/tpcc_disks/log_1_2') size 470000M;

alter system switch logfile;
alter system switch logfile;
alter system checkpoint;

alter database drop logfile group 3;
alter database drop logfile group 4;
spool off
exit;

EOF
```

chg_redo2temp.sh

```
#!/bin/sh

#rm -i /home/oracle/temp/templog1.dbf
#rm -i /home/oracle/temp/templog2.dbf

sqlplus "/ as sysdba" << EOF
  set echo on
  set serveroutput on
  spool chg_redo2temp.log

  alter database add logfile ('/home/oracle/temp/templog1.dbf') size 70M;
  alter database add logfile ('/home/oracle/temp/templog2.dbf') size 70M;

  alter system switch logfile;
  alter system switch logfile;
  alter system checkpoint;

  alter database drop logfile group 1;
  alter database drop logfile group 2;
  spool off
  exit;

EOF
```

createstoredprocs.sh

```
#!/bin/sh
cd $tpcc_genscripts_dir
$tpcc_sqlplus $tpcc_user_pass @${tpcc_genscripts_dir}/createstoredprocs > junk
2>&1

if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi
```

createstoredprocs.sql

```
spool createstoreprocs.log
@tkvcinin.sql
spool off
exit sql.sqlcode;
```

tkvcinin.sql

```
-- The initnew package for storing variables used in the
-- New Order anonymous block

CREATE OR REPLACE PACKAGE inittpcc
AS
 TYPE intarray IS TABLE OF INTEGER INDEX BY BINARY_INTEGER;
 TYPE distarray IS TABLE OF VARCHAR(24) INDEX BY BINARY_INTEGER;
 nulldate      DATE;
 TYPE rowidarray IS TABLE OF ROWID INDEX BY PLS_INTEGER;
 s_dist                    distarray;
 idx1arr           intarray;
 s_remote                  intarray;
 dist              intarray;
 row_id              rowidarray;
 cust_rowid            rowid;
 dist_name             VARCHAR2(11);
 ware_name              VARCHAR2(11);
```

```
 c_num                PLS_INTEGER;

 PROCEDURE init_no(idxarr intarray);
 PROCEDURE init_del;
 PROCEDURE init_pay;
END inittpcc;
/
show errors;

CREATE OR REPLACE PACKAGE BODY inittpcc AS
  PROCEDURE init_no (idxarr  intarray)
  IS
  BEGIN
       -- initialize null date
   nulldate := TO_DATE('01-01-1811', 'MM-DD-YYYY');
    idx1arr := idxarr;
  END init_no;

  PROCEDURE init_del
  IS
  BEGIN
    FOR i IN 1 .. 10 LOOP
      dist(i) := i;
    END LOOP;
  END init_del;

  PROCEDURE init_pay IS
  BEGIN
    NULL;
  END init_pay;

END inittpcc;
/
show errors
exit
```

createspacestats.sh

```
#!/bin/sh
cd $tpcc_genscripts_dir
$tpcc_sqlplus $tpcc_dba_user_pass @$tpcc_genscripts_dir/createspacestats > junk
2>&1
if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi
```

createspacestats.sql

```
@space_init
@space_get 254472 20240
@space_rpt
spool off
exit sql.sqlcode;
```

createuser.sh

```
#!/bin/sh

echo Creating user tpcc...
$tpcc_sqlplus $tpcc_dba_user_pass @$tpcc_sql_dir/createuser > junk 2>&1
if test $? -ne 0
then
  exit 1;
else
  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 ;
```

assigntemp.sh

```
#!/bin/sh

echo Assigning temporary tablespace to user tpcc...
$tpcc_sqlplus $tpcc_dba_user_pass @$tpcc_sql_dir/assigntemp > junk 2>&1
if test $? -ne 0
then
  exit 1;
else
  exit 0;
fi
```

assigntemp.sql

```
spool assigntemp.log;

set echo on;

alter user tpcc temporary tablespace temp_0;

set echo off;
spool off;

exit ;
```

shutdowndb.sh

```
#!/bin/sh

echo "Shutting down database..."

$tpcc_sqlplus $tpcc_sqlplus_args << !
$tpcc_internal_connect

spool shutdowndb.log;

set echo on;

alter system switch logfile;
alter system switch logfile;
shutdown immediate;

set echo off;
spool off;

exit
!
```

startupdb.sh

```
#!/bin/sh

echo "Starting up database using $1..."

init_file=${1}.ora

if test $tpcc_np -gt 1 ; then
  init_file=build_init_${tpcc_rac_id}.ora
fi

$tpcc_sqlplus $tpcc_sqlplus_args << !
$tpcc_internal_connect

spool startdb.log

set echo on

startup pfile=$init_file open

spool off
set echo off
exit sql.sqlcode
!
```

views.sql

```
connect tpcc/tpcc;
set echo on;

create or replace view wh_cust
(w_id, w_tax, c_id, c_d_id, c_w_id, c_discount, c_last, c_credit)
```

```
as select w.w_id, w.w_tax,
        c.c_id, c.c_d_id, c.c_w_id, c.c_discount, c.c_last, c.c_credit
   from cust c, ware w
   where w.w_id = c.c_w_id;

create or replace view wh_dist
(w_id, d_id, d_tax, d_next_o_id, w_tax )
as select w.w_id, d.d_id, d.d_tax, d.d_next_o_id, w.w_tax
   from dist d, ware w
   where w.w_id = d.d_w_id;

create  or replace view stock_item
(i_id, s_w_id, i_price, i_name, i_data, s_data, s_quantity,
 s_order_cnt, s_ytd, s_remote_cnt,
 s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
 s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10)
as
 select /*+ 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;
```

ddview.sh

```
#!/bin/sh

$tpcc_sqlplus $tpcc_sqlplus_args << !
$tpcc_internal_connect

spool ddview.log


REM
REM In an ade/nde view we might need to run standard.sql and dbmsstdx manually
REM catalog and catproc suppose to take care of it
REM

@$ORACLE_HOME/plsql/admin/standard

@$ORACLE_HOME/rdbms/admin/dbmsstdx

@$ORACLE_HOME/rdbms/admin/catalog
@$ORACLE_HOME/rdbms/admin/catproc

REM
REM In an ade/nde view we might need to run pupbld manually
REM catalog and catproc suppose to take care of it
REM

connect system/manager
REM @$ORACLE_HOME/sqlplus/admin/pupbld


REM
REM Oracle
REM

REM if test $NUMBER_ORACLE_NODE -qt 1
REM then

REM @$ORACLE_HOME/rdbms/admin/catparr

REM fi

spool off
!

#sh $tpcc_scripts/queue.sh
```

createmisc.sh

```
#!/bin/sh

$tpcc_sqlplus $tpcc_sqlplus_args << !
$tpcc_internal_connect

spool createmisc.log
set echo on;
alter user tpcc temporary tablespace system;
grant execute on dbms_lock to public;
grant execute on dbms_pipe to public;
grant select on v_\$parameter to public;
```

```
REM
REM begin plsql_mon.sql
REM

connect tpcc/tpcc;
set echo on;
CREATE OR REPLACE PACKAGE plsql_mon_pack
IS
  PROCEDURE print
  (
    info        VARCHAR2
  );
END;
/
show errors;

CREATE OR REPLACE PACKAGE BODY plsql_mon_pack
IS
  PROCEDURE print
  (
    info        VARCHAR2
  )
  IS
    s           NUMBER;
  BEGIN
    dbms_pipe.pack_message (info);
    s := dbms_pipe.send_message ('plsql_mon');
    IF (s <> 0) THEN
      raise_application_error (-20000, 'Error:' || to_char(s) ||
                               ' sending on pipe');
    END IF;
  END;
END;
/
show errors;

set echo off;

REM
REM end plsql_mon.sql
REM
```

```
REM
REM  begin cre_tab.sql
REM

connect tpcc/tpcc;
set echo on;

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

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

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

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

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

create table tpcc_audit_tab (starttime date);

delete from tpcc_audit_tab;

set echo off;

REM
REM  end cre_tab.sql
REM
```

```
REM
REM  begin views.sql
REM

connect tpcc/tpcc;
set echo on;

create or replace view wh_cust
(w_id, w_tax, c_id, c_d_id, c_w_id, c_discount, c_last, c_credit)
as select w.w_id, w.w_tax,
        c.c_id, c.c_d_id, c.c_w_id, c.c_discount, c.c_last, c.c_credit
   from cust c, ware w
   where w.w_id = c.c_w_id;

create or replace view wh_dist
(w_id, d_id, d_tax, d_next_o_id, w_tax )
as select w.w_id, d.d_id, d.d_tax, d.d_next_o_id, w.w_tax
   from dist d, ware w
   where w.w_id = d.d_w_id;

create  or replace view stock_item
(i_id, s_w_id, i_price, i_name, i_data, s_data, s_quantity,
 s_order_cnt, s_ytd, s_remote_cnt,
 s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
 s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10)
as
 select i.i_id, s_w_id, i.i_price, i.i_name, i.i_data, s_data, s_quantity,
 s_order_cnt, s_ytd, s_remote_cnt,
 s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
 s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10
  from stok s, item i
 where i.i_id = s.s_i_id;

set echo off;

REM
REM  end views.sql
REM


REM
REM  begin dml.sql
REM

connect tpcc/tpcc;
set echo on;

   alter table ware disable table lock;
   alter table dist disable table lock;
   alter table cust disable table lock;
   alter table hist disable table lock;
   alter table item disable table lock;
   alter table stok disable table lock;
   alter table ordr disable table lock;
   alter table nord disable table lock;
   alter table ordl disable table lock;

set echo off;

REM
REM  end dml.sql
REM

REM
REM  begin extent.sql
REM

$SYS_CONNECTION_STRING

@$tpcc_sql_dir/extent

@$tpcc_sql_dir/freeext

exit sql.sqlcode;

!
```

createstats.sh

```
#!/bin/sh

cstat=c_stat
if test $tpcc_np -gt 1 ; then
  cstat=c_stat_rac
fi

$tpcc_sqlplus $tpcc_sqlplus_args << !
```

```
$tpcc_internal_connect

REM
REM create tablespace for statspack user sp begin
REM

spool createstats.log

set echo on
  drop tablespace sp_0 including contents;
  create tablespace sp_0 datafile '${tpcc_disks_location}sp_0' size
$tpcc_statspack_size reuse autoextend on extent management local uniform size 1M
nologging ;
  spool off

REM
REM create tablespace for statspack user sp end
REM

REM
REM  begin now call spcreate to create statspack sp package
REM

$tpcc_internal_connect

define default_tablespace='sp_0'

define temporary_tablespace='temp_0'

@$ORACLE_HOME/rdbms/admin/spcreate
perfstat

REM note that the last thing (after spcreate) is the perfstat password.
REM since we're not worried about security, perfstat will do.

REM
REM tpcc stat table for NT, it is not working so I comment it out
REM shui.lau@oracle.com it is better to use perfmon
REM

@$tpcc_sql_dir/cs_tpcc
@$tpcc_sql_dir/cs_cpu
@$tpcc_sql_dir/cs_os
@$tpcc_sql_dir/cs_proc
@$tpcc_sql_dir/cs_thread

REM
REM tpcc result table for unix and NT
REM

@$tpcc_sql_dir/${cstat}
@$tpcc_sql_dir/pst_c

!
```

createts.sh

```
#created automatically by /home/oracle/tpcc-kit/scripts/buildcreatets.sh Sat Nov 19
02:34:48 JST 2005

# Tablespace ware, ts size 50M (51200K)
# each file 50M (51200K)
# extents 45952K (45952K)
# 1 files

$tpcc_createts ware 1 1       50M 45952K unix 0       0 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for ware failed.  Exiting.
     exit 0
   fi

# Tablespace cust, ts size 564060M (577597440K)
# each file 5530M (5662720K)
# extents 115432K (115432K)
# 102 files

$tpcc_createts cust 102 1       5530M 115432K unix 0       1 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for cust failed.  Exiting.
     exit 0
   fi

# Tablespace dist, ts size 450M (460800K)
# each file 450M (460800K)
# extents 450304K (450304K)
# 1 files
```

```
$tpcc_createts dist 1 1       450M 450304K unix 0       103 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for dist failed.  Exiting.
     exit 0
   fi

# Tablespace hist, ts size 69020M (70676480K)
# each file 2030M (2078720K)
# extents 98836K (98836K)
# 34 files

$tpcc_createts hist 34 1       2030M 98836K unix 0       104 8 auto d
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for hist failed.  Exiting.
     exit 0
   fi

# Tablespace stok, ts size 632400M (647577600K)
# each file 6200M (6348800K)
# extents 129442K (129442K)
# 102 files

$tpcc_createts stok 102 1       6200M 129442K unix 0       138 8 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       240 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for item failed.  Exiting.
     exit 0
   fi

# Tablespace ordr, ts size 1000960M (1024983040K)
# each file 29440M (30146560K)
# extents 103200K (103200K)
# 34 files

$tpcc_createts ordr 34 1       29440M 103200K unix 0       241 8 16K t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for ordr failed.  Exiting.
     exit 0
   fi

# Tablespace nord, ts size 7330M (7505920K)
# each file 7330M (7505920K)
# extents 749824K (749824K)
# 1 files

$tpcc_createts nord 1 1       7330M 749824K unix 0       275 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for nord failed.  Exiting.
     exit 0
   fi

# Tablespace iware, ts size 30M (30720K)
# each file 30M (30720K)
# extents 28024K (28024K)
# 1 files

$tpcc_createts iware 1 1       30M 28024K unix 0       276 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for iware failed.  Exiting.
     exit 0
   fi

# Tablespace icust1, ts size 15260M (15626240K)
# each file 15260M (15626240K)
# extents 244032K (244032K)
# 1 files

$tpcc_createts icust1 1 1       15260M 244032K unix 0       277 8 16K t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for icust1 failed.  Exiting.
     exit 0
   fi

# Tablespace icust2, ts size 40460M (41431040K)
# each file 1190M (1218560K)
```

```
# extents 18892K (18892K)
# 34 files

$tpcc_createts icust2 34 1       1190M 18892K unix 0       278 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for icust2 failed.  Exiting.
     exit 0
   fi

# Tablespace idist, ts size 110M (112640K)
# each file 110M (112640K)
# extents 109024K (109024K)
# 1 files

$tpcc_createts idist 1 1       110M 109024K unix 0       312 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for idist failed.  Exiting.
     exit 0
   fi

# Tablespace istok, ts size 45070M (46151680K)
# each file 45070M (46151680K)
# extents 721024K (721024K)
# 1 files

$tpcc_createts istok 1 1       45070M 721024K unix 0       313 8 16K t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for istok failed.  Exiting.
     exit 0
   fi

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

$tpcc_createts iitem 1 1       20M 11264K unix 0       314 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for iitem failed.  Exiting.
     exit 0
   fi

# Tablespace iordr2, ts size 40460M (41431040K)
# each file 1190M (1218560K)
# extents 18978K (18978K)
# 34 files

$tpcc_createts iordr2 34 1       1190M 18978K unix 0       315 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for iordr2 failed.  Exiting.
     exit 0
   fi

# Tablespace temp, ts size 113220M (115937280K)
# each file 3330M (3409920K)
# extents 200332K (200332K)
# 34 files

$tpcc_createts temp 34 1       3330M 200332K unix 1       349 8 auto t
   if expr $? != 0 > /dev/null; then
     echo Creating tablespace for temp failed.  Exiting.
     exit 0
   fi
```

freeext.sql

```
REM=============================================================
=========+
REM          Copyright (c) 1994  Oracle Corp, Belmont, CA         |
REM               OPEN SYSTEMS PERFORMANCE GROUP                  |
REM                  All Rights Reserved                          |
REM=============================================================
=========+
REM FILENAME
REM      freeext.sql
REM DESCRIPTION
REM      List all free extents in all the TPCC tablespace
REM
REM  Usage: sqlplus 'sys/change_on_install as sysdba' @freeext
REM=============================================================
=========*/
    set space 2
    set pagesize 2000
    set echo off
    set termout off
    set verify off
```

```
set feedback off
spool freeextent.rpt
select substr(e.tablespace_name,1,8) tspace, file_id, block_id, blocks,
      blocks * t.block_size / 1048576 size_MB
from dba_free_space e, dba_tablespaces t
where e.tablespace_name = t.tablespace_name
order by e.tablespace_name, file_id, block_id;

select substr(e.tablespace_name,1,8) tspace, sum(blocks) tot_blk,
      sum(blocks) * t.block_size / 1048576 size_MB
from dba_free_space e, dba_tablespaces t
where e.tablespace_name = t.tablespace_name
group by e.tablespace_name, t.block_size
order by e.tablespace_name;
```

plsql_mon.sql

```
rem
rem
================================================================
=====+
rem        Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
rem                OPEN SYSTEMS PERFORMANCE GROUP                   |
rem                    All Rights Reserved                          |
rem
================================================================
=====+
rem  FILENAME
rem      plsql_mon.sql
rem  DESCRIPTION
rem      SQL script to create a stored package for PL/SQL stored
rem      procedures to dump messages.
rem
================================================================
=====
rem
rem Usage:   sqlplus tpcc/tpcc @plsql_mon
rem

connect tpcc/tpcc;
set echo on;
CREATE OR REPLACE PACKAGE plsql_mon_pack
IS
  PROCEDURE print
  (
    info        VARCHAR2
  );
END;
/
show errors;

CREATE OR REPLACE PACKAGE BODY plsql_mon_pack
IS
  PROCEDURE print
  (
    info        VARCHAR2
  )
  IS
    s           NUMBER;
  BEGIN
    dbms_pipe.pack_message (info);
    s := dbms_pipe.send_message ('plsql_mon');
    IF (s <> 0) THEN
      raise_application_error (-20000, 'Error:' || to_char(s) ||
                               ' sending on pipe');
    END IF;
  END;
END;
/
show errors;

set echo off;
```

cre_tab.sql

```
rem
rem
================================================================
=====+
rem        Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
rem                OPEN SYSTEMS PERFORMANCE GROUP                   |
rem                    All Rights Reserved                          |
rem
================================================================
=====+
rem  FILENAME
```

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

connect tpcc/tpcc;
set echo on;

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

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

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

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

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

create table tpcc_audit_tab (starttime date);

delete from tpcc_audit_tab;

set echo off;
```

cs_cpu.sql

```
rem
rem
rem=============================================================
========+
rem       Copyright (c) 1997  Oracle Corp, Redwood Shores, CA      |
rem                    All Rights Reserved                    |
rem=============================================================
========+
rem FILENAME
rem     cs_cpu.sql
rem DESCRIPTION
rem     Create Table for CPU Specific Process Stat
rem=============================================================
========
rem usage: sqlplus tpcc/tpcc @cs_cpu.sql

connect tpcc/tpcc
set echo on

DROP TABLE pre_cpu_stats;
DROP TABLE post_cpu_stats;
DROP TABLE cpu_stats;

rem
rem  CPU statistics.
rem

       CREATE TABLE cpu_stats
       (
          runname        VARCHAR2(20),
                               cpu_id         NUMBER,
          dpc_cpu        NUMBER,
          interrupt_cpu   NUMBER,
          priv_cpu       NUMBER,
          processor_cpu   NUMBER,
          user_cpu       NUMBER,
          interrupt_rate   NUMBER
       );
```

```
rem
rem  Save Begining CPU Stat Values
rem

      CREATE TABLE pre_cpu_stats
      (
         runname        VARCHAR2(20),
                              cpu_id         NUMBER,
         dpc_cpu        NUMBER,
         interrupt_cpu   NUMBER,
         priv_cpu       NUMBER,
         processor_cpu   NUMBER,
         user_cpu       NUMBER,
         interrupt_rate   NUMBER
      );


rem
rem  Save Ending CPU Stat Values
rem

      CREATE TABLE post_cpu_stats
      (
         runname        VARCHAR2(20),
                              cpu_id         NUMBER,
         dpc_cpu        NUMBER,
         interrupt_cpu   NUMBER,
         priv_cpu       NUMBER,
         processor_cpu   NUMBER,
         user_cpu       NUMBER,
         interrupt_rate   NUMBER
      );
commit;
set echo off;
```

cs_os.sql

```
rem
rem
rem================================================================
========+
rem       Copyright (c) 1997  Oracle Corp, Redwood Shores, CA      |
rem                    All Rights Reserved                         |
rem================================================================
========+
rem FILENAME
rem     cs_os.sql
rem DESCRIPTION
rem     Create Table for OS Specific Process Stat
rem================================================================
========
rem usage: sqlplus tpcc/tpcc @cs_os.sql

connect tpcc/tpcc
set echo on

DROP TABLE pre_os_stats;
DROP TABLE post_os_stats;
DROP TABLE os_stats;

rem
rem  OS statistics.
rem

      CREATE TABLE os_stats
      (
         runname        VARCHAR2(20),
         time           NUMBER,
         syscall        NUMBER,
         intr           NUMBER,
         cswitch        NUMBER,
         freads         NUMBER,
         fwrites        NUMBER,
         fcontrolops    NUMBER,
         priv_cpu       NUMBER,
         user_cpu       NUMBER,
         processor_cpu   NUMBER,
                              interrupt_cpu   NUMBER
      );

rem
rem  Save Begining OS Stat Values
rem

      CREATE TABLE pre_os_stats
      (
```

```
    runname       VARCHAR2(20),
    time          NUMBER,
    syscall       NUMBER,
    intr          NUMBER,
    cswitch       NUMBER,
    freads        NUMBER,
    fwrites       NUMBER,
    fcontrolops   NUMBER,
    priv_cpu      NUMBER,
    user_cpu      NUMBER,
    processor_cpu   NUMBER,
                  interrupt_cpu   NUMBER
);


rem
rem  Save Ending OS Stat Values
rem

    CREATE TABLE post_os_stats
    (
    runname       VARCHAR2(20),
    time          NUMBER,
    syscall       NUMBER,
    intr          NUMBER,
    cswitch       NUMBER,
    freads        NUMBER,
    fwrites       NUMBER,
    fcontrolops   NUMBER,
    priv_cpu      NUMBER,
    user_cpu      NUMBER,
    processor_cpu   NUMBER,
                  interrupt_cpu   NUMBER
    );
commit;
set echo off;
```

cs_thread.sql

```
rem
rem
rem=============================================================
========+
rem       Copyright (c) 1997  Oracle Corp, Redwood Shores, CA      |
rem                    All Rights Reserved                        |
rem=============================================================
========+
rem FILENAME
rem    cs_thread.sql
rem DESCRIPTION
rem    Create Table for thread statistics
rem=============================================================
========
rem Usage: sqlplus tpcc/tpcc @cs_thread.sql

connect tpcc/tpcc
set echo on

DROP TABLE thread_stats;
DROP TABLE pre_thread_stats;
DROP TABLE post_thread_stats;

rem
rem  Resource usage for a thread.
rem

    CREATE TABLE thread_stats
    (
    runname       VARCHAR2(20),
            thread_id       VARCHAR2(10),
    user_cpu      NUMBER,
    priv_cpu      NUMBER,
    processor_cpu NUMBER,
    ctxswitch     NUMBER
    );



rem
rem  Save Begining Resource Values for a thread.
rem

    CREATE TABLE pre_thread_stats
    (
    runname       VARCHAR2(20),
            thread_id       VARCHAR2(10),
```

```
    user_cpu      NUMBER,
    priv_cpu      NUMBER,
    processor_cpu NUMBER,
    ctxswitch     NUMBER
  );


rem
rem  Save Ending Resource Values for a thread.
rem

  CREATE TABLE post_thread_stats
  (
    runname       VARCHAR2(20),
          thread_id       VARCHAR2(10),
    user_cpu      NUMBER,
    priv_cpu      NUMBER,
    processor_cpu NUMBER,
    ctxswitch     NUMBER
  );
commit;
set echo off
```

cs_proc.sql

```
rem
rem
rem=========================================================
========+
rem       Copyright (c) 1997  Oracle Corp, Redwood Shores, CA      |
rem                    All Rights Reserved                    |
rem=========================================================
========+
rem FILENAME
rem    cs_proc.sql
rem DESCRIPTION
rem    Create Table for OS Specific Process Stats
rem=========================================================
========
rem Usage: sqlplus tpcc/tpcc @cs_proc.sql

connect tpcc/tpcc
set echo on

DROP TABLE process_stats;
DROP TABLE pre_process_stats;
DROP TABLE post_process_stats;

rem
rem  Resource usage for a process.
rem

  CREATE TABLE process_stats
  (
    runname       VARCHAR2(20),
    user_cpu      NUMBER,
    priv_cpu      NUMBER,
    processor_cpu NUMBER,
    pagefaults    NUMBER
  );



rem
rem  Save Begining Resource Values for a process.
rem

  CREATE TABLE pre_process_stats
  (
    runname       VARCHAR2(20),
    user_cpu      NUMBER,
    priv_cpu      NUMBER,
    processor_cpu NUMBER,
    pagefaults    NUMBER
  );


rem
rem  Save Ending Resource Values for a process.
rem

  CREATE TABLE post_process_stats
  (
    runname       VARCHAR2(20),
    user_cpu      NUMBER,
    priv_cpu      NUMBER,
```

```
        processor_cpu NUMBER,
        pagefaults    NUMBER
      );
commit;
set echo off
```

cs_tpcc.sql

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

connect tpcc/tpcc;
set echo on

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

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

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

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

```
        );

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

rem
rem  Aggregate results for all generators.
rem  These results are from the measurement interval only.
rem  These results are used to calculate the TPS rate over
rem  the measurement interval.
rem
        CREATE TABLE tpcc_user_res
        (
           run_name       VARCHAR2(20),
           no_men         NUMBER,
           fast_men       NUMBER,
           in_flight_men  NUMBER,
        retry_men      NUMBER,
           min_time_men   NUMBER,
           max_time_men   NUMBER,
           sum_time_men   NUMBER,
        ninety_per_men NUMBER,
        think_min_men  NUMBER,
        think_max_men  NUMBER,
           think_sum_men  NUMBER,
        key_min_men    NUMBER,
        key_max_men    NUMBER,
           key_sum_men    NUMBER,
           no_new         NUMBER,
           fast_new       NUMBER,
           in_flight_new  NUMBER,
        retry_new      NUMBER,
           min_time_new   NUMBER,
           max_time_new   NUMBER,
           sum_time_new   NUMBER,
        ninety_per_new NUMBER,
        think_min_new  NUMBER,
        think_max_new  NUMBER,
           think_sum_new  NUMBER,
        key_min_new    NUMBER,
        key_max_new    NUMBER,
           key_sum_new    NUMBER,
           remote_new     NUMBER,
           rollback_new   NUMBER,
           sum_ol_new     NUMBER,
           remote_ol_new  NUMBER,
        allrollback_new NUMBER,
           no_pay         NUMBER,
           fast_pay       NUMBER,
           in_flight_pay  NUMBER,
        retry_pay      NUMBER,
           min_time_pay   NUMBER,
           max_time_pay   NUMBER,
           sum_time_pay   NUMBER,
        ninety_per_pay NUMBER,
        think_min_pay  NUMBER,
        think_max_pay  NUMBER,
           think_sum_pay  NUMBER,
        key_min_pay    NUMBER,
        key_max_pay    NUMBER,
           key_sum_pay    NUMBER,
           remote_pay     NUMBER,
           bylast_pay     NUMBER,
           no_ord         NUMBER,
           fast_ord       NUMBER,
           in_flight_ord  NUMBER,
        retry_ord      NUMBER,
           min_time_ord   NUMBER,
           max_time_ord   NUMBER,
           sum_time_ord   NUMBER,
        ninety_per_ord NUMBER,
        think_min_ord  NUMBER,
        think_max_ord  NUMBER,
           think_sum_ord  NUMBER,
        key_min_ord    NUMBER,
        key_max_ord    NUMBER,
           key_sum_ord    NUMBER,
```

```
        bylast_ord      NUMBER,
        no_del          NUMBER,
        fast_del        NUMBER,
        in_flight_del   NUMBER,
    retry_del       NUMBER,
        min_time_del    NUMBER,
        max_time_del    NUMBER,
        sum_time_del    NUMBER,
    ninety_per_del  NUMBER,
    think_min_del   NUMBER,
    think_max_del   NUMBER,
        think_sum_del   NUMBER,
    key_min_del     NUMBER,
    key_max_del     NUMBER,
        key_sum_del     NUMBER,
        no_sto          NUMBER,
        fast_sto        NUMBER,
        in_flight_sto   NUMBER,
    retry_sto       NUMBER,
        min_time_sto    NUMBER,
        max_time_sto    NUMBER,
        sum_time_sto    NUMBER,
    ninety_per_sto  NUMBER,
    think_min_sto   NUMBER,
    think_max_sto   NUMBER,
        think_sum_sto   NUMBER,
    key_min_sto     NUMBER,
    key_max_sto     NUMBER,
        key_sum_sto     NUMBER,
        cpu_time        NUMBER,
    deadlocks       NUMBER
      );

rem
rem  Results from individual generators.
rem  These results are from the measurement interval only.
rem  These results are used to calculate the TPS rate over
rem  the measurement interval.
rem
      CREATE TABLE bench_user_res
      (
        run_name        VARCHAR2(20),
        audit_str       VARCHAR2(10),
        proc_no         NUMBER,
        hid             NUMBER,
        no_men          NUMBER,
        fast_men        NUMBER,
        in_flight_men   NUMBER,
    retry_men       NUMBER,
        min_time_men    NUMBER,
        max_time_men    NUMBER,
        sum_time_men    NUMBER,
    ninety_per_men  NUMBER,
    think_min_men   NUMBER,
    think_max_men   NUMBER,
        think_sum_men   NUMBER,
    key_min_men     NUMBER,
    key_max_men     NUMBER,
        key_sum_men     NUMBER,
        no_new          NUMBER,
        fast_new        NUMBER,
        in_flight_new   NUMBER,
    retry_new       NUMBER,
        min_time_new    NUMBER,
        max_time_new    NUMBER,
        sum_time_new    NUMBER,
    ninety_per_new  NUMBER,
    think_min_new   NUMBER,
    think_max_new   NUMBER,
        think_sum_new   NUMBER,
    key_min_new     NUMBER,
    key_max_new     NUMBER,
        key_sum_new     NUMBER,
        remote_new      NUMBER,
        rollback_new    NUMBER,
        sum_ol_new      NUMBER,
        remote_ol_new   NUMBER,
    allrollback_new NUMBER,
        no_pay          NUMBER,
        fast_pay        NUMBER,
        in_flight_pay   NUMBER,
    retry_pay       NUMBER,
        min_time_pay    NUMBER,
        max_time_pay    NUMBER,
        sum_time_pay    NUMBER,
    ninety_per_pay  NUMBER,
```

```
think_min_pay   NUMBER,
think_max_pay   NUMBER,
    think_sum_pay   NUMBER,
key_min_pay     NUMBER,
key_max_pay     NUMBER,
    key_sum_pay     NUMBER,
    remote_pay      NUMBER,
    bylast_pay      NUMBER,
    no_ord          NUMBER,
    fast_ord        NUMBER,
    in_flight_ord   NUMBER,
retry_ord       NUMBER,
    min_time_ord    NUMBER,
    max_time_ord    NUMBER,
    sum_time_ord    NUMBER,
ninety_per_ord  NUMBER,
think_min_ord   NUMBER,
think_max_ord   NUMBER,
    think_sum_ord   NUMBER,
key_min_ord     NUMBER,
key_max_ord     NUMBER,
    key_sum_ord     NUMBER,
    bylast_ord      NUMBER,
    no_del          NUMBER,
    fast_del        NUMBER,
    in_flight_del   NUMBER,
retry_del       NUMBER,
    min_time_del    NUMBER,
    max_time_del    NUMBER,
    sum_time_del    NUMBER,
ninety_per_del  NUMBER,
think_min_del   NUMBER,
think_max_del   NUMBER,
    think_sum_del   NUMBER,
key_min_del     NUMBER,
key_max_del     NUMBER,
    key_sum_del     NUMBER,
    no_sto          NUMBER,
    fast_sto        NUMBER,
    in_flight_sto   NUMBER,
retry_sto       NUMBER,
    min_time_sto    NUMBER,
    max_time_sto    NUMBER,
    sum_time_sto    NUMBER,
ninety_per_sto  NUMBER,
think_min_sto   NUMBER,
think_max_sto   NUMBER,
    think_sum_sto   NUMBER,
key_min_sto     NUMBER,
key_max_sto     NUMBER,
    key_sum_sto     NUMBER,
    cpu_time        NUMBER,
deadlocks       NUMBER
  );

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

rem
rem  Aggregate results for new order transactions.
rem  These results are from the measurement interval only.
rem
        CREATE TABLE tpcc_new_res
        (
          run_name       VARCHAR2(20),
          rep1           NUMBER,
          rep2           NUMBER,
          rep3           NUMBER,
          rep4           NUMBER,
          rep5           NUMBER,
          rep6           NUMBER,
          rep7           NUMBER,
          rep8           NUMBER,
          rep9           NUMBER,
          rep10          NUMBER,
          rep11          NUMBER,
```

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

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

rem
rem  Results for new order transactions.
rem  These results are from the measurement interval only.
rem
CREATE TABLE bench_new_res
(
    run_name    VARCHAR2(20),
    audit_str   VARCHAR2(10),
    proc_no     NUMBER,
    rep1        NUMBER,
    rep2        NUMBER,
    rep3        NUMBER,
    rep4        NUMBER,
    rep5        NUMBER,
    rep6        NUMBER,
    rep7        NUMBER,
    rep8        NUMBER,
    rep9        NUMBER,
    rep10       NUMBER,
    rep11       NUMBER,
    rep12       NUMBER,
    rep13       NUMBER,
    rep14       NUMBER,
    rep15       NUMBER,
    rep16       NUMBER,
    rep17       NUMBER,
    rep18       NUMBER,
    rep19       NUMBER,
    rep20       NUMBER,
    rep21       NUMBER,
    rep22       NUMBER,
    rep23       NUMBER,
    rep24       NUMBER,
    rep25       NUMBER,
    rep26       NUMBER,
    rep27       NUMBER,
    rep28       NUMBER,
    rep29       NUMBER,
    rep30       NUMBER,
    rep31       NUMBER,
    rep32       NUMBER,
    rep33       NUMBER,
    rep34       NUMBER,
    rep35       NUMBER,
    rep36       NUMBER,
    rep37       NUMBER,
```

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

```
    thk24       NUMBER,
    thk25       NUMBER,
    key1        NUMBER,
    key2        NUMBER,
    key3        NUMBER,
    key4        NUMBER,
    key5        NUMBER,
    key6        NUMBER,
    key7        NUMBER,
    key8        NUMBER,
    key9        NUMBER,
    key10       NUMBER
);

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

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

rem
rem Results for payment transactions.
rem These results are from the measurement interval only.
rem
CREATE TABLE bench_pay_res
(
    run_name    VARCHAR2(20),
    audit_str   VARCHAR2(10),
    proc_no     NUMBER,
    rep1        NUMBER,
    rep2        NUMBER,
    rep3        NUMBER,
    rep4        NUMBER,
    rep5        NUMBER,
```

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

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

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

```
rep34       NUMBER,
rep35       NUMBER,
rep36       NUMBER,
rep37       NUMBER,
rep38       NUMBER,
rep39       NUMBER,
rep40       NUMBER,
rep41       NUMBER,
rep42       NUMBER,
rep43       NUMBER,
rep44       NUMBER,
rep45       NUMBER,
rep46       NUMBER,
rep47       NUMBER,
rep48       NUMBER,
rep49       NUMBER,
rep50       NUMBER,
rep51       NUMBER,
rep52       NUMBER,
rep53       NUMBER,
rep54       NUMBER,
rep55       NUMBER,
rep56       NUMBER,
rep57       NUMBER,
rep58       NUMBER,
rep59       NUMBER,
rep60       NUMBER,
rep61       NUMBER,
rep62       NUMBER,
rep63       NUMBER,
rep64       NUMBER,
rep65       NUMBER,
rep66       NUMBER,
rep67       NUMBER,
rep68       NUMBER,
rep69       NUMBER,
rep70       NUMBER,
rep71       NUMBER,
rep72       NUMBER,
rep73       NUMBER,
rep74       NUMBER,
rep75       NUMBER,
rep76       NUMBER,
rep77       NUMBER,
rep78       NUMBER,
rep79       NUMBER,
rep80       NUMBER,
rep81       NUMBER,
rep82       NUMBER,
rep83       NUMBER,
rep84       NUMBER,
rep85       NUMBER,
rep86       NUMBER,
rep87       NUMBER,
rep88       NUMBER,
rep89       NUMBER,
rep90       NUMBER,
rep91       NUMBER,
rep92       NUMBER,
rep93       NUMBER,
rep94       NUMBER,
rep95       NUMBER,
rep96       NUMBER,
rep97       NUMBER,
rep98       NUMBER,
rep99       NUMBER,
rep100      NUMBER,
thk1        NUMBER,
thk2        NUMBER,
thk3        NUMBER,
thk4        NUMBER,
thk5        NUMBER,
thk6        NUMBER,
thk7        NUMBER,
thk8        NUMBER,
thk9        NUMBER,
thk10       NUMBER,
thk11       NUMBER,
thk12       NUMBER,
thk13       NUMBER,
thk14       NUMBER,
thk15       NUMBER,
thk16       NUMBER,
thk17       NUMBER,
thk18       NUMBER,
thk19       NUMBER,
```

```
    thk20       NUMBER,
    thk21       NUMBER,
    thk22       NUMBER,
    thk23       NUMBER,
    thk24       NUMBER,
    thk25       NUMBER,
    key1        NUMBER,
    key2        NUMBER,
    key3        NUMBER,
    key4        NUMBER,
    key5        NUMBER,
    key6        NUMBER,
    key7        NUMBER,
    key8        NUMBER,
    key9        NUMBER,
    key10       NUMBER
  );

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

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

rem
rem Aggregate results for delivery transactions.
rem These results are from the measurement interval only.
rem
CREATE TABLE tpcc_del_res
(
    run_name    VARCHAR2(20),
    rep1        NUMBER,
```

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

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

rem
rem  Results for delivery transactions.
rem  These results are from the measurement interval only.
rem
    CREATE TABLE bench_del_res
    (
    run_name    VARCHAR2(20),
    audit_str   VARCHAR2(10),
    proc_no     NUMBER,
    rep1        NUMBER,
    rep2        NUMBER,
    rep3        NUMBER,
    rep4        NUMBER,
    rep5        NUMBER,
    rep6        NUMBER,
    rep7        NUMBER,
    rep8        NUMBER,
    rep9        NUMBER,
    rep10       NUMBER,
    rep11       NUMBER,
    rep12       NUMBER,
    rep13       NUMBER,
    rep14       NUMBER,
    rep15       NUMBER,
    rep16       NUMBER,
    rep17       NUMBER,
    rep18       NUMBER,
    rep19       NUMBER,
    rep20       NUMBER,
    rep21       NUMBER,
    rep22       NUMBER,
    rep23       NUMBER,
    rep24       NUMBER,
    rep25       NUMBER,
    rep26       NUMBER,
    rep27       NUMBER,
```

```
rep28       NUMBER,
rep29       NUMBER,
rep30       NUMBER,
rep31       NUMBER,
rep32       NUMBER,
rep33       NUMBER,
rep34       NUMBER,
rep35       NUMBER,
rep36       NUMBER,
rep37       NUMBER,
rep38       NUMBER,
rep39       NUMBER,
rep40       NUMBER,
rep41       NUMBER,
rep42       NUMBER,
rep43       NUMBER,
rep44       NUMBER,
rep45       NUMBER,
rep46       NUMBER,
rep47       NUMBER,
rep48       NUMBER,
rep49       NUMBER,
rep50       NUMBER,
rep51       NUMBER,
rep52       NUMBER,
rep53       NUMBER,
rep54       NUMBER,
rep55       NUMBER,
rep56       NUMBER,
rep57       NUMBER,
rep58       NUMBER,
rep59       NUMBER,
rep60       NUMBER,
rep61       NUMBER,
rep62       NUMBER,
rep63       NUMBER,
rep64       NUMBER,
rep65       NUMBER,
rep66       NUMBER,
rep67       NUMBER,
rep68       NUMBER,
rep69       NUMBER,
rep70       NUMBER,
rep71       NUMBER,
rep72       NUMBER,
rep73       NUMBER,
rep74       NUMBER,
rep75       NUMBER,
rep76       NUMBER,
rep77       NUMBER,
rep78       NUMBER,
rep79       NUMBER,
rep80       NUMBER,
rep81       NUMBER,
rep82       NUMBER,
rep83       NUMBER,
rep84       NUMBER,
rep85       NUMBER,
rep86       NUMBER,
rep87       NUMBER,
rep88       NUMBER,
rep89       NUMBER,
rep90       NUMBER,
rep91       NUMBER,
rep92       NUMBER,
rep93       NUMBER,
rep94       NUMBER,
rep95       NUMBER,
rep96       NUMBER,
rep97       NUMBER,
rep98       NUMBER,
rep99       NUMBER,
rep100      NUMBER,
thk1        NUMBER,
thk2        NUMBER,
thk3        NUMBER,
thk4        NUMBER,
thk5        NUMBER,
thk6        NUMBER,
thk7        NUMBER,
thk8        NUMBER,
thk9        NUMBER,
thk10       NUMBER,
thk11       NUMBER,
thk12       NUMBER,
thk13       NUMBER,
```

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

rem
rem  Aggregate results for stock level transactions.
rem  These results are from the measurement interval only.
rem
  CREATE TABLE tpcc_sto_res
  (
    run_name    VARCHAR2(20),
    rep1        NUMBER,
    rep2        NUMBER,
    rep3        NUMBER,
    rep4        NUMBER,
    rep5        NUMBER,
    rep6        NUMBER,
    rep7        NUMBER,
    rep8        NUMBER,
    rep9        NUMBER,
    rep10       NUMBER,
    rep11       NUMBER,
    rep12       NUMBER,
    rep13       NUMBER,
    rep14       NUMBER,
    rep15       NUMBER,
    rep16       NUMBER,
    rep17       NUMBER,
    rep18       NUMBER,
    rep19       NUMBER,
    rep20       NUMBER,
    rep21       NUMBER,
    rep22       NUMBER,
    rep23       NUMBER,
    rep24       NUMBER,
    rep25       NUMBER,
    rep26       NUMBER,
    rep27       NUMBER,
    rep28       NUMBER,
    rep29       NUMBER,
    rep30       NUMBER,
    rep31       NUMBER,
    rep32       NUMBER,
    rep33       NUMBER,
    rep34       NUMBER,
    rep35       NUMBER,
    rep36       NUMBER,
    rep37       NUMBER,
    rep38       NUMBER,
    rep39       NUMBER,
    rep40       NUMBER,
    rep41       NUMBER,
    rep42       NUMBER,
    rep43       NUMBER,
    rep44       NUMBER,
    rep45       NUMBER,
    rep46       NUMBER,
    rep47       NUMBER,
    rep48       NUMBER,
    rep49       NUMBER,
    rep50       NUMBER,
    rep51       NUMBER,
    rep52       NUMBER,
    rep53       NUMBER,
    rep54       NUMBER,
    rep55       NUMBER,
```

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

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

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

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

bcexpr.sh

```
#!/bin/sh
# send command line to bc
echo "$*" | bc
```

runscript.sh

```
#!/bin/sh
# run a script from the script directory.
# go to log directory so we don't leave a mess in the bench dir.
cd $tpcc_bench/log

if test $tpcc_np -gt 1 ; then
  if test "$1" = "createts" ; then
    ptr=$tpcc_rac_id
    count=0
    total=$tpcc_rac_createts_count
    if test "x$tpcc_rac_createts_count" = "x" ; then
      total=$tpcc_np
    fi
    if test $total -gt $tpcc_np ; then
      total=$tpcc_np
    fi
    if expr x$tpcc_listfiles = xt > /dev/null; then
```

```
    ptr=1
    total=$tpcc_np
  fi
  while test $count -lt $total ; do
    /bin/sh $tpcc_genscripts_dir/createts_node${ptr}.sh
    ptr=`expr $ptr + 1`
    if test $ptr -gt $tpcc_np ; then
      ptr=1
    fi
    count=`expr $count + 1`
  done
 else
  if test "$1" = "loadordrordl" ; then
    /bin/sh $tpcc_genscripts_dir/loadordrordl_node${tpcc_rac_id}.sh
  else
    if test "$1" = "loadnord" ; then
      /bin/sh $tpcc_genscripts_dir/loadnord_node${tpcc_rac_id}.sh
    else
      /bin/sh $tpcc_genscripts_dir/${1}.sh
    fi
  fi
 fi
else
  /bin/sh $tpcc_genscripts_dir/${1}.sh
fi
exit $?
```

c_stats.sql

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

connect tpcc/tpcc;
set echo on

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

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

rem
rem  throughput of new order transactions
```

```
rem
        CREATE TABLE tpcc_run_int
        (
           run_name       VARCHAR2(20),
           interval       NUMBER,
           interval_count  NUMBER,
           response_time   NUMBER,
           think_time      NUMBER
        );

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

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

rem
rem  Aggregate results for all generators.
rem  These results are from the measurement interval only.
rem  These results are used to calculate the TPS rate over
rem  the measurement interval.
rem
        CREATE TABLE tpcc_user_res
        (
           run_name       VARCHAR2(20),
           no_men         NUMBER,
           fast_men       NUMBER,
           in_flight_men   NUMBER,
retry_men       NUMBER,
           min_time_men    NUMBER,
           max_time_men    NUMBER,
           sum_time_men    NUMBER,
ninety_per_men  NUMBER,
think_min_men   NUMBER,
think_max_men   NUMBER,
           think_sum_men   NUMBER,
key_min_men     NUMBER,
key_max_men     NUMBER,
           key_sum_men     NUMBER,
           no_new          NUMBER,
           fast_new        NUMBER,
           in_flight_new   NUMBER,
retry_new       NUMBER,
           min_time_new    NUMBER,
           max_time_new    NUMBER,
           sum_time_new    NUMBER,
ninety_per_new  NUMBER,
think_min_new   NUMBER,
think_max_new   NUMBER,
           think_sum_new   NUMBER,
key_min_new     NUMBER,
key_max_new     NUMBER,
           key_sum_new     NUMBER,
           remote_new      NUMBER,
           rollback_new    NUMBER,
           sum_ol_new      NUMBER,
           remote_ol_new   NUMBER,
allrollback_new NUMBER,
           no_pay          NUMBER,
           fast_pay        NUMBER,
           in_flight_pay   NUMBER,
retry_pay       NUMBER,
           min_time_pay    NUMBER,
           max_time_pay    NUMBER,
           sum_time_pay    NUMBER,
ninety_per_pay  NUMBER,
think_min_pay   NUMBER,
```

```
think_max_pay   NUMBER,
    think_sum_pay   NUMBER,
key_min_pay     NUMBER,
key_max_pay     NUMBER,
    key_sum_pay     NUMBER,
    remote_pay      NUMBER,
    bylast_pay      NUMBER,
    no_ord          NUMBER,
    fast_ord        NUMBER,
    in_flight_ord   NUMBER,
retry_ord       NUMBER,
    min_time_ord    NUMBER,
    max_time_ord    NUMBER,
    sum_time_ord    NUMBER,
ninety_per_ord  NUMBER,
think_min_ord   NUMBER,
think_max_ord   NUMBER,
    think_sum_ord   NUMBER,
key_min_ord     NUMBER,
key_max_ord     NUMBER,
    key_sum_ord     NUMBER,
    bylast_ord      NUMBER,
    no_del          NUMBER,
    fast_del        NUMBER,
    in_flight_del   NUMBER,
retry_del       NUMBER,
    min_time_del    NUMBER,
    max_time_del    NUMBER,
    sum_time_del    NUMBER,
ninety_per_del  NUMBER,
think_min_del   NUMBER,
think_max_del   NUMBER,
    think_sum_del   NUMBER,
key_min_del     NUMBER,
key_max_del     NUMBER,
    key_sum_del     NUMBER,
    no_sto          NUMBER,
    fast_sto        NUMBER,
    in_flight_sto   NUMBER,
retry_sto       NUMBER,
    min_time_sto    NUMBER,
    max_time_sto    NUMBER,
    sum_time_sto    NUMBER,
ninety_per_sto  NUMBER,
think_min_sto   NUMBER,
think_max_sto   NUMBER,
    think_sum_sto   NUMBER,
key_min_sto     NUMBER,
key_max_sto     NUMBER,
    key_sum_sto     NUMBER,
    cpu_time        NUMBER,
deadlocks       NUMBER
 );

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

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

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

```
    no_new          NUMBER
  );

rem
rem  Aggregate results for new order transactions.
rem  These results are from the measurement interval only.
rem
  CREATE TABLE tpcc_new_res
  (
    run_name        VARCHAR2(20),
    rep1            NUMBER,
    rep2            NUMBER,
    rep3            NUMBER,
    rep4            NUMBER,
    rep5            NUMBER,
    rep6            NUMBER,
    rep7            NUMBER,
    rep8            NUMBER,
    rep9            NUMBER,
    rep10           NUMBER,
    rep11           NUMBER,
    rep12           NUMBER,
    rep13           NUMBER,
    rep14           NUMBER,
    rep15           NUMBER,
    rep16           NUMBER,
    rep17           NUMBER,
    rep18           NUMBER,
    rep19           NUMBER,
    rep20           NUMBER,
    rep21           NUMBER,
    rep22           NUMBER,
    rep23           NUMBER,
    rep24           NUMBER,
    rep25           NUMBER,
    rep26           NUMBER,
    rep27           NUMBER,
    rep28           NUMBER,
    rep29           NUMBER,
    rep30           NUMBER,
    rep31           NUMBER,
    rep32           NUMBER,
    rep33           NUMBER,
    rep34           NUMBER,
    rep35           NUMBER,
    rep36           NUMBER,
    rep37           NUMBER,
    rep38           NUMBER,
    rep39           NUMBER,
    rep40           NUMBER,
    rep41           NUMBER,
    rep42           NUMBER,
    rep43           NUMBER,
    rep44           NUMBER,
    rep45           NUMBER,
    rep46           NUMBER,
    rep47           NUMBER,
    rep48           NUMBER,
    rep49           NUMBER,
    rep50           NUMBER,
    rep51           NUMBER,
    rep52           NUMBER,
    rep53           NUMBER,
    rep54           NUMBER,
    rep55           NUMBER,
    rep56           NUMBER,
    rep57           NUMBER,
    rep58           NUMBER,
    rep59           NUMBER,
    rep60           NUMBER,
    rep61           NUMBER,
    rep62           NUMBER,
    rep63           NUMBER,
    rep64           NUMBER,
    rep65           NUMBER,
    rep66           NUMBER,
    rep67           NUMBER,
    rep68           NUMBER,
    rep69           NUMBER,
    rep70           NUMBER,
    rep71           NUMBER,
    rep72           NUMBER,
    rep73           NUMBER,
    rep74           NUMBER,
    rep75           NUMBER,
    rep76           NUMBER,
```

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

rem
rem  Results for new order transactions.
rem  These results are from the measurement interval only.
rem
  CREATE TABLE bench_new_res
  (
    run_name      VARCHAR2(20),
    audit_str     VARCHAR2(10),
    proc_no       NUMBER,
    rep1          NUMBER,
    rep2          NUMBER,
    rep3          NUMBER,
    rep4          NUMBER,
    rep5          NUMBER,
    rep6          NUMBER,
    rep7          NUMBER,
    rep8          NUMBER,
    rep9          NUMBER,
    rep10         NUMBER,
    rep11         NUMBER,
    rep12         NUMBER,
    rep13         NUMBER,
    rep14         NUMBER,
    rep15         NUMBER,
    rep16         NUMBER,
```

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

```
    thk3        NUMBER,
    thk4        NUMBER,
    thk5        NUMBER,
    thk6        NUMBER,
    thk7        NUMBER,
    thk8        NUMBER,
    thk9        NUMBER,
    thk10       NUMBER,
    thk11       NUMBER,
    thk12       NUMBER,
    thk13       NUMBER,
    thk14       NUMBER,
    thk15       NUMBER,
    thk16       NUMBER,
    thk17       NUMBER,
    thk18       NUMBER,
    thk19       NUMBER,
    thk20       NUMBER,
    thk21       NUMBER,
    thk22       NUMBER,
    thk23       NUMBER,
    thk24       NUMBER,
    thk25       NUMBER,
    key1        NUMBER,
    key2        NUMBER,
    key3        NUMBER,
    key4        NUMBER,
    key5        NUMBER,
    key6        NUMBER,
    key7        NUMBER,
    key8        NUMBER,
    key9        NUMBER,
    key10       NUMBER
  );

rem
rem  Aggregate results for payment transactions.
rem  These results are from the measurement interval only.
rem
  CREATE TABLE tpcc_pay_res
  (
    run_name    VARCHAR2(20),
    rep1        NUMBER,
    rep2        NUMBER,
    rep3        NUMBER,
    rep4        NUMBER,
    rep5        NUMBER,
    rep6        NUMBER,
    rep7        NUMBER,
    rep8        NUMBER,
    rep9        NUMBER,
    rep10       NUMBER,
    rep11       NUMBER,
    rep12       NUMBER,
    rep13       NUMBER,
    rep14       NUMBER,
    rep15       NUMBER,
    rep16       NUMBER,
    rep17       NUMBER,
    rep18       NUMBER,
    rep19       NUMBER,
    rep20       NUMBER,
    rep21       NUMBER,
    rep22       NUMBER,
    rep23       NUMBER,
    rep24       NUMBER,
    rep25       NUMBER,
    rep26       NUMBER,
    rep27       NUMBER,
    rep28       NUMBER,
    rep29       NUMBER,
    rep30       NUMBER,
    rep31       NUMBER,
    rep32       NUMBER,
    rep33       NUMBER,
    rep34       NUMBER,
    rep35       NUMBER,
    rep36       NUMBER,
    rep37       NUMBER,
    rep38       NUMBER,
    rep39       NUMBER,
    rep40       NUMBER,
    rep41       NUMBER,
    rep42       NUMBER,
    rep43       NUMBER,
    rep44       NUMBER,
```

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

```
    key6        NUMBER,
    key7        NUMBER,
    key8        NUMBER,
    key9        NUMBER,
    key10       NUMBER
  );

rem
rem  Results for payment transactions.
rem  These results are from the measurement interval only.
rem
  CREATE TABLE bench_pay_res
  (
    run_name      VARCHAR2(20),
    audit_str     VARCHAR2(10),
    proc_no       NUMBER,
    rep1          NUMBER,
    rep2          NUMBER,
    rep3          NUMBER,
    rep4          NUMBER,
    rep5          NUMBER,
    rep6          NUMBER,
    rep7          NUMBER,
    rep8          NUMBER,
    rep9          NUMBER,
    rep10         NUMBER,
    rep11         NUMBER,
    rep12         NUMBER,
    rep13         NUMBER,
    rep14         NUMBER,
    rep15         NUMBER,
    rep16         NUMBER,
    rep17         NUMBER,
    rep18         NUMBER,
    rep19         NUMBER,
    rep20         NUMBER,
    rep21         NUMBER,
    rep22         NUMBER,
    rep23         NUMBER,
    rep24         NUMBER,
    rep25         NUMBER,
    rep26         NUMBER,
    rep27         NUMBER,
    rep28         NUMBER,
    rep29         NUMBER,
    rep30         NUMBER,
    rep31         NUMBER,
    rep32         NUMBER,
    rep33         NUMBER,
    rep34         NUMBER,
    rep35         NUMBER,
    rep36         NUMBER,
    rep37         NUMBER,
    rep38         NUMBER,
    rep39         NUMBER,
    rep40         NUMBER,
    rep41         NUMBER,
    rep42         NUMBER,
    rep43         NUMBER,
    rep44         NUMBER,
    rep45         NUMBER,
    rep46         NUMBER,
    rep47         NUMBER,
    rep48         NUMBER,
    rep49         NUMBER,
    rep50         NUMBER,
    rep51         NUMBER,
    rep52         NUMBER,
    rep53         NUMBER,
    rep54         NUMBER,
    rep55         NUMBER,
    rep56         NUMBER,
    rep57         NUMBER,
    rep58         NUMBER,
    rep59         NUMBER,
    rep60         NUMBER,
    rep61         NUMBER,
    rep62         NUMBER,
    rep63         NUMBER,
    rep64         NUMBER,
    rep65         NUMBER,
    rep66         NUMBER,
    rep67         NUMBER,
    rep68         NUMBER,
    rep69         NUMBER,
    rep70         NUMBER,
```

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

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

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

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

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

CREATE TABLE bench_ord_res
(
  run_name      VARCHAR2(20),
  audit_str     VARCHAR2(10),
  proc_no       NUMBER,
  rep1          NUMBER,
  rep2          NUMBER,
  rep3          NUMBER,
  rep4          NUMBER,
  rep5          NUMBER,
  rep6          NUMBER,
  rep7          NUMBER,
  rep8          NUMBER,
  rep9          NUMBER,
  rep10         NUMBER,
  rep11         NUMBER,
  rep12         NUMBER,
  rep13         NUMBER,
  rep14         NUMBER,
  rep15         NUMBER,
  rep16         NUMBER,
  rep17         NUMBER,
  rep18         NUMBER,
  rep19         NUMBER,
  rep20         NUMBER,
  rep21         NUMBER,
  rep22         NUMBER,
  rep23         NUMBER,
  rep24         NUMBER,
  rep25         NUMBER,
  rep26         NUMBER,
  rep27         NUMBER,
  rep28         NUMBER,
  rep29         NUMBER,
  rep30         NUMBER,
  rep31         NUMBER,
  rep32         NUMBER,
  rep33         NUMBER,
  rep34         NUMBER,
  rep35         NUMBER,
  rep36         NUMBER,
  rep37         NUMBER,
  rep38         NUMBER,
```

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

```
    thk25       NUMBER,
    key1        NUMBER,
    key2        NUMBER,
    key3        NUMBER,
    key4        NUMBER,
    key5        NUMBER,
    key6        NUMBER,
    key7        NUMBER,
    key8        NUMBER,
    key9        NUMBER,
    key10       NUMBER
  );

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

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

rem
rem  Results for delivery transactions.
rem  These results are from the measurement interval only.
rem
    CREATE TABLE bench_del_res
    (
      run_name      VARCHAR2(20),
      audit_str     VARCHAR2(10),
      proc_no       NUMBER,
      rep1          NUMBER,
      rep2          NUMBER,
      rep3          NUMBER,
      rep4          NUMBER,
      rep5          NUMBER,
      rep6          NUMBER,
```

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

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

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

```
rep35       NUMBER,
rep36       NUMBER,
rep37       NUMBER,
rep38       NUMBER,
rep39       NUMBER,
rep40       NUMBER,
rep41       NUMBER,
rep42       NUMBER,
rep43       NUMBER,
rep44       NUMBER,
rep45       NUMBER,
rep46       NUMBER,
rep47       NUMBER,
rep48       NUMBER,
rep49       NUMBER,
rep50       NUMBER,
rep51       NUMBER,
rep52       NUMBER,
rep53       NUMBER,
rep54       NUMBER,
rep55       NUMBER,
rep56       NUMBER,
rep57       NUMBER,
rep58       NUMBER,
rep59       NUMBER,
rep60       NUMBER,
rep61       NUMBER,
rep62       NUMBER,
rep63       NUMBER,
rep64       NUMBER,
rep65       NUMBER,
rep66       NUMBER,
rep67       NUMBER,
rep68       NUMBER,
rep69       NUMBER,
rep70       NUMBER,
rep71       NUMBER,
rep72       NUMBER,
rep73       NUMBER,
rep74       NUMBER,
rep75       NUMBER,
rep76       NUMBER,
rep77       NUMBER,
rep78       NUMBER,
rep79       NUMBER,
rep80       NUMBER,
rep81       NUMBER,
rep82       NUMBER,
rep83       NUMBER,
rep84       NUMBER,
rep85       NUMBER,
rep86       NUMBER,
rep87       NUMBER,
rep88       NUMBER,
rep89       NUMBER,
rep90       NUMBER,
rep91       NUMBER,
rep92       NUMBER,
rep93       NUMBER,
rep94       NUMBER,
rep95       NUMBER,
rep96       NUMBER,
rep97       NUMBER,
rep98       NUMBER,
rep99       NUMBER,
rep100      NUMBER,
thk1        NUMBER,
thk2        NUMBER,
thk3        NUMBER,
thk4        NUMBER,
thk5        NUMBER,
thk6        NUMBER,
thk7        NUMBER,
thk8        NUMBER,
thk9        NUMBER,
thk10       NUMBER,
thk11       NUMBER,
thk12       NUMBER,
thk13       NUMBER,
thk14       NUMBER,
thk15       NUMBER,
thk16       NUMBER,
thk17       NUMBER,
thk18       NUMBER,
thk19       NUMBER,
thk20       NUMBER,
```

```
    thk21       NUMBER,
    thk22       NUMBER,
    thk23       NUMBER,
    thk24       NUMBER,
    thk25       NUMBER,
    key1        NUMBER,
    key2        NUMBER,
    key3        NUMBER,
    key4        NUMBER,
    key5        NUMBER,
    key6        NUMBER,
    key7        NUMBER,
    key8        NUMBER,
    key9        NUMBER,
    key10       NUMBER
  );

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

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

analyze.sql

```
spool analyze.log;
set echo on;
```

```
connect tpcc/tpcc;

ANALYZE TABLE stok ESTIMATE STATISTICS;
ANALYZE TABLE cust ESTIMATE STATISTICS;
ANALYZE TABLE ordr ESTIMATE STATISTICS;
ANALYZE TABLE ordl ESTIMATE STATISTICS;
ANALYZE TABLE hist ESTIMATE STATISTICS;
ANALYZE TABLE dist ESTIMATE STATISTICS;
ANALYZE TABLE item ESTIMATE STATISTICS;
ANALYZE TABLE ware ESTIMATE STATISTICS;
ANALYZE TABLE nord ESTIMATE STATISTICS;
ANALYZE index iware ESTIMATE STATISTICS;
ANALYZE index idist ESTIMATE STATISTICS;
ANALYZE index iitem ESTIMATE STATISTICS;
ANALYZE index icust1 ESTIMATE STATISTICS;
ANALYZE index icust2 ESTIMATE STATISTICS;
ANALYZE index istok ESTIMATE STATISTICS;
ANALYZE index iordr1 ESTIMATE STATISTICS;
ANALYZE index iordr2 ESTIMATE STATISTICS;

set echo off;
spool off;

exit sql.sqlcode;
```

runsql.sh

```
#!/bin/sh
# run a sql script from the script directory.
# go to log directory so we don't leave a mess in the bench dir.
cd $tpcc_bench/log
$tpcc_sqlplus $tpcc_user_pass @$tpcc_genscripts_dir/${1}
exit $?
```

runsqllocal.sh

```
#!/bin/sh
# run a sql script from the script directory.
$tpcc_sqlplus '/ as sysdba' @$tpcc_genscripts_dir/${1}
exit $?
```

B2. Loader Source Code

dpbcpu.c

```
/* Copyright (c) Oracle Corporation 1993.  All Rights Reserved. */

/*
NAME    DPBTIME.C

DESCRIPTION
  Get time in seconds.

NOTES
  Desktop Performance Group

MODIFIED      (MM/DD/YY)
  bmoriart     05/10/95 -  V4.7 Convert from double to clock_t
  MBHULLAR     02/06/95 -  V4.5
*/

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


clock_t dpbcpu(void)
{
  clock_t begin_cpu;

  begin_cpu = clock();
  return(begin_cpu);
}
#endif /* ORA_NT */
```

dpbetime.c

```
/* Copyright (c) Oracle Corporation 1995.  All Rights Reserved. */
/*
NAME    DPBETIME.C

DESCRIPTION
```

```
  Get elapsed time in 10ths of millaseconds as a clock_t.

NOTES
  Desktop Performance Group

MODIFIED      (MM/DD/YY)
  B Moriarty   06/02/95 -  V4.8 Initial Version

*/

#ifdef ORA_OS2
#endif /* ORA_OS2 */

#ifdef ORA_NT
#include <windows.h>
#include <sys\types.h>
#include <time.h>
#include <stdio.h>

BOOL First = TRUE;
LARGE_INTEGER ICount;          /* Initial Time */
LARGE_INTEGER Tptms;                    /* Ticks per tenth of millisecond */
#endif /* ORA_NT */

# ifdef __STDC__
clock_t dpbetime(void)
# else
clock_t dpbetime()
# endif /* __STDC__ */
{

#ifdef ORA_NT

  LARGE_INTEGER PFreq;                  /* Ticks per Second */
  LARGE_INTEGER PCount;       /* Ticks Since 1970 */
  clock_t etime;              /* Elapsed time in tenths of millaseconds */

  if (First)  {
    if (!QueryPerformanceFrequency(&PFreq))
      return((clock_t)-1);
    if (!QueryPerformanceCounter(&ICount))
      return((clock_t)-1);
    Tptms.QuadPart = PFreq.QuadPart / 10000;
    First = FALSE;
    return((clock_t)0);
  }
  if (!QueryPerformanceCounter(&PCount))
    return((clock_t)-1);
  etime = (clock_t) ((PCount.QuadPart - ICount.QuadPart) / Tptms.QuadPart);
  return(etime);

#endif /* ORA_NT */

}
```

dpbfsync.c

```
/* Copyright (c) Oracle Corporation 1993, 1992.  All Rights Reserved. */

/*
NAME    DPBFSYNC.C

DESCRIPTION

  Flush o/s buffers to disk for a file.

  Calling fclose() or fflush() is not enough.  These calls will only flush
  the buffer in the FILE struture by making a write() call to the o/s, and
  the o/s will probably place these data in its own disk buffers.
  dpbfsync() will cause the o/s disk buffers for a file to be written to disk.

  This function should normally be called *after* an fflush() is done, or you
  will miss the data that is buffered in the FILE structure.

NOTES
  Desktop Performance Group

MODIFIED      (MM/DD/YY)
  C Kelly     02/24/94 -  V4.4 initial version
*/


#include <stdio.h>
```

```
#ifdef ORA_OS2
int dpbfsync(FILE *fp)
{
  return 0;
}
#endif /* ORA_OS2 */



#ifdef ORA_NT
# include <windows.h>

int dpbfsync(FILE *fp)
{
  if (FlushFileBuffers((HANDLE)(fp->_file)) == FALSE)
  {
    return 1;
  };

  return 0;
}
#endif /* ORA_NT */



#ifdef ORA_AUX

int dpbfsync(fp)
FILE *fp;
{
  if (fsync(fp->_file) == -1)
  {
    return 1;
  };

  return 0;
}
#endif /* ORA_AUX */



#ifdef ORA_NW
int dpbfsync(FILE *fp)
{
  return 0;
}
#endif /* ORA_NW */



#ifdef ORA_DOS
int dpbfsync(FILE *fp)
{
  return 0;
}
#endif /* ORA_DOS */



#ifdef ORA_MAC
#endif /* ORA_MAC */
```

dpbinpgm.c

```
/* Copyright (c) Oracle Corporation 1994.  All Rights Reserved. */

/*
NAME    DPBINPGM.C

DESCRIPTION
  Routine that performs any o/s specific program initialization.

NOTES
  Desktop Performance Group

MODIFIED      (MM/DD/YY)
  C Kelly     04/21/94 -  V4.4 created to support Netware NLMs

*/



#ifdef ORA_NW
#include <process.h>
#include <library.h>
```

```
extern int samtid;
extern int samtgid;

#else /* ORA_NW */
#endif /* ORA_NW */



#ifdef __STDC__
void dpbinpgm(void)
#else
void dpbinpgm()
#endif /* __STDC__ */
{
# ifdef ORA_NW

  samtid  = GetThreadID();      /* get this program's thread id         */
  samtgid = GetThreadGroupID(); /* get this program's thread group id   */

# else /* ORA_NW */

  return;               /* do nothing for everything else */

# endif /* ORA_NW */
}
```

dpboradt.c

```
/* Copyright (c) Oracle Corporation 1993.  All Rights Reserved. */

/*
NAME    DPBORADT.C

DESCRIPTION
   Get System Date and Time and
    Return in Oracle External SQLT_DAT (Date) Format
    Returns 1-JAN-2000 00:00:00
    when not implemented or when conversion fails

NOTES
   Desktop Performance Group

MODIFIED      (MM/DD/YY)
   bmoriart     05/26/95 -  V4.8 Created
*/

#ifdef ORA_NT
# include <windows.h>
#endif /* ORA_NT */

#ifdef __STDC__
void dpboradt(char *oradt)
#else
void dpboradt(oradt)
unsigned char *oradt;
#endif /* __STDC__ */
{
   char cnvrtOK=TRUE;

# ifdef ORA_NT
   SYSTEMTIME lpst;

   GetLocalTime(&lpst);
   *oradt = (unsigned char)(lpst.wYear / 100) + 100;
   if (*oradt < 119 || *oradt > 120) cnvrtOK=FALSE;
   *(++oradt) = (unsigned char)(lpst.wYear % 100) + 100;
   if (*oradt < 100 || *oradt > 199) cnvrtOK=FALSE;
   *(++oradt) = (unsigned char)(lpst.wMonth);
   if (*oradt < 1 || *oradt > 12) cnvrtOK=FALSE;
   *(++oradt) = (unsigned char)(lpst.wDay);
   if (*oradt < 1 || *oradt > 31) cnvrtOK=FALSE;
   *(++oradt) = (unsigned char)(lpst.wHour) + 1;
   if (*oradt < 1 || *oradt > 24) cnvrtOK=FALSE;
   *(++oradt) = (unsigned char)(lpst.wMinute) + 1;
   if (*oradt < 1 || *oradt > 60) cnvrtOK=FALSE;
   *(++oradt) = (unsigned char)(lpst.wSecond) + 1;
   if (*oradt < 1 || *oradt > 60) cnvrtOK=FALSE;
#else /* ORA_NT */
   cnvrtOK = FALSE;
#endif /* ORA_NT */

   if(!cnvrtOK) { /* Use 1-JAN-2000 00:00:00 */
     *oradt++ = 120;
     *oradt++ = 100;
     *oradt++ = 1;
     *oradt++ = 1;
```

```
    *oradt++ = 1;
    *oradt++ = 1;
    *oradt++ = 1;
   }
   return;              /* do nothing for everything else */
}
```

dpbpchk.c

```
/*  Copyright (c) Oracle Corporation 1993, 1992.  All Rights Reserved. */

/*
NAME    DPBPCHK.C

DESCRIPTION
  Check New Process

NOTES
  Desktop Performance Group

MODIFIED      (MM/DD/YY)
  W Brumiller  02/08/93 -  Correct error handling for NT
  R Keller     01/08/92 -  Initial version

*/



#include "dpbpcntl.h"



#ifdef ORA_OS2                          /* IBM OS/2 2.0          */
# define INCL_DOSPROCESS                        /*                       */
# include <os2.h>                       /*                       */

unsigned long dpbpchk(pcntl *info)
{
 ULONG pid;
 APIRET rc;

 rc = DosWaitChild(DCWA_PROCESS,
                   DCWW_WAIT,
                   &info->rcodes,
                   &pid,
                   0);

 return(info->rcodes.codeResult);
};
#endif /* ORA_OS2 */



#ifdef ORA_NT
# include <windows.h>

int dpbpchk(pcntl *info)
{
 DWORD rc;

 if (WaitForSingleObject(info->proc_info.hProcess, INFINITE) == 0XFFFFFFFF)
 {
  return -1;
 };

 if (GetExitCodeProcess(info->proc_info.hProcess, &rc) == FALSE)
 {
  return -1;
 };

 (void)CloseHandle(info->proc_info.hProcess);
 (void)CloseHandle(info->proc_info.hThread);

 return((int)rc);
}
#endif /* ORA_NT */



#ifdef ORA_AUX
# include <errno.h>
```

```
int dpbpchk(info)
pcntl *info;
{
  extern int errno;
  int byte_mask;
  int status;
  int high_byte;
  int child;
  int i;

  byte_mask = 255;       /*  low order 8 bits are 1, bits 8..31 are 0  */

  do
  {
    child = wait(&status);
    if (errno != ECHILD)
    {
      high_byte = ((status & (byte_mask << 8)) >> 8);
    };
  } while (errno != ECHILD);

  return high_byte;
}
#endif /* ORA_AUX */
```

dpbproc.c

```
/*  Copyright (c) Oracle Corporation 1993, 1992.  All Rights Reserved. */

/*
NAME    DPBPROC.C

DESCRIPTION
 Create New Process

NOTES
   Desktop Performance Group

MODIFIED       (MM/DD/YY)
   W Brumiller  02/08/93 -  Add flags for minimized window under NT
   R Keller     01/08/92 -  Initial version

*/

#include "dpbpcntl.h"



#ifdef ORA_OS2                          /* IBM OS/2 2.0             */
# define INCL_DOSPROCESS
# include <os2.h>                       /*                          */
# include <stdlib.h>                    /*                          */
# include <string.h>                    /*                          */

unsigned long dpbproc(char *i_argv[], pcntl *info)
{
  char *args;
  char *args2;
  char load_error[100];
  char pgm[44];
  APIRET rc;
  int i;

  args2 = args = (char *)malloc(128);

  strcpy(args, i_argv[0]);
  strcpy(pgm, i_argv[0]);
  strcat(pgm, ".exe");

  args2 += strlen(args) + 1;

  if (i_argv[1] != NULL)
  {
    strcpy(args2, i_argv[1]);
    for (i = 2; i_argv[i] != NULL; i++)
    {
      strcat(args2, " ");
      strcat(args2, i_argv[i]);
    };
  }
  else
  {
    *args2 = '\0';
  };
```

```
  rc = DosExecPgm(load_error,                   /* spawn process */
                  sizeof(load_error),
                  EXEC_ASYNCRESULT,
                  args,
                  0,
                  &info->rcodes,
                  pgm);

  free(args);

  return rc;
}
#endif /* ORA_OS2 */
```

```
#ifdef ORA_NT                          /* Microsoft Windows NT      */
#include <windows.h>
#include <stdlib.h>                    /*                           */
#include <string.h>                    /*                           */

int dpbproc(char *i_argv[], pcntl *info)
{
  BOOL rc;
  int i;
  char *args;
  STARTUPINFO start_info;

  args = (char *)malloc(128);

  memset(&start_info, 0x0, sizeof(STARTUPINFO));
  start_info.cb = sizeof(STARTUPINFO);
  start_info.lpTitle = i_argv[0];
  start_info.dwFlags = STARTF_USESHOWWINDOW;
  start_info.wShowWindow = SW_SHOWMINNOACTIVE;

  strcpy(args, i_argv[0]);                    /* get first str */

  for (i = 1; i_argv[i] != NULL; i++)
  {
    strcat(args," ");
    strcat(args,i_argv[i]);
  };

  if ((rc = CreateProcess(NULL,                // image name
                  args,                // command line
                  NULL,                // process security attr
                  NULL,                // thread security attr
                  TRUE,                // inherit handles
                  CREATE_NEW_CONSOLE,   // creation flags
                  NULL,                // environment blocks
                  NULL,                // current directory
                  &start_info,
                  &info->proc_info)) == FALSE)
  {
    return rc;
  };

  return 0;
};

#endif /* ORA_NT */
```

```
#ifdef ORA_AUX
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>

int dpbproc(arg_list, info)
char *arg_list[];
pcntl *info;
{
  char *path = (char *)malloc(strlen(arg_list[0]) + 3);
  pid_t child;

  sprintf(path, "./%s", arg_list[0]);

  if ((child = fork()) == (pid_t)-1)
  {
```

```
   free(path);
   return -1;
  }
  else if (child == (pid_t)0)
  {
   return execv(path, arg_list);
  }
  else
  {
   free(path);
   return 0;
  };
}
#endif /* ORA_AUX */
```

dpbprty.c

```
/* Copyright (c) Oracle Corporation 1993.  All Rights Reserved. */

/*
NAME    DPBPRTY.C

DESCRIPTION
 Set O/S Priority.

NOTES
   Desktop Performance Group

MODIFIED      (MM/DD/YY)
   MBHULLAR     03/25/94 -  Change prty_str[1) to case statement
   B Moriarty   11/11/93 -  Add Get Priority
   R Keller     10/18/93 -  Redesign
   R Keller     10/16/93 -  Iniital version

*/


#ifdef ORA_OS2
#include <string.h>
#include <sys\types.h>
#endif /* ORA_OS2 */

#ifdef ORA_NW
#endif /* ORA_NW */


#ifdef ORA_NT
# include <windows.h>
# include <string.h>
# define REALCLASS 'R'
# define HIGHCLASS 'H'
# define NORMALCLASS 'N'
# define IDLECLASS 'I'
#endif /* ORA_NT */


#ifdef ORA_AUX
#endif /* ORA_AUX */


#ifdef __STDC__
int dpbprty(char *prty_str)
#else
int dpbprty(prty_str)
char *prty_str;
#endif
{
#ifdef ORA_OS2
  return 0;
#endif /* ORA_OS2 */

#ifdef ORA_AUX
  return 0;
#endif /* ORA_AUX */

#ifdef ORA_NW
  return 0;
#endif /* ORA_NW */

#ifdef ORA_NT

  HANDLE this_process, this_thread;

  DWORD class;
```

```
 int prios;

 if ( (strlen(prty_str) > 2) || prty_str[0] == '0')
 {
  return(0);                    /* return if invalid length or 0 */
 };

 this_process = GetCurrentProcess();

 switch (prty_str[0])
 {
 case IDLECLASS:
 case 'i':
  class = IDLE_PRIORITY_CLASS;
  break;

 case NORMALCLASS:
 case 'n':
  class = NORMAL_PRIORITY_CLASS;
  break;

 case HIGHCLASS:
 case 'h':
  class = HIGH_PRIORITY_CLASS;
  break;

 case REALCLASS:
 case 'r':
  class = REALTIME_PRIORITY_CLASS;
  break;
 };

 if (!SetPriorityClass(this_process, class))
 {
  return(1);
 };

this_thread = GetCurrentThread();
switch(prty_str[1])
 {
 case '1':
 prios = THREAD_PRIORITY_IDLE;
 break;

 case '2':
 prios = THREAD_PRIORITY_LOWEST;
 break;

 case '3':
 prios = THREAD_PRIORITY_BELOW_NORMAL;
 break;

 case '4':
 prios = THREAD_PRIORITY_NORMAL;
 break;

 case '5':
 prios = THREAD_PRIORITY_ABOVE_NORMAL;
 break;

 case '6':
 prios = THREAD_PRIORITY_HIGHEST;
 break;

 case '7':
 prios = THREAD_PRIORITY_TIME_CRITICAL;
 break;

 default:
 break;
} /* End of switch statement */

 if (!SetThreadPriority(this_thread, prios))
 {
  return(2);
 }

 return 0;

# endif /* ORA_NT */

 }
```

```
#ifdef __STDC__
int dpbgetprty(char *os_pri, char *prty_str, int os_pri_len)
#else
int dpbgetprty(os_pri, prty_str, os_pri_len)
char *os_pri;
char *prty_str;
int os_pri_len;
#endif /* __STDC__ */
{
#ifdef ORA_OS2
  strncpy(os_pri,prty_str,(size_t)os_pri_len);
  return 0;
#endif /* ORA_OS2 */

#ifdef ORA_AUX
  strncpy(os_pri,prty_str,os_pri_len);
  return 0;
#endif /* ORA_AUX */

#ifdef ORA_NW
  strncpy(os_pri, prty_str, os_pri_len);
  return 0;
#endif /* ORA_NW */

#ifdef ORA_NT

  HANDLE this_process, this_thread;
  DWORD pclass;
  int tpri;

  this_process = GetCurrentProcess();
  pclass = GetPriorityClass(this_process);

  switch (pclass)
  {
  case IDLE_PRIORITY_CLASS:
    strcpy(os_pri,"I");
    break;

  case NORMAL_PRIORITY_CLASS:
    strcpy(os_pri,"N");
    break;

case HIGH_PRIORITY_CLASS:
  strcpy(os_pri,"H");
  break;

case REALTIME_PRIORITY_CLASS:
  strcpy(os_pri,"R");
  break;

default:
  strcpy(os_pri,"?");
  break;
};

this_thread=GetCurrentThread();
tpri=GetThreadPriority(this_thread);
switch (tpri)
{
case THREAD_PRIORITY_IDLE:
  strcat(os_pri,"1");
  break;

case THREAD_PRIORITY_LOWEST:
  strcat(os_pri,"2");
  break;

case THREAD_PRIORITY_BELOW_NORMAL:
  strcat(os_pri,"3");
  break;

case THREAD_PRIORITY_NORMAL:
  strcat(os_pri,"4");
  break;

case THREAD_PRIORITY_ABOVE_NORMAL:
  strcat(os_pri,"5");
  break;

case THREAD_PRIORITY_HIGHEST:
  strcat(os_pri,"6");
  break;

case THREAD_PRIORITY_TIME_CRITICAL:
```

```
   strcat(os_pri,"7");
   break;

  default:
   strcat(os_pri,"?");
   break;
  };

  return 0;

#endif /* ORA_NT */
}
```

dpbtimef.c

```
/* Copyright (c) Oracle Corporation 1993, 1992.  All Rights Reserved. */

/*
NAME    DPBTIMEF.C

DESCRIPTION
  Get time in seconds as a clock_t.

NOTES
  Desktop Performance Group

MODIFIED      (MM/DD/YY)
  B Moriarty   02/14/95 -  V4.6 fix NT & OS/2
  C Kelly      01/20/94 -  V4.4 added Netware support
  C Kelly      02/05/93 -  V3.1 added A/UX support
  R Keller     03/02/92 -  V3.0

*/

#ifdef ORA_OS2
# define ORA_PC
#endif /* ORA_OS2 */

#ifdef ORA_NT
# define ORA_PC
#endif /* ORA_NT */

#ifdef ORA_PC
# include <sys/types.h>
# include <sys/timeb.h>
# include <stdio.h>
# include <time.h>

# ifdef __STDC__
clock_t dpbtimef(void)
# else
clock_t dpbtimef()
# endif /* __STDC__ */
{
  struct timeb buf;

  ftime(&buf);
  return((clock_t) (buf.time));
}
#endif /* ORA_PC */



#ifdef ORA_AUX
# include <sys/time.h>
double dpbtimef()
{
  struct timeval t;
  int rc;

  do
  {
    rc = gettimeofday(&t, (struct timezone *)0);
  } while (rc != 0);

  return (((double)t.tv_sec) + (((double)t.tv_usec)/1000000));
}
#endif



#ifdef ORA_NW
# include <time.h>
double dpbtimef()
```

```
{
 return (double)time(NULL);    /* there is no function with greater precision */
}
#endif /* ORA_NW */


#ifdef ORA_MAC
# include <types.h>
# include <OSUtils.h>

double dpbtimef()
{
 unsigned long secs;
 GetDateTime(&secs);
 return((double) secs);
}
#endif /* ORA_MAC */
```

dpbwait.c

```
/* Copyright (c) Oracle Corporation 1993.  All Rights Reserved. */

/*
NAME    DPBWAIT.C

DESCRIPTION
  Wait for n milliseconds.

NOTES
  Desktop Performance Group

MODIFIED      (MM/DD/YY)
  R Keller    03/02/92 -  V3.0
*/



#ifdef ORA_OS2
# define INCL_DOS
# include <os2.h>
# include <time.h>

void dpbwait(clock_t i)
{
 DosSleep(i);
}
#endif /* ORA_OS2 */


#ifdef ORA_NW
# include <process.h>
void dpbwait(long i)
{
 delay((unsigned)i);
};
#endif /* ORA_NW */


#ifdef ORA_AUX
void dpbwait(wait_time)
long wait_time;
{
 unsigned secs = (unsigned)(wait_time / 1000);

 while (secs)
 {
  secs = sleep(secs);
 };
}
#endif /* ORA_AUX */


#ifdef ORA_NT
# include <windows.h>

void dpbwait(long i)
{
 Sleep(i);
}
#endif /* ORA_NT */
```

```
#ifdef ORA_DOS
# include <time.h>

void dpbwait(long i)
{
  long current_time;
  long target_time;

  current_time = time(NULL);
  target_time = current_time + i/1000;

  while (current_time < target_time)
  {
    current_time = time(NULL);
  };
}
#endif /* ORA_DOS */
```

dpbxtpgm.c

```
/* Copyright (c) Oracle Corporation 1994.  All Rights Reserved. */

/*
NAME    DPBXTPGM.C

DESCRIPTION
   Routine that performs any o/s specific program exit operations.

NOTES
   Desktop Performance Group

MODIFIED      (MM/DD/YY)
   C Kelly     04/21/94 -  V4.4 created to support Netware NLMs

*/



#ifdef ORA_NW
#include <process.h>
#include <library.h>
extern int samtid;
extern int samtgid;

#else /* ORA_NW */
#endif /* ORA_NW */



#ifdef __STDC__
void dpbxtpgm(void)
#else
void dpbxtpgm()
#endif /* __STDC__ */
{
#ifdef ORA_NW

/*
** Cleanup code for NetWare.
**   This routine will cleanup any Oracle connection should the module
**   be unexpectedly unloaded.
*/

  int oldtgid;

  oldtgid = SetThreadGroupID(samtgid);  /* switch to this NLM's thread group */
  OraClientExit(samtid);              /* cleanup Oracle connection    */
  SetThreadGroupID(oldtgid);          /* reset the thread group       */

#else /* ORA_NW */

  return;             /* do nothing for everything else */

#endif /* ORA_NW */
}
```

gettime.c

```
#ifdef RCSID
static char *RCSid =
   "$Header: gettime.c 7030100.1 96/05/21 15:31:36 plai Generic<base> $ Copyr (c)
1993 Oracle";
#endif /* RCSID */
```

```
/*+=========================================================
======+
|      Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
|                OPEN SYSTEMS PERFORMANCE GROUP                 |
|                   All Rights Reserved                         |

+=========================================================
======+
|
| FILENAME
|          gettime.c
|
| ROUTINES
|          gettime
|          getcpu
| DESCRIPTION
|          get wall clock time.
|          get cpu time.
| NOTES
|          Both routines return time in seconds as a double.

+=========================================================
=====*/
/*
**  Options:
**     TIME_W_TIMES:      implement gettime() with times().
**     TIME_W_GETTIME:    implement gettime() with gettimeofday().
**     CPU_W_TIMES:       implement getcpu()  with times().
**     CPU_W_GETRU:       implement getcpu()  with getrusage().
**     GETRU_STATS:       collect getrusage statistics
**     GET_P_STATS:       collect get_process_stats statistics
*/

#if defined(sequent) || defined(SEQ_PSX)
# define GET_P_STATS
#endif /* sequent */

#if defined(aix) || defined(AIXRIOS)
# define TIME_W_GETTIME
# define CPU_W_TIMES
# define GETRU_STATS
#endif /* AIXRIOS */

#if defined(a_osf) || defined(A_OSF)
# define TIME_W_GETTIME
# define CPU_W_GETRU
# define GETRU_STATS
#endif /* AIXRIOS */

#if !defined(TIME_W_GETTIME) && !defined(TIME_W_TIMES)
# define TIME_W_TIMES
#endif

#if !defined(CPU_W_GETRU) && !defined(CPU_W_TIMES)
# define CPU_W_TIMES
#endif

#ifdef GET_P_STATS
# ifdef GETRU_STATS
#  undef GETRU_STATS
# endif
#endif

#if defined(TIME_W_GETTIME) || defined(CPU_W_GETRU) ||
defined(GETRU_STATS)
# include <sys/time.h>
#endif /* TIME_W_GETTIME || CPU_W_GETRU || GETRU_STATS */

#if defined(CPU_W_GETRU) || defined(GETRU_STATS)
# include <sys/resource.h>
#endif /* CPU_W_GETRU || GETRU_STATS */

#if defined(TIME_W_TIMES) || defined (CPU_W_TIMES)
# include <sys/types.h>
# include <sys/times.h>
# include <sys/param.h>  /* most systems define HZ here */
# include <sys/unistd.h>
#endif /* TIME_W_TIMES or CPU_W_TIMES */

#ifdef GET_P_STATS
# include <sys/types.h>
# include <sys/procstats.h>
#endif /* GET_P_STATS */

# include <stdio.h>
```

```
#ifdef GETRU_STATS
struct rusage selfru;
struct rusage kidsru;
#endif /* GETRU_STATS */

#ifdef GET_P_STATS
struct process_stats selfru;
struct process_stats kidsru;
#endif /* GET_P_STATS */




void getwait(clock_t secs)
{
  printf("sleep = %lu\n", (secs/1000) / HZ);
  printf("hz = %lu\n", HZ);
  sleep((secs/1000) / HZ);
}

clock_t getetime()
{
   struct tms buf;

   return ((times (&buf) / HZ)*10000);
}

double gettime ()

{

#ifdef TIME_W_GETTIME
   struct timeval tv;

   (void) gettimeofday (&tv, (struct timezone *) 0);
   return ((double) tv.tv_sec + (1.0e-6 * (double) tv.tv_usec));
#endif /* TIME_W_GETTIME */

#ifdef TIME_W_TIMES
   struct tms buf;

   return ((double) times (&buf) / HZ);
#endif /* TIME_W_TIMES */
}



double getcpu ()

{

#ifdef CPU_W_TIMES
   struct tms buf;

   (void) times (&buf);
   return (((double) buf.tms_utime + (double) buf.tms_stime) / HZ);
#endif /* CPU_W_TIMES */

#ifdef CPU_W_GETRU
   struct rusage ru;
   double usecs;

   (void) getrusage (0, &ru);
   usecs = 1.0e-6 * (double) (ru.ru_utime.tv_usec + ru.ru_stime.tv_usec);
   return ((double) (ru.ru_utime.tv_sec + ru.ru_stime.tv_sec) + usecs);
#endif /* CPU_W_GETRU */

}



getru (fp, kids, config, runname, proc_no)

FILE *fp;
int kids;
char *config;
char *runname;
int proc_no;

{

#ifdef GETRU_STATS
   struct rusage ru;

   fprintf (fp, "%-10.10s %-10.10s %10d %10d ", config,runname, proc_no, kids);
```

```
  getrusage (kids ? RUSAGE_CHILDREN : RUSAGE_SELF, &ru);
  print_ru (fp, &ru);
  fprintf (fp, "\n");
#endif /* GETRU_STATS */

#ifdef GET_P_STATS
  timeval_t tv;
  struct process_stats ru;

  fprintf (fp, "%-10.10s %-10.10s %10d %10d ", config,runname, proc_no, kids);
  if (kids)
    get_process_stats (&tv, PS_SELF, (struct process_stats *) 0, &ru);
  else
    get_process_stats (&tv, PS_SELF, &ru, (struct process_stats *) 0);
  print_ru (fp, &ru);
  fprintf (fp, "\n");
#endif /* GET_P_STATS */

}



getru1 (kids)

int kids;

{

#ifdef GETRU_STATS
  if (kids) {
    memset (&kidsru, 0, sizeof (kidsru));
    getrusage (RUSAGE_CHILDREN, &kidsru);
  }
  else {
    memset (&selfru, 0, sizeof (selfru));
    getrusage (RUSAGE_SELF, &selfru);
  }
#endif /* GETRU_STATS */

#ifdef GET_P_STATS
  timeval_t tv;

  if (kids) {
    memset (&kidsru, 0, sizeof (kidsru));
    get_process_stats (&tv, PS_SELF, (struct process_stats *) 0, &kidsru);
  }
  else {
    memset (&selfru, 0, sizeof (selfru));
    get_process_stats (&tv, PS_SELF, &selfru, (struct process_stats *) 0);
  }
#endif /* GET_P_STATS */

}



getru2 (fp, kids, config, runname, proc_no)

FILE *fp;
int kids;
char *config;
char *runname;
int proc_no;

{

#ifdef GETRU_STATS
  struct rusage ru;

  fprintf (fp, "%-10.10s %-10.10s %10d %10d ", config, runname, proc_no, kids);
  getrusage (kids ? RUSAGE_CHILDREN : RUSAGE_SELF, &ru);
  if (kids)
    diffru (&ru, &kidsru);
  else
    diffru (&ru, &selfru);
  print_ru (fp, &ru);
  fprintf (fp, "\n");
#endif /* GETRU_STATS */

#ifdef GET_P_STATS
  timeval_t tv;
  struct process_stats ru;

  fprintf (fp, "%-10.10s %-10.10s %10d %10d ", config, runname, proc_no, kids);
  if (kids)
    get_process_stats (&tv, PS_SELF, (struct process_stats *) 0, &ru);
```

```
  else
    get_process_stats (&tv, PS_SELF, &ru, (struct process_stats *) 0);
  if (kids)
    diffru (&ru, &kidsru);
  else
    diffru (&ru, &selfru);
  print_ru (fp, &ru);
  fprintf (fp, "\n");
#endif /* GET_P_STATS */

}



#ifdef GETRU_STATS

print_ru (fp, ru)

FILE *fp;
struct rusage *ru;

{

  fprintf (fp, "%10ld ", ru->ru_utime.tv_sec * 1000 +
                   (ru->ru_utime.tv_usec/1000));
  fprintf (fp, "%10ld ", ru->ru_stime.tv_sec * 1000 +
                   (ru->ru_stime.tv_usec/1000));
  fprintf (fp, "%10ld ", ru->ru_maxrss);
  fprintf (fp, "%10ld ", ru->ru_majflt);
  fprintf (fp, "%10ld ", ru->ru_minflt);
  fprintf (fp, "%10ld ", 0);
  fprintf (fp, "%10ld ", 0);
  fprintf (fp, "%10ld ", 0);
  fprintf (fp, "%10ld ", ru->ru_nswap);
  fprintf (fp, "%10ld ", 0);
  fprintf (fp, "%10ld ", ru->ru_nvcsw);
  fprintf (fp, "%10ld ", ru->ru_nivcsw);
  fprintf (fp, "%10ld ", ru->ru_nsignals);
  fprintf (fp, "%10ld ", 0);
  fprintf (fp, "%10ld ", 0);
  fprintf (fp, "%10ld ", ru->ru_inblock);
  fprintf (fp, "%10ld ", ru->ru_oublock);
  fprintf (fp, "%10ld ", 0);
  fprintf (fp, "%10ld",  0);

}



diffru (ru2, ru)

struct rusage *ru2;
struct rusage *ru;

{

  ru2->ru_utime.tv_sec -= ru->ru_utime.tv_sec;
  ru2->ru_utime.tv_usec -= ru->ru_utime.tv_usec;
  ru2->ru_stime.tv_sec -= ru->ru_stime.tv_sec;
  ru2->ru_stime.tv_usec -= ru->ru_stime.tv_usec;
  ru2->ru_maxrss -= ru->ru_maxrss;
  ru2->ru_ixrss -= ru->ru_ixrss;
  ru2->ru_idrss -= ru->ru_idrss;
  ru2->ru_minflt -= ru->ru_minflt;
  ru2->ru_majflt -= ru->ru_majflt;
  ru2->ru_nswap -= ru->ru_nswap;
  ru2->ru_inblock -= ru->ru_inblock;
  ru2->ru_oublock -= ru->ru_oublock;
  ru2->ru_msgsnd -= ru->ru_msgsnd;
  ru2->ru_msgrcv -= ru->ru_msgrcv;
  ru2->ru_nsignals -= ru->ru_nsignals;
  ru2->ru_nvcsw -= ru->ru_nvcsw;
  ru2->ru_nivcsw -= ru->ru_nivcsw;

}

#endif /* GETRU_STATS */



#ifdef GET_P_STATS

print_ru (fp, ps)

FILE *fp;
struct process_stats *ps;
```

```
{

   fprintf (fp, "%lu ", ps->ps_utime.tv_sec * 1000 +
                    (ps->ps_utime.tv_usec/1000));
   fprintf (fp, "%lu ", ps->ps_stime.tv_sec * 1000 +
                     (ps->ps_stime.tv_usec/1000));
   fprintf (fp, "%lu ", ps->ps_maxrss);
   fprintf (fp, "%lu ", ps->ps_pagein);
   fprintf (fp, "%lu ", ps->ps_reclaim);
   fprintf (fp, "%lu ", ps->ps_zerofill);
   fprintf (fp, "%lu ", ps->ps_pffincr);
   fprintf (fp, "%lu ", ps->ps_pffdecr);
   fprintf (fp, "%lu ", ps->ps_swap);
   fprintf (fp, "%lu ", ps->ps_syscall);
   fprintf (fp, "%lu ", ps->ps_volcsw);
   fprintf (fp, "%lu ", ps->ps_involcsw);
   fprintf (fp, "%lu ", ps->ps_signal);
   fprintf (fp, "%lu ", ps->ps_lread);
   fprintf (fp, "%lu ", ps->ps_lwrite);
   fprintf (fp, "%lu ", ps->ps_bread);
   fprintf (fp, "%lu ", ps->ps_bwrite);
   fprintf (fp, "%lu ", ps->ps_phread);
   fprintf (fp, "%lu",  ps->ps_phwrite);

}



diffru (ru2, ru)

struct process_stats *ru2;
struct process_stats *ru;

{

   ru2->ps_utime.tv_sec -= ru->ps_utime.tv_sec;
   ru2->ps_utime.tv_usec -= ru->ps_utime.tv_usec;
   ru2->ps_stime.tv_sec -= ru->ps_stime.tv_sec;
   ru2->ps_stime.tv_usec -= ru->ps_stime.tv_usec;
   ru2->ps_maxrss -= ru->ps_maxrss;
   ru2->ps_pagein -= ru->ps_pagein;
   ru2->ps_reclaim -= ru->ps_reclaim;
   ru2->ps_zerofill -= ru->ps_zerofill;
   ru2->ps_pffincr -= ru->ps_pffincr;
   ru2->ps_pffdecr -= ru->ps_pffdecr;
   ru2->ps_swap -= ru->ps_swap;
   ru2->ps_syscall -= ru->ps_syscall;
   ru2->ps_volcsw -= ru->ps_volcsw;
   ru2->ps_involcsw -= ru->ps_involcsw;
   ru2->ps_signal -= ru->ps_signal;
   ru2->ps_lread -= ru->ps_lread;
   ru2->ps_lwrite -= ru->ps_lwrite;
   ru2->ps_bread -= ru->ps_bread;
   ru2->ps_bwrite -= ru->ps_bwrite;
   ru2->ps_phread -= ru->ps_phread;
   ru2->ps_phwrite -= ru->ps_phwrite;

}

#endif /* GET_P_STATS */
```

tpccload.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: tpccload.c 7030100.1 96/05/13 16:20:36 plai Generic<base> $ Copyr (c)
1993 Oracle";
#endif /* RCSID */

/*=============================================================
======+
 |      Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
 |                OPEN SYSTEMS PERFORMANCE GROUP                 |
 |                    All Rights Reserved                        |

 +=============================================================
======+
 | FILENAME
 |   tpccload.c
 | DESCRIPTION
 |   Load or generate TPC-C database tables.
 |   Usage: tpccload -M <# of wares> [options]
 |            options: -A  load all tables
 |                     -w  load ware table
 |                     -d  load dist table
```

```
|                 -c  load cust table (cluster around c_w_id)
|                 -C  load cust table (cluster around c_id)
|                 -i  load item table
|                 -s  load stok table (cluster around s_w_id)
|                 -S  load stok table (cluster around s_i_id)
|                 -h  load hist table
|                 -n  load new-order table
|                 -o <oline file> load order and order-line table
|                 -b <ware#>  beginning ware number
|                 -e <ware#>  ending ware number
|                 -j <item#>  beginning item number (with -S)
|                 -k <item#>  ending item number (with -S)
|                 -l <cid#>  beginning cid number (with -C)
|                 -m <cid#>  ending cid number (with -C)
|                 -g  generate rows to standard output

+=============================================================
======*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/types.h>
#include "tpcc.h"

#ifdef ORA_NT
#undef boolean
#include <process.h>
#include "dpbcore.h"
# define gettime dpbtimef
# define getcpu  dpbcpu
#define lrand48() ((long)rand() <<15 | rand())
#ifdef __STDC__
#  define PROTO(args)    args
#else
#  define PROTO(args)    ()
#endif
#endif

#define DISTARR 10              /* dist insert array size     */
#define CUSTARR 100             /* cust insert array size     */
#define STOCARR 100             /* stok insert array size     */
#define ITEMARR 100             /* item insert array size     */
#define HISTARR 100          /* hist insert array size    */
#define ORDEARR 100          /* order insert array size      */
#define NEWOARR 100          /* new order insert array size  */

#define DISTFAC 10              /* max. dist id      */
#define CUSTFAC 3000            /* max. cust id      */
#define STOCFAC 100000                  /* max. stok id      */
#define ITEMFAC 100000                  /* max. item id              */
#define HISTFAC 30000        /* history / warehouse  */
#define ORDEFAC 3000         /* order / district     */
#define NEWOFAC 900          /* new order / district */

#define C       0           /* constant in non-uniform dist. eqt. */
#define CNUM1   1           /* first constant in non-uniform dist. eqt. */
#define CNUM2   2           /* second constant in non-uniform dist. eqt. */
#define CNUM3   3           /* third constant in non-uniform dist. eqt. */

#define SEED    2           /* seed for random functions */

#define NOT_SERIALIZABLE  8177  /* ORA-08177: transaction not serializable */
#define SNAPSHOT_TOO_OLD  1555  /* ORA-01555: snapshot too old */
#define RECOVERR -10
#define IRRECERR -20

#define SQLTXTW "INSERT INTO ware (w_id, w_ytd, w_tax, w_name, w_street_1,
w_street_2, w_city, w_state, w_zip) VALUES (:w_id,
30000000, :w_tax, :w_name, :w_street_1, \
  :w_street_2, :w_city, :w_state, :w_zip)"

#define SQLTXTD "INSERT INTO dist (d_id, d_w_id, d_ytd, d_tax, d_next_o_id,
d_name, d_street_1, d_street_2, d_city, d_state, d_zip) VALUES
(:d_id, :d_w_id,3000000, :d_tax, \
    3001, :d_name, :d_street_1, :d_street_2, :d_city, :d_state, :d_zip)"

#define SQLTXTCQUERY "select /*+ HASH ( cust ) */ count(*) from cust where
c_w_id = :s_c_w_id and c_d_id = :s_c_d_id and c_id = :s_c_id"

#define SQLTXTC "INSERT INTO cust (C_ID, C_D_ID, C_W_ID, C_FIRST,
C_MIDDLE, C_LAST, C_STREET_1, C_STREET_2, C_CITY, C_STATE, C_ZIP,
C_PHONE, C_SINCE, C_CREDIT, C_CREDIT_LIM, C_DISCOUNT,
C_BALANCE, C_YTD_PAYMENT, C_PAYMENT_CNT, C_DELIVERY_CNT,
C_DATA) VALUES (:c_id, :c_d_id, :c_w_id, \
```

```
  :c_first, 'OE', :c_last, :c_street_1, :c_street_2, :c_city, :c_state, \
  :c_zip, :c_phone, SYSDATE, :c_credit, 5000000, :c_discount, -1000, 1000, 1, \
  0, :c_data)"

#define SQLTXTH "INSERT INTO hist (h_c_id, h_c_d_id, h_c_w_id, h_d_id,
h_w_id, h_date, h_amount, h_data) VALUES (:h_c_id, :h_c_d_id, :h_c_w_id, \
  :h_d_id, :h_w_id, SYSDATE, 1000, :h_data)"

#define SQLTXTSQUERY "select /*+ HASH ( stok ) */ count(*) from stok where
s_w_id = :s_s_w_id and s_i_id = :s_s_i_id "

#define SQLTXTS "INSERT INTO stok (s_i_id, s_w_id, s_quantity,s_dist_01,
s_dist_02, s_dist_03, s_dist_04, s_dist_05 , s_dist_06, s_dist_07, s_dist_08, s_dist_09,
s_dist_10, s_ytd, s_order_cnt, s_remote_cnt, s_data) \
VALUES (:s_i_id, :s_w_id, :s_quantity, \
  :s_dist_01, :s_dist_02, :s_dist_03, :s_dist_04, :s_dist_05, :s_dist_06, \
  :s_dist_07, :s_dist_08, :s_dist_09, :s_dist_10, 0, 0, 0, :s_data)" \

#define SQLTXTI "INSERT INTO item
(I_ID,I_IM_ID,I_NAME,I_PRICE,I_DATA) VALUES
(:i_id, :i_im_id, :i_name, :i_price, \
  :i_data)"

#define SQLTXTO1 "INSERT INTO ordr (O_ID,
O_D_ID,O_W_ID,O_C_ID,O_ENTRY_D,O_CARRIER_ID,O_OL_CNT,O_ALL_L
OCAL) \
  VALUES (:o_id, :o_d_id, :o_w_id, :o_c_id, \
  SYSDATE, :o_carrier_id, :o_ol_cnt, 1)"

#define SQLTXTO2 "INSERT INTO ordr (O_ID,
O_D_ID,O_W_ID,O_C_ID,O_ENTRY_D,O_CARRIER_ID,O_OL_CNT,O_ALL_L
OCAL) \
  VALUES (:o_id, :o_d_id, :o_w_id, :o_c_id, \
  SYSDATE, 11, :o_ol_cnt, 1)"

#define SQLTXTOL1 "INSERT INTO ordl (OL_O_ID, OL_D_ID, OL_W_ID,
OL_NUMBER, OL_DELIVERY_D, OL_I_ID, OL_SUPPLY_W_ID,
OL_QUANTITY, OL_AMOUNT, OL_DIST_INFO) \
  VALUES (:ol_o_id, :ol_d_id, \
  :ol_w_id, :ol_number, SYSDATE, :ol_i_id, :ol_supply_w_id, 5, 0, \
  :ol_dist_info)"

#define SQLTXTOL2 "INSERT INTO ordl (OL_O_ID, OL_D_ID, OL_W_ID,
OL_NUMBER, OL_DELIVERY_D, OL_I_ID, OL_SUPPLY_W_ID,
OL_QUANTITY, OL_AMOUNT, OL_DIST_INFO) \
  VALUES (:ol_o_id, :ol_d_id, \
  :ol_w_id, :ol_number, to_date('01-Jan-1811'), :ol_i_id, :ol_supply_w_id,
5, :ol_amount, \
  :ol_dist_info)"

#define SQLTXTNO "INSERT INTO nord (no_o_id, no_d_id, no_w_id) VALUES
(:no_o_id, :no_d_id, :no_w_id)"

#define SQLTXTENHA "alter session set \"_enable_hash_overflow\"=true"
#define SQLTXTDIHA "alter session set \"_enable_hash_overflow\"=false"

static char *lastname[] = {
  "BAR",
  "OUGHT",
  "ABLE",
  "PRI",
  "PRES",
  "ESE",
  "ANTI",
  "CALLY",
  "ATION",
  "EING"
};

char num9[10];
char num16[17];
char str2[3];
char str24[15][25];
int randperm3000[3000];

void initperm();
void randstr();
void randdatastr();
void randnum();
void randlastname (char*, int);
int NURand();
void sysdate();

OCIEnv *tpcenv;
OCIServer *tpcsrv;
OCIError *errhp;
```

```
OCISvcCtx *tpcsvc;
OCISession *tpcusr;

OCIStmt *curw;
OCIStmt *curd;
OCIStmt *curc;
OCIStmt *curcs;
OCIStmt *curh;
OCIStmt *curs;
OCIStmt *curss;
OCIStmt *curi;
OCIStmt *curo1;
OCIStmt *curo2;
OCIStmt *curol1;
OCIStmt *curol2;
OCIStmt *curno;

OCIBind *w_id_bp = (OCIBind *) 0;
OCIBind *w_name_bp = (OCIBind *) 0;
OCIBind *w_street1_bp = (OCIBind *) 0;
OCIBind *w_street2_bp = (OCIBind *) 0;
OCIBind *w_city_bp = (OCIBind *) 0;
OCIBind *w_state_bp = (OCIBind *) 0;
OCIBind *w_zip_bp = (OCIBind *) 0;
OCIBind *w_tax_bp = (OCIBind *) 0;

OCIBind *d_id_bp = (OCIBind *) 0;
OCIBind *d_w_id_bp = (OCIBind *) 0;
OCIBind *d_name_bp = (OCIBind *) 0;
OCIBind *d_street1_bp = (OCIBind *) 0;
OCIBind *d_street2_bp = (OCIBind *) 0;
OCIBind *d_city_bp = (OCIBind *) 0;
OCIBind *d_state_bp = (OCIBind *) 0;
OCIBind *d_zip_bp = (OCIBind *) 0;
OCIBind *d_tax_bp = (OCIBind *) 0;

OCIDefine *s_c_ret_bp = (OCIDefine *) 0;
OCIBind *s_c_id_bp = (OCIBind *) 0;
OCIBind *s_c_d_id_bp = (OCIBind *) 0;
OCIBind *s_c_w_id_bp = (OCIBind *) 0;

OCIBind *c_id_bp = (OCIBind *) 0;
OCIBind *c_d_id_bp = (OCIBind *) 0;
OCIBind *c_w_id_bp = (OCIBind *) 0;
OCIBind *c_first_bp = (OCIBind *) 0;
OCIBind *c_last_bp = (OCIBind *) 0;
OCIBind *c_street1_bp = (OCIBind *) 0;
OCIBind *c_street2_bp = (OCIBind *) 0;
OCIBind *c_city_bp = (OCIBind *) 0;
OCIBind *c_state_bp = (OCIBind *) 0;
OCIBind *c_zip_bp = (OCIBind *) 0;
OCIBind *c_phone_bp = (OCIBind *) 0;
OCIBind *c_discount_bp = (OCIBind *) 0;
OCIBind *c_credit_bp = (OCIBind *) 0;
OCIBind *c_data_bp = (OCIBind *) 0;

OCIBind *i_id_bp = (OCIBind *) 0;
OCIBind *i_im_id_bp = (OCIBind *) 0;
OCIBind *i_name_bp = (OCIBind *) 0;
OCIBind *i_price_bp = (OCIBind *) 0;
OCIBind *i_data_bp = (OCIBind *) 0;

OCIDefine *s_s_ret_bp = (OCIDefine *) 0;
OCIBind *s_s_i_id_bp = (OCIBind *) 0;
OCIBind *s_s_w_id_bp = (OCIBind *) 0;

OCIBind *s_i_id_bp = (OCIBind *) 0;
OCIBind *s_w_id_bp = (OCIBind *) 0;
OCIBind *s_quantity_bp = (OCIBind *) 0;
OCIBind *s_dist_01_bp = (OCIBind *) 0;
OCIBind *s_dist_02_bp = (OCIBind *) 0;
OCIBind *s_dist_03_bp = (OCIBind *) 0;
OCIBind *s_dist_04_bp = (OCIBind *) 0;
OCIBind *s_dist_05_bp = (OCIBind *) 0;
OCIBind *s_dist_06_bp = (OCIBind *) 0;
OCIBind *s_dist_07_bp = (OCIBind *) 0;
OCIBind *s_dist_08_bp = (OCIBind *) 0;
OCIBind *s_dist_09_bp = (OCIBind *) 0;
OCIBind *s_dist_10_bp = (OCIBind *) 0;
OCIBind *s_data_bp = (OCIBind *) 0;

OCIBind *h_c_id_bp = (OCIBind *) 0;
OCIBind *h_c_d_id_bp = (OCIBind *) 0;
OCIBind *h_c_w_id_bp = (OCIBind *) 0;
OCIBind *h_d_id_bp = (OCIBind *) 0;
OCIBind *h_w_id_bp = (OCIBind *) 0;
```

```
OCIBind *h_data_bp = (OCIBind *) 0;

OCIBind *ol_o_id_bp = (OCIBind *) 0;
OCIBind *ol_d_id_bp = (OCIBind *) 0;
OCIBind *ol_w_id_bp = (OCIBind *) 0;
OCIBind *ol_i_id_bp = (OCIBind *) 0;
OCIBind *ol_number_bp = (OCIBind *) 0;
OCIBind *ol_supply_w_id_bp = (OCIBind *) 0;
OCIBind *ol_dist_info_bp = (OCIBind *) 0;
OCIBind *ol_amount_bp = (OCIBind *) 0;

OCIBind *o_id_bp = (OCIBind *) 0;
OCIBind *o_d_id_bp = (OCIBind *) 0;
OCIBind *o_w_id_bp = (OCIBind *) 0;
OCIBind *o_c_id_bp = (OCIBind *) 0;
OCIBind *o_carrier_id_bp = (OCIBind *) 0;
OCIBind *o_ol_cnt_bp  = (OCIBind *) 0;
OCIBind *o_ocnt_bp = (OCIBind *) 0;
OCIBind *o_olcnt_bp  = (OCIBind *) 0;


OCIBind *no_o_id_bp  = (OCIBind *) 0;
OCIBind *no_d_id_bp  = (OCIBind *) 0;
OCIBind *no_w_id_bp  = (OCIBind *) 0;

void myusage()
{
  fprintf (stderr, "\n");
  fprintf (stderr, "Usage:\ttpccload -M <multiplier> [options]\n");
  fprintf (stderr, "options:\n");
  fprintf (stderr, "\t-A :\tload all tables\n");
  fprintf (stderr, "\t-w :\tload ware table\n");
  fprintf (stderr, "\t-d :\tload dist table\n");
  fprintf (stderr, "\t-c :\tload cust table (cluster around c_w_id\n");
  fprintf (stderr, "\t-C :\tload cust table (cluster around c_id\n");
  fprintf (stderr, "\t-i :\tload item table\n");
  fprintf (stderr, "\t-s :\tload stok table (cluster around s_w_id)\n");
  fprintf (stderr, "\t-S :\tload stok table (cluster around s_i_id)\n");
  fprintf (stderr, "\t-h :\tload hist table\n");
  fprintf (stderr, "\t-n :\tload new-order table\n");
  fprintf (stderr, "\t-o <oline file> :\tload order and order-line table\n");
  fprintf (stderr, "\t-b <ware#> :\tbeginning ware number\n");
  fprintf (stderr, "\t-e <ware#> :\tending ware number\n");
  fprintf (stderr, "\t-j <item#> :\tbeginning item number (with -S)\n");
  fprintf (stderr, "\t-k <item#> :\tending item number (with -S)\n");
  fprintf (stderr, "\t-l <cid#> :\tbeginning cid number (with -C)\n");
  fprintf (stderr, "\t-m <cid#> :\tending cid number (with -C)\n");
  fprintf (stderr, "\t-g :\tgenerate rows to standard output\n");
  fprintf (stderr,"\t   $tpcc_bench must be set to the location of the kit\n");
  fprintf (stderr, "\n");
  exit(1);
}

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

   sprintf(realfile,"%s",fnam);
   fd = fopen(realfile,"r");
   if (!fd)
   {
    return (0);
   }
   while (fgets((char *)linebuf+nulpt, SQL_BUF_SIZE, fd))
   {
      nulpt = strlen((char *)linebuf);
   }
   return(nulpt);
}

void quit()
{
  OCIERROR(errhp,OCISessionEnd ( tpcsvc,errhp, tpcusr, OCI_DEFAULT));
  OCIERROR(errhp,OCIServerDetach ( tpcsrv, errhp, OCI_DEFAULT));
  OCIHandleFree((dvoid *)tpcusr, OCI_HTYPE_SESSION);
  OCIHandleFree((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX);
  OCIHandleFree((dvoid *)errhp, OCI_HTYPE_ERROR);
  OCIHandleFree((dvoid *)tpcsrv, OCI_HTYPE_SERVER);
  OCIHandleFree((dvoid *)tpcenv, OCI_HTYPE_ENV);
}

void main (argc, argv)
```

```
int argc;
char *argv[];
{
  char *uid="tpcc";
  char *pwd="tpcc";
  int scale=0;
  int i, j;
  int loop;
  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 o_cnt;
  int ol_cnt;

  ub2 ol_o_id_len[1500];
  ub2 ol_d_id_len[1500];
  ub2 ol_w_id_len[1500];
  ub2 ol_number_len[1500];
  ub2 ol_i_id_len[1500];
  ub2 ol_supply_w_id_len[1500];
  ub2 ol_dist_info_len[1500];
  ub2 ol_amount_len[1500];

  ub4 ol_o_id_clen;
  ub4 ol_d_id_clen;
  ub4 ol_w_id_clen;
  ub4 ol_number_clen;
  ub4 ol_i_id_clen;
  ub4 ol_supply_w_id_clen;
  ub4 ol_dist_info_clen;
  ub4 ol_amount_clen;

  ub2 o_id_len[100];
  ub2 o_d_id_len[100];
  ub2 o_w_id_len[100];
  ub2 o_c_id_len[100];
  ub2 o_carrier_id_len[100];
  ub2 o_ol_cnt_len[100];

  ub4 o_id_clen;
  ub4 o_d_id_clen;
  ub4 o_w_id_clen;
  ub4 o_c_id_clen;
  ub4 o_carrier_id_clen;
  ub4 o_ol_cnt_clen;

  text stmbuf[16*1024];

  int no_o_id[100];
  int no_d_id[100];
  int no_w_id[100];

  char sdate[30];

#ifdef ORA_NT
  clock_t begin_time, end_time;
  clock_t begin_cpu, end_cpu;

  char *arg_ptr,  **end_args;
#else
  double begin_time, end_time;
  double begin_cpu, end_cpu;
  double gettime(), getcpu();

  extern int getopt();
  extern char *optarg;
  extern int optind, opterr;
  int opt;
#endif

  char    *argstr="M:AwdcCisShno:b:e:j:k:l:m:g";
  int do_A=0;
  int do_w=0;
  int do_d=0;
  int do_i=0;
  int do_c=0;
  int do_C=0;
  int do_s=0;
  int do_S=0;
  int do_h=0;
```

```
  int do_o=0;
  int do_n=0;
  int gen=0;
  int bware=1;
  int eware=0;
  int bitem=1;
  int eitem=0;
  int bcid=1;
  int ecid=0;

  FILE *olfp=NULL;
  char olfname[100];
  char* basename;
  int status;
#ifdef ORA_NT
  char fname[100];
  FILE *logfile;
#endif /* ORA_NT */

/*---------------------------------------------------------------+
 | Parse command line -- look for scale factor.                  |
 +---------------------------------------------------------------*/

  if (argc == 1) {
    myusage ();
  }

#ifdef ORA_NT
  end_args = argv + argc;
  for (++argv; argv < end_args; )
  {
    arg_ptr = *argv++;

      if (*arg_ptr != '-')
      {
        myusage ();
      } else
      {
     switch (arg_ptr[1]) {
      case '?': myusage ();
                break;
      case 'M': scale = atoi (*argv++);
                break;
      case 'A': do_A = 1;
                break;
      case 'w': do_w = 1;
                break;
      case 'd': do_d = 1;
                break;
      case 'c': do_c = 1;
                break;
      case 'C': do_C = 1;
                break;
      case 'i': do_i = 1;
                break;
      case 's': do_s = 1;
                break;
      case 'S': do_S = 1;
                break;
      case 'h': do_h = 1;
                break;
      case 'n': do_n = 1;
                break;
      case 'o': do_o = 1;
                strcpy (olfname, *argv++);
                break;
      case 'b': bware = atoi (*argv++);
                break;
      case 'e': eware = atoi (*argv++);
                break;
      case 'j': bitem = atoi (*argv++);
                break;
      case 'k': eitem = atoi (*argv++);
                break;
      case 'l': bcid = atoi (*argv++);
                break;
      case 'm': ecid = atoi (*argv++);
                break;
      case 'g': gen = 1;
                strcpy (fname, *argv++);
                break;
      case 'l': logfile=fopen(*argv++,"w");
                break;
      default: fprintf (stderr, "THIS SHOULD NEVER HAPPEN!!!\n");
               fprintf (stderr, "(reached default case in getopt ())\n");
               myusage ();
```

```
      }
       }
  }


#else

  while ((opt = getopt (argc, argv, argstr)) != -1) {
    switch (opt) {
      case '?': myusage ();
                break;
      case 'M': scale = atoi (optarg);
                break;
      case 'A': do_A = 1;
                break;
      case 'w': do_w = 1;
                break;
      case 'd': do_d = 1;
                break;
      case 'c': do_c = 1;
                break;
      case 'C': do_C = 1;
                break;
      case 'i': do_i = 1;
                break;
      case 's': do_s = 1;
                break;
      case 'S': do_S = 1;
                break;
      case 'h': do_h = 1;
                break;
      case 'n': do_n = 1;
                break;
      case 'o': do_o = 1;
                strcpy (olfname, optarg);
                break;
      case 'b': bware = atoi (optarg);
                break;
      case 'e': eware = atoi (optarg);
                break;
      case 'j': bitem = atoi (optarg);
                break;
      case 'k': eitem = atoi (optarg);
                break;
      case 'l': bcid = atoi (optarg);
                break;
      case 'm': ecid = atoi (optarg);
                break;
      case 'g': gen = 1;
                break;
      default: fprintf (stderr, "THIS SHOULD NEVER HAPPEN!!!\n");
               fprintf (stderr, "(reached default case in getopt ())\n");
               myusage ();
    }
  }

# endif /* ORA_NT */

/*----------------------------------------------------------------*|
|            Rudimentary error checking                           |
|*----------------------------------------------------------------*/

  if (scale  < 1) {
    fprintf (stderr, "Invalid scale factor: '%d'\n", scale);
    myusage ();
  }

  if (!(do_A || do_w || do_d || do_c || do_C || do_i || do_s || do_S || do_h || do_o ||
      do_n)) {
    fprintf (stderr, "What should I load???\n");
    myusage ();
  }

  if (gen && (do_A || (do_w + do_d + do_c + do_C + do_i + do_s + do_S + do_h +
do_o +
                       do_n > 1))) {
    fprintf (stderr, "Can only generate table one at a time\n");
    myusage ();
  }

  if (do_S && (do_A || do_s)) {
    fprintf (stderr, "Cluster stock table around s_w_id or s_i_id?\n");
    myusage ();
  }

  if (do_C && (do_A || do_c)) {
```

```
    fprintf (stderr, "Cluster cust table around c_w_id or c_id?\n");
    myusage ();
  }

  if (eware <= 0)
    eware = scale;
  if (ecid <= 0)
    ecid = CUSTFAC;
  if (eitem <= 0)
    eitem = STOCFAC;

  if (do_C) {
    if ((bcid  < 1) || (bcid > CUSTFAC)) {
      fprintf (stderr, "Invalid beginning cid number: '%d'\n", bcid);
      myusage ();
    }

    if ((ecid  < bcid) || (ecid > CUSTFAC)) {
      fprintf (stderr, "Invalid ending cid number: '%d'\n", ecid);
      myusage ();
    }
  }
  if (do_S) {
    if ((bitem  < 1) || (bitem > STOCFAC)) {
      fprintf (stderr, "Invalid beginning item number: '%d'\n", bitem);
      myusage ();
    }

    if ((eitem  < bitem) || (eitem > STOCFAC)) {
      fprintf (stderr, "Invalid ending item number: '%d'\n", eitem);
      myusage ();
    }
  }
  if (do_o) {
    if ((basename = getenv ("tpcc_bench")) == NULL)
    {
      fprintf (stderr, "$tpcc_bench is not set");
      myusage ();
    }
  }

  if ((bware  < 1) || (bware > scale)) {
    fprintf (stderr, "Invalid beginning warehouse number: '%d'\n", bware);
    myusage ();
  }

  if ((eware  < bware) || (eware > scale)) {
    fprintf (stderr, "Invalid ending warehouse number: '%d'\n", eware);
    myusage ();
  }

  if (gen && do_o) {
    if ((olfp = fopen (olfname, "w")) == NULL) {
      fprintf (stderr, "Can't open '%s' for writing order lines\n", olfname);
      myusage ();
    }


  }

/*---------------------------------------------------------------+
 | Prepare to insert into database.                              |
 +---------------------------------------------------------------*/

  sysdate (sdate);
  if (!gen) {

    /* log on to Oracle */

    OCIInitialize(OCI_DEFAULT|OCI_OBJECT,(dvoid *)0,0,0,0);
    OCIEnvInit(&tpcenv, OCI_DEFAULT, 0, (dvoid **)0);
    OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsrv, OCI_HTYPE_SERVER, 0 ,
(dvoid **)0);
    OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&errhp, OCI_HTYPE_ERROR, 0 ,
(dvoid **)0);
    OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsvc, OCI_HTYPE_SVCCTX,
0 , (dvoid **)0);
    OCIServerAttach(tpcsrv, errhp, (text *)0,0,OCI_DEFAULT);
    OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX, (dvoid *)tpcsrv,
            (ub4)0,OCI_ATTR_SERVER, errhp);
    OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr, OCI_HTYPE_SESSION,
0 , (dvoid **)0);
    OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)uid,
            (ub4)strlen(uid),OCI_ATTR_USERNAME, errhp);
    OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)pwd,
(ub4)strlen(pwd),
```

```
        OCI_ATTR_PASSWORD, errhp);
    OCIERROR(errhp, OCISessionBegin(tpcsvc, errhp, tpcusr, OCI_CRED_RDBMS,
OCI_DEFAULT));

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

    fprintf (stderr, "\nConnected to Oracle userid '%s/%s'.\n", uid, pwd);

    /* open cursors and parse statement */
    if (do_A || do_w)  {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curw),
OCI_HTYPE_STMT, 0, (dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curw, errhp, (text *)SQLTXTW,
            strlen((char *)SQLTXTW), (ub4) OCI_NTV_SYNTAX, (ub4)
OCI_DEFAULT));
    }

    if (do_A || do_d) {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curd),
OCI_HTYPE_STMT, 0, (dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curd, errhp, (text *)SQLTXTD,
            strlen((char *)SQLTXTD), (ub4) OCI_NTV_SYNTAX, (ub4)
OCI_DEFAULT));
    }

    if (do_A || do_c || do_C) {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curc),
OCI_HTYPE_STMT, 0, (dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curc, errhp, (text *)SQLTXTC,
            strlen((char *)SQLTXTC), (ub4) OCI_NTV_SYNTAX, (ub4)
OCI_DEFAULT));
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curcs),
OCI_HTYPE_STMT, 0, (dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curcs, errhp, (text *)SQLTXTCQUERY,
            strlen((char *)SQLTXTCQUERY), (ub4) OCI_NTV_SYNTAX, (ub4)
OCI_DEFAULT));
    }

    if (do_A || do_h) {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curh),
OCI_HTYPE_STMT, 0, (dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curh, errhp, (text *)SQLTXTH,
            strlen((char *)SQLTXTH), (ub4) OCI_NTV_SYNTAX, (ub4)
OCI_DEFAULT));
    }

    if (do_A || do_s || do_S) {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curs),
OCI_HTYPE_STMT, 0, (dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curs, errhp, (text *)SQLTXTS,
            strlen((char *)SQLTXTS), (ub4) OCI_NTV_SYNTAX, (ub4)
OCI_DEFAULT));
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curss),
OCI_HTYPE_STMT, 0, (dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curss, errhp, (text *)SQLTXTSQUERY,
            strlen((char *)SQLTXTSQUERY), (ub4) OCI_NTV_SYNTAX, (ub4)
OCI_DEFAULT));
    }

    if (do_A || do_i) {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curi),
OCI_HTYPE_STMT, 0, (dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curi, errhp, (text *)SQLTXTI,
            strlen((char *)SQLTXTI), (ub4) OCI_NTV_SYNTAX, (ub4)
OCI_DEFAULT));
    }

    if (do_A || do_o) {
      int stat;
      char fname[160];
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curo1),
OCI_HTYPE_STMT, 0, (dvoid**)0));
      DISCARD strcpy(fname,basename);
      DISCARD strcat(fname, "/");
      DISCARD strcat(fname, "benchrun/blocks/load_ordordl.sql");
      stat = sqlfile(fname, stmbuf);
      if (!stat)
      {
        fprintf (stderr, "unable to open %s \n",fname);
        quit();
        exit(1);
      }
      OCIERROR(errhp,OCIStmtPrepare(curo1, errhp, stmbuf,
            strlen((char *)stmbuf), (ub4) OCI_NTV_SYNTAX, (ub4)
OCI_DEFAULT));
```

```
    }

    if (do_A || do_n) {
      OCIERROR(errhp,OCIHandleAlloc(tpcenv,(dvoid **)(&curno),
OCI_HTYPE_STMT, 0, (dvoid**)0));
      OCIERROR(errhp,OCIStmtPrepare(curno, errhp, (text *)SQLTXTNO,
            strlen((char *)SQLTXTNO), (ub4) OCI_NTV_SYNTAX, (ub4)
OCI_DEFAULT));
    }

    /* bind variables */

    /* warehouse */

    if (do_A || do_w) {
      OCIERROR(errhp, OCIBindByName(curw, &w_id_bp, errhp, (text *)(":w_id"),
strlen((":w_id")),
            (ub1 *)&(w_id), sizeof(w_id), SQLT_INT, (dvoid *) 0, (ub2 *)0, (ub2
*)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_name_bp, errhp,(text
*)":w_name", strlen(":w_name"),
            (ub1 *)w_name, 11, SQLT_STR, (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_street1_bp, errhp, (text
*)":w_street_1",
                  strlen(":w_street_1"), (ub1 *)w_street_1, 21, SQLT_STR,
                  (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_street2_bp, errhp, (text
*)":w_street_2",
                  strlen(":w_street_2"), (ub1 *)w_street_2, 21, SQLT_STR,
                  (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_city_bp, errhp, (text
*)":w_city",
                  strlen(":w_city"), (ub1 *)w_city, 21, SQLT_STR,
                  (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_state_bp, errhp, (text
*)":w_state",
                  strlen(":w_state"), (ub1 *)w_state, 2, SQLT_CHR,
                  (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_zip_bp, errhp, (text
*)":w_zip",
                  strlen(":w_zip"), (ub1 *)w_zip, 9, SQLT_CHR,
                  (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curw, &w_tax_bp, errhp, (text
*)":w_tax",
                  strlen(":w_tax"), (ub1 *) & w_tax, sizeof(w_tax), SQLT_FLT,
                  (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
    }

    /* district */

    if (do_A || do_d) {
      OCIERROR(errhp, OCIBindByName(curd, &d_id_bp, errhp, (text *)":d_id",
                  strlen(":d_id"), (ub1 *)d_id, sizeof(int), SQLT_INT,
                  (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_w_id_bp, errhp, (text
*)":d_w_id",
                  strlen(":d_w_id"), (ub1 *)d_w_id, sizeof(int), SQLT_INT,
                  (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_name_bp, errhp, (text
*)":d_name",
                  strlen(":d_name"), (ub1 *)d_name, 11, SQLT_STR,
                  (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curd, &d_street1_bp, errhp, (text
*)":d_street_1",
                  strlen(":d_street_1"), (ub1 *)d_street_1, 21, SQLT_STR,
```

```
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curd, &d_street2_bp, errhp, (text
*)":d_street_2",
                strlen(":d_street_2"), (ub1 *)d_street_2, 21, SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curd, &d_city_bp, errhp, (text *)":d_city",
                strlen(":d_city"), (ub1 *)d_city, 21, SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curd, &d_state_bp, errhp, (text
*)":d_state",
            strlen(":d_state"), (ub1 *)d_state, 2, SQLT_CHR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curd, &d_zip_bp, errhp, (text *)":d_zip",
            strlen(":d_zip"), (ub1 *)d_zip, 9, SQLT_CHR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curd, &d_tax_bp, errhp, (text *)":d_tax",
            strlen(":d_tax"), (ub1 *)d_tax, sizeof(float), SQLT_FLT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
  }

  /* customer */

  if (do_A || do_c || do_C) {
    OCIERROR(errhp, OCIBindByName(curcs, &s_c_id_bp, errhp, (text
*)":s_c_id",
            strlen(":s_c_id"), (ub1 *)&s_c_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curcs, &s_c_w_id_bp, errhp, (text
*)":s_c_w_id",
            strlen(":s_c_w_id"), (ub1 *)&s_c_w_id, sizeof(int), SQLT_INT,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curcs, &s_c_d_id_bp, errhp, (text
*)":s_c_d_id",
            strlen(":s_c_d_id"), (ub1 *)&s_c_d_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


OCIDefineByPos(curcs,&s_c_ret_bp,errhp,1,&s_c_count,sizeof(int),SQLT_INT,\
            0,0,0,OCI_DEFAULT);

    OCIERROR(errhp, OCIBindByName(curc, &c_id_bp, errhp, (text *)":c_id",
            strlen(":c_id"), (ub1 *)c_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_d_id_bp, errhp, (text
*)":c_d_id",
            strlen(":c_d_id"), (ub1 *)c_d_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_w_id_bp, errhp, (text
*)":c_w_id",
            strlen(":c_w_id"), (ub1 *)c_w_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_first_bp, errhp, (text *)":c_first",
                strlen(":c_first"), (ub1 *)c_first, 17, SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_last_bp, errhp, (text *)":c_last",
                strlen(":c_last"), (ub1 *)c_last, 17, SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_street1_bp, errhp, (text
*)":c_street_1",
            strlen(":c_street_1"), (ub1 *)c_street_1, 21, SQLT_STR,
```

```
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_street2_bp, errhp, (text
*)":c_street_2",
            strlen(":c_street_2"), (ub1 *)c_street_2, 21, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_city_bp, errhp, (text *)":c_city",
            strlen(":c_city"), (ub1 *)c_city, 21, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_state_bp, errhp, (text
*)":c_state",
            strlen(":c_state"), (ub1 *)c_state, 2, SQLT_CHR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_zip_bp, errhp, (text *)":c_zip",
            strlen(":c_zip"), (ub1 *)c_zip, 9, SQLT_CHR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_phone_bp, errhp, (text
*)":c_phone",
            strlen(":c_phone"), (ub1 *)c_phone, 16, SQLT_CHR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_credit_bp, errhp, (text
*)":c_credit",
            strlen(":c_credit"), (ub1 *)c_credit, 2, SQLT_CHR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_discount_bp, errhp, (text
*)":c_discount",
            strlen(":c_discount"), (ub1 *)c_discount, sizeof(float), SQLT_FLT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curc, &c_data_bp, errhp, (text
*)":c_data",
            strlen(":c_data"), (ub1 *)c_data, 501, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
  }

  /* item */

  if (do_A || do_i) {
    OCIERROR(errhp, OCIBindByName(curi, &i_id_bp, errhp, (text *)":i_id",
            strlen(":i_id"), (ub1 *)i_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curi, &i_im_id_bp, errhp, (text
*)":i_im_id",
            strlen(":i_im_id"), (ub1 *)i_im_id, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curi, &i_name_bp, errhp, (text
*)":i_name",
            strlen(":i_name"), (ub1 *)i_name, 25, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curi, &i_price_bp, errhp, (text
*)":i_price",
            strlen(":i_price"), (ub1 *)i_price, sizeof(int), SQLT_INT,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curi, &i_data_bp, errhp, (text *)":i_data",
            strlen(":i_data"), (ub1 *)i_data, 51, SQLT_STR,
            (dvoid *) 0, (ub2 *)0, (ub2 *)0,
            (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
  }

  /* stock */

  if (do_A || do_s || do_S) {
    OCIERROR(errhp, OCIBindByName(curss, &s_s_i_id_bp, errhp, (text
```

```
*)":s_s_i_id",
                strlen(":s_s_i_id"), (ub1 *)&s_s_i_id, sizeof(int), SQLT_INT,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curss, &s_s_w_id_bp, errhp, (text
*)":s_s_w_id",
                strlen(":s_s_w_id"), (ub1 *)&s_s_w_id, sizeof(int), SQLT_INT,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIDefineByPos(curss,&s_s_ret_bp,errhp,1,&s_s_count,sizeof(int),SQLT_INT,\
                0,0,0,OCI_DEFAULT);

        OCIERROR(errhp, OCIBindByName(curs, &s_i_id_bp, errhp, (text *)":s_i_id",
                strlen(":s_i_id"), (ub1 *)s_i_id, sizeof(int), SQLT_INT,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curs, &s_w_id_bp, errhp, (text
*)":s_w_id",
                strlen(":s_w_id"), (ub1 *)s_w_id, sizeof(int), SQLT_INT,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curs, &s_quantity_bp, errhp, (text
*)":s_quantity",
                strlen(":s_quantity"), (ub1 *)s_quantity, sizeof(int), SQLT_INT,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curs, &s_dist_01_bp, errhp, (text
*)":s_dist_01",
                strlen(":s_dist_01"), (ub1 *)s_dist_01, 25, SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


        OCIERROR(errhp, OCIBindByName(curs, &s_dist_02_bp, errhp, (text
*)":s_dist_02",
                strlen(":s_dist_02"), (ub1 *)s_dist_02, 25, SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curs, &s_dist_03_bp, errhp, (text
*)":s_dist_03",
                strlen(":s_dist_03"), (ub1 *)s_dist_03, 25, SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


        OCIERROR(errhp, OCIBindByName(curs, &s_dist_04_bp, errhp, (text
*)":s_dist_04",
                strlen(":s_dist_04"), (ub1 *)s_dist_04, 25, SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


        OCIERROR(errhp, OCIBindByName(curs, &s_dist_05_bp, errhp, (text
*)":s_dist_05",
                strlen(":s_dist_05"), (ub1 *)s_dist_05, 25, SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


        OCIERROR(errhp, OCIBindByName(curs, &s_dist_06_bp, errhp, (text
*)":s_dist_06",
                strlen(":s_dist_06"), (ub1 *)s_dist_06, 25, SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


        OCIERROR(errhp, OCIBindByName(curs, &s_dist_07_bp, errhp, (text
*)":s_dist_07",
                strlen(":s_dist_07"), (ub1 *)s_dist_07, 25, SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


        OCIERROR(errhp, OCIBindByName(curs, &s_dist_08_bp, errhp, (text
*)":s_dist_08",
                strlen(":s_dist_08"), (ub1 *)s_dist_08, 25, SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
```

```
        OCIERROR(errhp, OCIBindByName(curs, &s_dist_09_bp, errhp, (text
*)":s_dist_09",
                strlen(":s_dist_09"), (ub1 *)s_dist_09, 25, SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


        OCIERROR(errhp, OCIBindByName(curs, &s_dist_10_bp, errhp, (text
*)":s_dist_10",
                strlen(":s_dist_10"), (ub1 *)s_dist_10, 25, SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curs, &s_data_bp, errhp, (text *)":s_data",
                strlen(":s_data"), (ub1 *)s_data, 51, SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
    }

    /* history */

    if (do_A || do_h) {
        OCIERROR(errhp, OCIBindByName(curh, &h_c_id_bp, errhp, (text
*)":h_c_id",
                strlen(":h_c_id"), (ub1 *)h_c_id, sizeof(int), SQLT_INT,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curh, &h_c_d_id_bp, errhp, (text
*)":h_c_d_id",
                strlen(":h_c_d_id"), (ub1 *)h_d_id, sizeof(int), SQLT_INT,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curh, &h_c_w_id_bp, errhp, (text
*)":h_c_w_id",
                strlen(":h_c_w_id"), (ub1 *)h_w_id, sizeof(int), SQLT_INT,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curh, &h_d_id_bp, errhp, (text
*)":h_d_id",
                strlen(":h_d_id"), (ub1 *)h_d_id, sizeof(int), SQLT_INT,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curh, &h_w_id_bp, errhp, (text
*)":h_w_id",
                strlen(":h_w_id"), (ub1 *)h_w_id, sizeof(int), SQLT_INT,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curh, &h_data_bp, errhp, (text
*)":h_data",
                strlen(":h_data"), (ub1 *)h_data, 25, SQLT_STR,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));


    }

    /* order and order_line (delivered) */

    if (do_A || do_o) {

      for (i = 0; i < ORDEARR; i++ ) {
        o_id_len[i] = sizeof(int);
        o_d_id_len[i] = sizeof(int);
        o_w_id_len[i] = sizeof(int);
        o_c_id_len[i] = sizeof(int);
        o_carrier_id_len[i] = sizeof(int);
        o_ol_cnt_len[i] = sizeof(int);
      }

        OCIERROR(errhp, OCIBindByName(curo1, &ol_o_id_bp, errhp, (text
*)":ol_o_id",
                strlen(":ol_o_id"), (ub1 *)ol_o_id, sizeof(int), SQLT_INT,
                (dvoid *) 0, (ub2 *)ol_o_id_len, (ub2 *)0,
                (ub4) 15*ORDEARR, (ub4 *)&ol_o_id_clen, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &ol_d_id_bp, errhp, (text
*)":ol_d_id",
                strlen(":ol_d_id"), (ub1 *)ol_d_id, sizeof(int), SQLT_INT,
                (dvoid *) 0, (ub2 *)ol_d_id_len, (ub2 *)0,
                (ub4) 15*ORDEARR, (ub4 *) &ol_d_id_clen, (ub4)
OCI_DEFAULT));
```

```
	OCIERROR(errhp, OCIBindByName(curo1, &ol_w_id_bp, errhp, (text
*)":ol_w_id",
		strlen(":ol_w_id"), (ub1 *)ol_w_id, sizeof(int), SQLT_INT,
		(dvoid *) 0, (ub2 *)ol_w_id_len, (ub2 *)0,
		(ub4) 15*ORDEARR, (ub4 *) &ol_w_id_clen, (ub4)
OCI_DEFAULT));

	OCIERROR(errhp, OCIBindByName(curo1, &ol_number_bp, errhp, (text
*)":ol_number",
		strlen(":ol_number"), (ub1 *)ol_number, sizeof(int), SQLT_INT,
		(dvoid *) 0, (ub2 *)ol_number_len, (ub2 *)0,
		(ub4) 15*ORDEARR, (ub4 *) &ol_number_clen, (ub4)
OCI_DEFAULT));

	OCIERROR(errhp, OCIBindByName(curo1, &ol_i_id_bp, errhp, (text
*)":ol_i_id",
		strlen(":ol_i_id"), (ub1 *)ol_i_id, sizeof(int), SQLT_INT,
		(dvoid *) 0, (ub2 *)ol_i_id_len, (ub2 *)0,
		(ub4) 15*ORDEARR, (ub4 *) &ol_i_id_clen, (ub4) OCI_DEFAULT));

	OCIERROR(errhp, OCIBindByName(curo1, &ol_supply_w_id_bp, errhp, (text
*)":ol_supply_w_id",
		strlen(":ol_supply_w_id"), (ub1 *)ol_supply_w_id, sizeof(int),
SQLT_INT,
		(dvoid *) 0, (ub2 *)ol_supply_w_id_len, (ub2 *)0,
		(ub4) 15*ORDEARR, (ub4 *) &ol_supply_w_id_clen, (ub4)
OCI_DEFAULT));

	OCIERROR(errhp, OCIBindByName(curo1, &ol_dist_info_bp, errhp, (text
*)":ol_dist_info",
		strlen(":ol_dist_info"), (ub1 *)ol_dist_info, 24, SQLT_CHR,
		(dvoid *) 0, (ub2 *)ol_dist_info_len, (ub2 *)0,
		(ub4) 15*ORDEARR, (ub4 *) &ol_dist_info_clen, (ub4)
OCI_DEFAULT));

	OCIERROR(errhp, OCIBindByName(curo1, &ol_amount_bp, errhp, (text
*)":ol_amount",
		strlen(":ol_amount"), (ub1 *)ol_amount, sizeof(int), SQLT_INT,
		(dvoid *) 0, (ub2 *)ol_amount_len, (ub2 *)0,
		(ub4) 15*ORDEARR, (ub4 *) &ol_amount_clen, (ub4)
OCI_DEFAULT));

	OCIERROR(errhp, OCIBindByName(curo1, &o_id_bp, errhp, (text *)":o_id",
		strlen(":o_id"), (ub1 *)o_id, sizeof(int), SQLT_INT,
		(dvoid *) 0, (ub2 *)o_id_len, (ub2 *)0,
		(ub4) ORDEARR, (ub4 *) &o_id_clen, (ub4) OCI_DEFAULT));

	OCIERROR(errhp, OCIBindByName(curo1, &o_d_id_bp, errhp, (text
*)":o_d_id",
		strlen(":o_d_id"), (ub1 *)o_d_id, sizeof(int), SQLT_INT,
		(dvoid *) 0, (ub2 *)o_d_id_len, (ub2 *)0,
		(ub4) ORDEARR, (ub4 *) &o_d_id_clen, (ub4) OCI_DEFAULT));

	OCIERROR(errhp, OCIBindByName(curo1, &o_w_id_bp, errhp, (text
*)":o_w_id",
		strlen(":o_w_id"), (ub1 *)o_w_id, sizeof(int), SQLT_INT,
		(dvoid *) 0, (ub2 *)o_w_id_len, (ub2 *)0,
		(ub4) ORDEARR, (ub4 *) &o_w_id_clen, (ub4) OCI_DEFAULT));

	OCIERROR(errhp, OCIBindByName(curo1, &o_c_id_bp, errhp, (text
*)":o_c_id",
		strlen(":o_c_id"), (ub1 *)o_c_id, sizeof(int), SQLT_INT,
		(dvoid *) 0, (ub2 *)o_c_id_len, (ub2 *)0,
		(ub4) ORDEARR, (ub4 *) &o_c_id_clen, (ub4) OCI_DEFAULT));

	OCIERROR(errhp, OCIBindByName(curo1, &o_carrier_id_bp, errhp, (text
*)":o_carrier_id",
		strlen(":o_carrier_id"), (ub1 *)o_carrier_id, sizeof(int), SQLT_INT,
		(dvoid *) 0, (ub2 *)o_carrier_id_len, (ub2 *)0,
		(ub4) ORDEARR, (ub4 *) &o_carrier_id_clen, (ub4)
OCI_DEFAULT));

	OCIERROR(errhp, OCIBindByName(curo1, &o_ol_cnt_bp, errhp, (text
*)":o_ol_cnt",
		strlen(":o_ol_cnt"), (ub1 *)o_ol_cnt, sizeof(int), SQLT_INT,
		(dvoid *) 0, (ub2 *)o_ol_cnt_len, (ub2 *)0,
		(ub4) ORDEARR, (ub4 *) &o_ol_cnt_clen, (ub4) OCI_DEFAULT));

	OCIERROR(errhp, OCIBindByName(curo1, &o_ocnt_bp, errhp, (text
*)":order_rows",
		strlen(":order_rows"), (ub1 *)&o_cnt, sizeof(int), SQLT_INT,
		(dvoid *) 0, (ub2 *)0, (ub2 *)0,
		(ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

	OCIERROR(errhp, OCIBindByName(curo1, &o_olcnt_bp, errhp, (text
```

```
*)":ordl_rows",
                strlen(":ordl_rows"), (ub1 *)&ol_cnt, sizeof(int), SQLT_INT,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
    }

    /* new order */

    if (do_A || do_n) {
      OCIERROR(errhp, OCIBindByName(curno, &no_o_id_bp, errhp, (text
*)":no_o_id",
                strlen(":no_o_id"), (ub1 *)no_o_id, sizeof(int), SQLT_INT,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curno, &no_d_id_bp, errhp, (text
*)":no_d_id",
                strlen(":no_d_id"), (ub1 *)no_d_id, sizeof(int), SQLT_INT,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

      OCIERROR(errhp, OCIBindByName(curno, &no_w_id_bp, errhp, (text
*)":no_w_id",
                strlen(":no_w_id"), (ub1 *)no_w_id, sizeof(int), SQLT_INT,
                (dvoid *) 0, (ub2 *)0, (ub2 *)0,
                (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
    }
  }

/*---------------------------------------------------------------+
 | Initialize random number generator                            |
 +---------------------------------------------------------------*/

  srand (SEED);
#ifndef ORA_NT
  srand48 (SEED);
#endif
  initperm ();


/*---------------------------------------------------------------+
 | Load the WAREHOUSE table.                                     |
 +---------------------------------------------------------------*/

if (do_A || do_w) {
  nrows = eware - bware + 1;

  fprintf (stderr, "Loading/generating warehouse: w%d - w%d (%d rows)\n",
        bware, eware, nrows);

  begin_time = gettime ();
  begin_cpu = getcpu ();

  for (loop = bware; loop <= eware; loop++) {

    w_tax = (float) ((lrand48 () % 2001) * 0.0001);
    randstr (w_name, 6, 10);
    randstr (w_street_1, 10, 20);
    randstr (w_street_2, 10, 20);
    randstr (w_city, 10, 20);
    randstr (str2, 2, 2);
    randnum (num9, 9);
    num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';

    if (gen) {
      printf ("%d 30000000 %6.4f %s %s %s %s %s %s\n", loop, w_tax,
            w_name, w_street_1, w_street_2, w_city, str2, num9);
      fflush (stdout);
    }
    else {
      w_id = loop;
      strncpy (w_state, str2, 2);
      strncpy (w_zip, num9, 9);

          status = OCIStmtExecute(tpcsvc, curw, errhp, (ub4) 1, (ub4) 0,
              (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
              (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
          if (status != OCI_SUCCESS) {
            fprintf (stderr, "Error at ware %d\n", loop);
            OCIERROR(errhp, status);
            quit ();
        exit (1);
          }
      }
  }
```

```
    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
            nrows, end_time - begin_time, end_cpu - begin_cpu);
  }

/*---------------------------------------------------------------+
 | Load the DISTRICT table.                                      |
 +---------------------------------------------------------------*/

  if (do_A || do_d) {
    nrows = (eware - bware + 1) * DISTFAC;

    fprintf (stderr, "Loading/generating district: w%d - w%d (%d rows)\n",
            bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    dwid = bware - 1;

    for (row = 0; row < nrows; ) {
      dwid++;

      for (i = 0; i < DISTARR; i++, row++) {
        d_tax[i] = (float) ((lrand48 () % 2001) * 0.0001);
        randstr (d_name[i], 6, 10);
        randstr (d_street_1[i], 10, 20);
        randstr (d_street_2[i], 10, 20);
        randstr (d_city[i], 10, 20);
        randstr (str2, 2, 2);
        randnum (num9, 9);
        num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';

        if (gen) {
          printf ("%d %d 3000000 %6.4f 3001 %s %s %s %s %s %s\n",
                i + 1, dwid, d_tax[i], d_name[i], d_street_1[i],
                        d_street_2[i], d_city[i], str2, num9 );
        }
        else {
          d_id[i] = i + 1;
          d_w_id[i] = dwid;
          strncpy (d_state[i], str2, 2);
          strncpy (d_zip[i], num9, 9);
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        status = OCIStmtExecute(tpcsvc, curd, errhp, (ub4) DISTARR, (ub4) 0,
                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
                fprintf (stderr, "Aborted at ware %d, dist 1\n", dwid);
          OCIERROR(errhp, status);
          quit ();
          exit (1);
        }
      }
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
            nrows, end_time - begin_time, end_cpu - begin_cpu);
  }

/*---------------------------------------------------------------+
 | Load the CUSTOMER table.                                      |
 +---------------------------------------------------------------*/

  if (do_A || do_c) {

    nrows = (eware - bware + 1) * CUSTFAC * DISTFAC;

    fprintf (stderr, "Loading/generating customer: w%d - w%d (%d rows)\n   ",
            bware, eware, nrows);

    if (getenv("tpcc_hash_overflow")) {
      fprintf(stderr,"Hash overflow is enabled\n");
      OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0,
(dvoid**)0);
```

```
    sprintf ((char *) stmbuf, SQLTXTENHA);
    OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
                OCI_NTV_SYNTAX, OCI_DEFAULT);
    OCIERROR(errhp,OCIStmtExecute(tpcsvc, curi,
errhp,1,0,0,0,OCI_DEFAULT));
    OCIHandleFree(curi, OCI_HTYPE_STMT);
    fprintf (stderr,"Customer loaded for horizontal partitioning\n");
   }
   else
   {
    fprintf (stderr,"Customer not loaded for horizontal partitioning\n");
   }
   begin_time = gettime ();
   begin_cpu = getcpu ();

   s_c_id = 1;
   s_c_d_id = 1;
   s_c_w_id = bware;

   while (s_c_w_id <= eware) {
     status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4) 0,
           (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
           (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
       if (status != OCI_SUCCESS) {
           OCIERROR(errhp, status);
           quit ();
           exit (1);
       }

       if (s_c_count == 0) {
         s_c_w_id--;
         break;
       }
       else s_c_w_id++;
   }

   if (s_c_w_id < bware ) s_c_w_id = bware;
   else {
    if (s_c_w_id > eware ) s_c_w_id = eware;
    while (s_c_d_id <= DISTFAC) {
      status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4) 0,
           (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
           (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
       if (status != OCI_SUCCESS) {
                fprintf (stderr, "Select failed\n");
           OCIERROR(errhp, status);
           quit ();
           exit (1);
       }
       if (s_c_count == 0) {
         s_c_d_id--;
         break;
       }
       else s_c_d_id++;
    }
    if (s_c_d_id > DISTFAC) s_c_d_id = DISTFAC;

    while (s_c_id <= CUSTFAC) {
      status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4) 0,
           (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
           (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
       if (status != OCI_SUCCESS) {
           OCIERROR(errhp, status);
           quit ();
           exit (1);
       }
       if (s_c_count == 0) break;
       else s_c_id++;
    }
   }
   if (s_c_id > CUSTFAC) {
    if (s_c_d_id == DISTFAC) {
      s_c_d_id=1;
      s_c_w_id++;
    } else {
          s_c_d_id++;
    }
    s_c_id=1;
   }

   fprintf (stderr, "start at wid: %d, did: %d, cid: %d\n   ",s_c_w_id, s_c_d_id,
s_c_id);
   cid = s_c_id - 1;
   cdid = s_c_d_id;
   cwid = s_c_w_id;
   nrows = (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;
            cwid++;                 /* shift ware cycle */
          }
        }
        c_id[i] = cid;
        c_d_id[i] = cdid;
        c_w_id[i] = cwid;
        if (cid <= 1000)
          randlastname (c_last[i], cid - 1);
        else
          randlastname (c_last[i], NURand (255, 0, 999, CNUM1));
        c_credit[i][1] = 'C';
        if (lrand48 () % 10)
          c_credit[i][0] = 'G';
        else
          c_credit[i][0] = 'B';
        c_discount[i] = (float)((lrand48 () % 5001) * 0.0001);
        randstr (c_first[i], 8, 16);
        randstr (c_street_1[i], 10, 20);
        randstr (c_street_2[i], 10, 20);
        randstr (c_city[i], 10, 20);
        randstr (str2, 2, 2);
        randnum (num9, 9);
        num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';
        randnum (num16, 16);
        randstr (c_data[i], 300, 500);

        if (gen) {
          printf ("%d %d %d %s OE %s %s %s %s %s %s %s %cC
5000000 %6.4f -1000 1000 1 0 %s\n",
                  cid, cdid, cwid, c_first[i], c_last[i],
                  c_street_1[i], c_street_2[i], c_city[i], str2, num9,
                  num16, sdate, c_credit[i][0], c_discount[i], c_data[i]);
        }
        else {
          strncpy (c_state[i], str2, 2);
          strncpy (c_zip[i], num9, 9);
          strncpy (c_phone[i], num16, 16);
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        status = OCIStmtExecute(tpcsvc, curc, errhp, (ub4) 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, SQLTXTDIHA);
      OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
```

```
                OCI_NTV_SYNTAX, OCI_DEFAULT);
      OCIERROR(errhp,OCIStmtExecute(tpcsvc, curi,
errhp,1,0,0,0,OCI_DEFAULT));
      OCIHandleFree(curi, OCI_HTYPE_STMT);
    }
  }


/*---------------------------------------------------------------+
 | Load the CUSTOMER table (cluster around c_id)                 |
 +---------------------------------------------------------------*/

  if (do_C) {

    srand (bcid);
#ifndef ORA_NT
    srand48 (bcid);
#endif

    nrows = (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, eware, nrows);

    if (getenv("tpcc_hash_overflow")) {
      fprintf(stderr,"Hash overflow is enabled\n");
      OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0,
(dvoid**)0);
      sprintf ((char *) stmbuf, SQLTXTENHA);
      OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
                OCI_NTV_SYNTAX, OCI_DEFAULT);
      OCIERROR(errhp,OCIStmtExecute(tpcsvc, curi,
errhp,1,0,0,0,OCI_DEFAULT));
      OCIHandleFree(curi, OCI_HTYPE_STMT);
      fprintf (stderr,"Customer loaded for horizontal partitioning\n");
    }
    else
    {
      fprintf (stderr,"Customer not loaded for horizontal partitioning\n");
    }
    begin_time = gettime ();
    begin_cpu = getcpu ();

    s_c_id = bcid;
    s_c_d_id = 1;
    s_c_w_id = bware;

    while (s_c_id <= ecid) {
      status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4) 0,
            (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
            (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
      if (status != OCI_SUCCESS) {
            OCIERROR(errhp, status);
            quit ();
            exit (1);
      }

      if (s_c_count == 0) {
        s_c_id--;
        break;
      }
      else s_c_id++;
    }

    if (s_c_id < bcid ) s_c_id = bcid;
    else {
      if (s_c_id > ecid ) s_c_id = ecid;
      while (s_c_w_id <= eware) {
        status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4) 0,
              (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
              (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
                    fprintf (stderr, "Select failed\n");
              OCIERROR(errhp, status);
              quit ();
              exit (1);
        }
        if (s_c_count == 0) {
          s_c_w_id--;
          break;
        }
        else s_c_w_id++;
      }
      if (s_c_w_id > eware) s_c_w_id = eware;
      else if (s_c_w_id < bware) s_c_w_id = bware;
```

```
  while (s_c_d_id <= DISTFAC) {
    status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4) 0,
          (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
          (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
      if (status != OCI_SUCCESS) {
          OCIERROR(errhp, status);
          quit ();
          exit (1);
      }
      if (s_c_count == 0) break;
      else s_c_d_id++;
  }
 }

 if (s_c_d_id > DISTFAC) {
  s_c_d_id=1;
  if (s_c_w_id==eware) {
   s_c_w_id=bware;
   s_c_id++;
  }
  else s_c_w_id++;
 }

 fprintf (stderr, "start at cid: %d, wid: %d, did: %d\n   ",s_c_id, s_c_w_id,
s_c_d_id);
 cid = s_c_id;
 cdid = s_c_d_id-1;
 cwid = s_c_w_id;
 nrows = (ecid - s_c_id + 1) * (eware - 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) {       /* cycle dist id */
       cdid = 1;               /* cheap mod */
       cwid++;                 /* shift dist cycle */
       if (cwid > eware) {
         cwid = bware;             /* shift ware cycle */
         cid++;
       }
     }
     c_id[i] = cid;
     c_d_id[i] = cdid;
     c_w_id[i] = cwid;
     if (cid <= 1000)
       randlastname (c_last[i], cid - 1);
     else
       randlastname (c_last[i], NURand (255, 0, 999, CNUM1));
     c_credit[i][1] = 'C';
     if (lrand48 () % 10)
       c_credit[i][0] = 'G';
     else
       c_credit[i][0] = 'B';
     c_discount[i] = (float)((lrand48 () % 5001) * 0.0001);
     randstr (c_first[i], 8, 16);
     randstr (c_street_1[i], 10, 20);
     randstr (c_street_2[i], 10, 20);
     randstr (c_city[i], 10, 20);
     randstr (str2, 2, 2);
     randnum (num9, 9);
     num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';
     randnum (num16, 16);
     randstr (c_data[i], 300, 500);

     if (gen) {
       printf ("%d %d %d %s OE %s %s %s %s %s %s %s %s %cC
5000000 %6.4f -1000 1000 1 0 %s\n",
           cid, cdid, cwid, c_first[i], c_last[i],
           c_street_1[i], c_street_2[i], c_city[i], str2, num9,
           num16, sdate, c_credit[i][0], c_discount[i], c_data[i]);
     }
     else {
       strncpy (c_state[i], str2, 2);
       strncpy (c_zip[i], num9, 9);
       strncpy (c_phone[i], num16, 16);
     }
   }

   if (gen) {
     fflush (stdout);
   }
   else {
```

```
        status = OCIStmtExecute(tpcsvc, curc, errhp, (ub4) i, (ub4) 0,
                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);

        if (status != OCI_SUCCESS) {
                    fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
                c_w_id[0], c_d_id[0], c_id[0]);
            OCIERROR(errhp, status);
            quit ();
            exit (1);
        }
      }

      if ((++loopcount) % 50)
        fprintf (stderr, ".");
      else
        fprintf (stderr, " %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, SQLTXTDIHA);
      OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
              OCI_NTV_SYNTAX, OCI_DEFAULT);
      OCIERROR(errhp,OCIStmtExecute(tpcsvc, curi,
errhp,1,0,0,0,OCI_DEFAULT));
      OCIHandleFree(curi, OCI_HTYPE_STMT);
    }
  }


/*---------------------------------------------------------------+
 | Load the ITEM table.                                          |
 +---------------------------------------------------------------*/

  if (do_A || do_i) {
    nrows = ITEMFAC;

    fprintf (stderr, "Loading/generating item: (%d rows)\n   ", nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    loopcount = 0;

    for (row = 0; row < nrows; ) {
      for (i = 0; i < ITEMARR; i++, row++) {
        i_im_id[i] = (lrand48 () % 10000) + 1;
        i_price[i] = ((lrand48 () % 9901) + 100);
        randstr (i_name[i], 14, 24);
        randdatastr (i_data[i], 26, 50);

        if (gen) {
          printf ("%d %d %s %d %s\n", row + 1, i_im_id[i], i_name[i],
                i_price[i], i_data[i]);
        }
        else {
          i_id[i] = row + 1;
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        status = OCIStmtExecute(tpcsvc, curi, errhp, (ub4) ITEMARR, (ub4) 0,
                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
                    fprintf (stderr, "Aborted at i_id %d\n", i_id[0]);
            OCIERROR(errhp, status);
            quit ();
            exit (1);
        }
      }

      if ((++loopcount) % 50)
        fprintf (stderr, ".");
      else
```

```
        fprintf (stderr, " %d rows committed\n   ", row);
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
            nrows, end_time - begin_time, end_cpu - begin_cpu);
  }


/*---------------------------------------------------------------+
 | Load the STOCK table.                                         |
 +---------------------------------------------------------------*/

  if (do_A || do_s) {

    nrows = (eware - bware + 1) * STOCFAC;

    fprintf (stderr, "Loading/generating stock: w%d - w%d (%d rows)\n   ",
          bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    s_s_i_id = 1;
    s_s_w_id = bware;

    while (s_s_w_id <= eware) {
      status = OCIStmtExecute(tpcsvc, curss, errhp, (ub4) 1, (ub4) 0,
            (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
            (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
            OCIERROR(errhp, status);
            quit ();
            exit (1);
        }
        if (s_s_count == 0) {
          s_s_w_id--;
          break;
        }
        else s_s_w_id++;
    }

if (s_s_w_id < bware ) s_s_w_id = bware;
else {
  if (s_s_w_id > eware) s_s_w_id = eware;
  while (s_s_i_id<=STOCFAC) {
    status = OCIStmtExecute(tpcsvc, curss, errhp, (ub4) 1, (ub4) 0,
            (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
            (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
      if (status != OCI_SUCCESS) {
            OCIERROR(errhp, status);
            quit ();
            exit (1);
      }
      if (s_s_count == 0) {
        break;
      }
      else s_s_i_id++;
  }
}
if (s_s_i_id > STOCFAC) {
  s_s_i_id=1;
  s_s_w_id++;
}

fprintf(stderr,"start at s_i_id: %d, s_w_id: %d\n   ", s_s_i_id, s_s_w_id);

sid = s_s_i_id - 1;
swid = s_s_w_id;
nrows = (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 %s 0 0 0 %s\n",
                sid, swid, s_quantity[i], s_dist_01[i], s_dist_02[i],
                s_dist_03[i], s_dist_04[i], s_dist_05[i], s_dist_06[i],
                s_dist_07[i], s_dist_08[i], s_dist_09[i], s_dist_10[i],
                s_data[i]);
        }
        else {
          s_i_id[i] = sid;
          s_w_id[i] = swid;
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
           /* Changed to STOCKARR to i - alex.ni */
        status = OCIStmtExecute(tpcsvc, curs, errhp, (ub4) i, (ub4) 0,
                  (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                  (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
                  fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
s_i_id[0]);
                OCIERROR(errhp, status);
                quit ();
                exit (1);
        }
      }

      if ((++loopcount) % 50)
        fprintf (stderr, ".");
      else
        fprintf (stderr, " %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);
  }

/*---------------------------------------------------------------+
 | Load the STOCK table (cluster around s_i_id).                 |
 +---------------------------------------------------------------*/

  if (do_S) {

    nrows = (eitem - bitem + 1) * (eware - bware + 1);

    fprintf (stderr, "Loading/generating stock: i%d - i%d, w%d - w%d (%d rows)\n   ",
          bitem, eitem, bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    s_s_i_id = bitem;
    s_s_w_id = bware;

    while (s_s_i_id <= eitem) {
      status = OCIStmtExecute(tpcsvc, curss, errhp, (ub4) 1, (ub4) 0,
            (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
            (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
            OCIERROR(errhp, status);
            quit ();
            exit (1);
        }
        if (s_s_count == 0) {
          s_s_i_id--;
          break;
        }
        else s_s_i_id++;
    }

    if (s_s_i_id < bitem ) s_s_i_id = bitem;
```

```
else {
  if (s_s_i_id > eitem) s_s_i_id = eitem;
  while (s_s_w_id <= eware) {
    status = OCIStmtExecute(tpcsvc, curss, errhp, (ub4) 1, (ub4) 0,
          (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
          (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
      if (status != OCI_SUCCESS) {
          OCIERROR(errhp, status);
          quit ();
          exit (1);
      }
      if (s_s_count == 0) {
        break;
      }
      else s_s_w_id++;
  }
}
if (s_s_w_id > eware) {
  s_s_w_id=bware;
  s_s_i_id++;
}

fprintf(stderr,"start at s_i_id: %d, s_w_id: %d\n   ", s_s_i_id, s_s_w_id);

sid = s_s_i_id;
swid = s_s_w_id - 1;
nrows = (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 > eware) {         /* cheap mod */
        swid = bware;
        sid++;
      }
      s_quantity[i] = (lrand48 () % 91) + 10;
      randstr (s_dist_01[i], 24, 24);
      randstr (s_dist_02[i], 24, 24);
      randstr (s_dist_03[i], 24, 24);
      randstr (s_dist_04[i], 24, 24);
      randstr (s_dist_05[i], 24, 24);
      randstr (s_dist_06[i], 24, 24);
      randstr (s_dist_07[i], 24, 24);
      randstr (s_dist_08[i], 24, 24);
      randstr (s_dist_09[i], 24, 24);
      randstr (s_dist_10[i], 24, 24);
      randdatastr (s_data[i], 26, 50);

      if (gen) {
        printf ("%d %d %d %s %s %s %s %s %s %s %s %s %s 0 0 0 %s\n",
              sid, swid, s_quantity[i], s_dist_01[i], s_dist_02[i],
              s_dist_03[i], s_dist_04[i], s_dist_05[i], s_dist_06[i],
              s_dist_07[i], s_dist_08[i], s_dist_09[i], s_dist_10[i],
              s_data[i]);
      }
      else {
        s_i_id[i] = sid;
        s_w_id[i] = swid;
      }
    }

    if (gen) {
      fflush (stdout);
    }
    else {
      status = OCIStmtExecute(tpcsvc, curs, errhp, (ub4) i, (ub4) 0,
              (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
              (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
      if (status != OCI_SUCCESS) {
          fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0], s_i_id[0]);
          OCIERROR(errhp, status);
          quit ();
          exit (1);
      }
    }

    if ((++loopcount) % 50)
      fprintf (stderr, ".");
    else
      fprintf (stderr, " %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);
  }

/*---------------------------------------------------------------+
 | Load the HISTORY table.                                       |
 +---------------------------------------------------------------*/

  if (do_A || do_h) {
    nrows = (eware - bware + 1) * HISTFAC;

    fprintf (stderr, "Loading/generating history: w%d - w%d (%d rows)\n   ",
            bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    cid = 0;
    cdid = 1;
    cwid = bware;
    loopcount = 0;

    for (row = 0; row < nrows; ) {
      for (i = 0; i < HISTARR; i++, row++) {
        cid++;
        if (cid > CUSTFAC) {        /* cycle cust id */
          cid = 1;                  /* cheap mod */
          cdid++;                   /* shift district cycle */
          if (cdid > DISTFAC) {
            cdid = 1;
            cwid++;                 /* shift warehouse cycle */
          }
        }
        h_c_id[i] = cid;
        h_d_id[i] = cdid;
        h_w_id[i] = cwid;
        randstr (h_data[i], 12, 24);
        if (gen) {
          printf ("%d %d %d %d %d %s 1000 %s\n", cid, cdid, cwid, cdid,
                  cwid, sdate, h_data[i]);
        }
      }
      if (gen) {
        fflush (stdout);
      }
      else {
        status = OCIStmtExecute(tpcsvc, curh, errhp, (ub4) HISTARR, (ub4) 0,
                  (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                  (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
                    fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
                  h_w_id[0], h_d_id[0], h_c_id[0]);
          OCIERROR(errhp, status);
          quit ();
          exit (1);
        }
      }

      if ((++loopcount) % 50)
        fprintf (stderr, ".");
      else
        fprintf (stderr, " %d rows committed\n   ", row);
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
            nrows, end_time - begin_time, end_cpu - begin_cpu);
  }

/*---------------------------------------------------------------+
 | Load the ORDERS and ORDER-LINE table.                         |
 +---------------------------------------------------------------*/

  if (do_A || do_o) {

    int batch_olcnt;

    nrows = (eware - bware + 1) * ORDEFAC * DISTFAC;

    fprintf (stderr, "Loading/generating orders and order-line: w%d - w%d (%d ord,
~%d ordl)\n   ",
            bware, eware, nrows, nrows * 10);
```

```
begin_time = gettime ();
begin_cpu = getcpu ();

cid = 0;
cdid = 1;
cwid = bware;
loopcount = 0;

for (row = 0; row < nrows; ) {

  batch_olcnt = 0;

  for (i = 0; i < ORDEARR; i++, row++) {
    cid++;
    if (cid > ORDEFAC) {        /* cycle cust id */
      cid = 1;                  /* cheap mod */
      cdid++;                   /* shift district cycle */
      if (cdid > DISTFAC) {
        cdid = 1;
        cwid++;                 /* shift warehouse cycle */
      }
    }
    o_carrier_id[i] = lrand48 () % 10 + 1;
    o_ol_cnt[i] = olcnt = lrand48 () % 11 + 5;

    if (gen) {
      if (cid < 2101) {
        printf ("%d %d %d %d %s %d %d 1\n", cid, cdid, cwid,
                randperm3000[cid - 1], sdate,o_carrier_id[i],
                o_ol_cnt[i]);
      }
      else {
              /* set carrierid to 11 instead of null */
        printf ("%d %d %d %d %s 11 %d 1\n", cid, cdid, cwid,
                randperm3000[cid - 1], sdate, o_ol_cnt[i]);
      }
    }
    else {
      o_id[i] = cid;
      o_d_id[i] = cdid;
      o_w_id[i] = cwid;
      o_c_id[i] = randperm3000[cid - 1];
      if (cid >= 2101 ) {
        o_carrier_id[i] = 11;
      }
    }

    for (j = 0; j < o_ol_cnt[i]; j++, batch_olcnt++ ) {
      ol_i_id[batch_olcnt] = sid = lrand48 () % 100000 + 1;
      if (cid < 2101)
        ol_amount[batch_olcnt] = 0;
      else
        ol_amount[batch_olcnt] = (lrand48 () % 999999 + 1) ;
      randstr (str24[j], 24, 24);

      if (gen) {
        if (cid < 2101) {
          fprintf (olfp, "%d %d %d %d %s %d %d 5 %ld %s\n", cid,
                  cdid, cwid, j + 1, sdate, ol_i_id[batch_olcnt], cwid,
                  ol_amount[batch_olcnt], str24[j]);
        }
        else {
          /* Insert a default date instead of null date */
          fprintf (olfp, "%d %d %d %d 01-Jan-1811 %d %d 5 %ld %s\n", cid,
                  cdid, cwid, j + 1, ol_i_id[batch_olcnt], cwid,
                  ol_amount[batch_olcnt], str24[j]);
        }
      }
      else {
        ol_o_id[batch_olcnt] = cid;
        ol_d_id[batch_olcnt] = cdid;
        ol_w_id[batch_olcnt] = cwid;
        ol_number[batch_olcnt] = j + 1;
        ol_supply_w_id[batch_olcnt] = cwid;
        strncpy (ol_dist_info[batch_olcnt], str24[j], 24);
      }
    }
    if (gen) {
      fflush (olfp);
    }
  }

  o_cnt =  ORDEARR;
  ol_cnt =  batch_olcnt;

  for (j = 0; j < batch_olcnt; j++) {
```

```
  ol_o_id_len[j] = sizeof(int);
  ol_d_id_len[j] = sizeof(int);
  ol_w_id_len[j] = sizeof(int);
  ol_number_len[j] = sizeof(int);
  ol_i_id_len[j] = sizeof(int);
  ol_supply_w_id_len[j] = sizeof(int);
  ol_dist_info_len[j] = 24;
  ol_amount_len[j] = sizeof(int);
}
for (j = batch_olcnt; j < 15*ORDEARR; j++) {
  ol_o_id_len[j] = 0;
  ol_d_id_len[j] = 0;
  ol_w_id_len[j] = 0;
  ol_number_len[j] = 0;
  ol_i_id_len[j] = 0;
  ol_supply_w_id_len[j] = 0;
  ol_dist_info_len[j] = 0;
  ol_amount_len[j] = 0;
}

o_id_clen = ORDEARR;
o_d_id_clen = ORDEARR;
o_w_id_clen = ORDEARR;
o_c_id_clen = ORDEARR;
o_carrier_id_clen = ORDEARR;
o_ol_cnt_clen = ORDEARR;

ol_o_id_clen = batch_olcnt;
ol_d_id_clen = batch_olcnt;
ol_w_id_clen = batch_olcnt;
ol_number_clen = batch_olcnt;
ol_i_id_clen = batch_olcnt;
ol_supply_w_id_clen = batch_olcnt;
ol_dist_info_clen = batch_olcnt;
ol_amount_clen = batch_olcnt;

OCIERROR(errhp, OCIStmtExecute(tpcsvc, curo1, errhp, (ub4) 1, (ub4) 0,
            (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
            (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS ));

if ((++loopcount) % 50) {
    fprintf (stderr, ".");
} else {
        fprintf (stderr, " %d orders committed\n   ", row);
      }
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d orders loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
            nrows, end_time - begin_time, end_cpu - begin_cpu);
  }

/*---------------------------------------------------------------+
 | Load the NEW-ORDER table.                                     |
 +---------------------------------------------------------------*/

  if (do_A || do_n) {
    nrows = (eware - bware + 1) * NEWOFAC * DISTFAC;

    fprintf (stderr, "Loading/generating new-order: w%d - w%d (%d rows)\n   ",
            bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    cid = 0;
    cdid = 1;
    cwid = bware;
    loopcount = 0;

    for (row = 0; row < nrows; ) {
      for (i = 0; i < NEWOARR; i++, row++) {
        cid++;
        if (cid > NEWOFAC) {
          cid = 1;
          cdid++;
          if (cdid > DISTFAC) {
            cdid = 1;
            cwid++;
          }
        }

        if (gen) {
          printf ("%d %d %d\n", cid + 2100, cdid, cwid);
```

```
        }
        else {
          no_o_id[i] = cid + 2100;
          no_d_id[i] = cdid;
          no_w_id[i] = cwid;
        }
      }

      if (gen) {
        fflush (stdout);
      }
      else {
        status = OCIStmtExecute(tpcsvc, curno, errhp, (ub4) NEWOARR, (ub4) 0,
                        (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                        (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
                    fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n", cwid,
cdid, cid + 2100);
          OCIERROR(errhp, status);
          quit ();
          exit (1);
        }
      }

      if ((++loopcount) % 45)
        fprintf (stderr, ".");
      else
        fprintf (stderr, " %d rows committed\n   ", row);
    }

    end_time = gettime ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done.  %d rows loaded/generated in %10.2f sec. (%10.2f
cpu)\n\n",
          nrows, end_time - begin_time, end_cpu - begin_cpu);
  }

/*-------------------------------------------------------------+
 | clean up and exit.                                          |
 +-------------------------------------------------------------*/

  if (olfp)
    fclose (olfp);
  if (!gen)
    quit ();
  exit (0);

}

void initperm ()
{
  int i;
  int pos;
  int temp;

  /* init randperm3000 */

  for (i = 0; i < 3000; i++)
    randperm3000[i] = i + 1;
  for (i = 3000; i > 0; i--) {
    pos = lrand48 () % i;
    temp = randperm3000[i - 1];
    randperm3000[i - 1] = randperm3000[pos];
    randperm3000[pos] = temp;
  }
}

void randstr (str, x, y)
char *str;
int x;
int y;
{
  int i, j;
  int len;

  len = (lrand48 () % (y - x + 1)) + x;
  for (i = 0; i < len; i++) {
    j = lrand48 () % 62;
    if (j < 26)
      str[i] = (char) (j + 'a');
    else if (j < 52)
      str[i] = (char) (j - 26 + 'A');
    else
      str[i] = (char) (j - 52 + '0');
  }
  str[len] = '\0';
```

```
}

void randdatastr (str, x, y)
char *str;
int x;
int y;
{
  int i, j;
  int len;
  int pos;

  len = (lrand48 () % (y - x + 1)) + x;
  for (i = 0; i < len; i++) {
    j = lrand48 () % 62;
    if (j < 26)
      str[i] = (char) (j + 'a');
    else if (j < 52)
      str[i] = (char) (j - 26 + 'A');
    else
      str[i] = (char) (j - 52 + '0');
  }
  str[len] = '\0';
  if ((lrand48 () % 10) == 0) {
    pos = (lrand48 () % (len - 8));
    str[pos] = 'O';
    str[pos + 1] = 'R';
    str[pos + 2] = 'I';
    str[pos + 3] = 'G';
    str[pos + 4] = 'I';
    str[pos + 5] = 'N';
    str[pos + 6] = 'A';
    str[pos + 7] = 'L';
  }
}

void randnum (str, len)
char *str;
int len;
{
  int i;

  for (i = 0; i < len; i++)
    str[i] = (char) (lrand48 () % 10 + '0');
  str[len] = '\0';

}

void randlastname (str, id)
char *str;
int id;
{
  id = id % 1000;
  strcpy (str, lastname[id / 100]);
  strcat (str, lastname[(id / 10) % 10]);
  strcat (str, lastname[id % 10]);
}

int NURand (A, x, y, cnum)
int A, x, y, cnum;
{
  int a, b;

  a = lrand48 () % (A + 1);
  b = (lrand48 () % (y - x + 1)) + x;
  return ((((a | b) + cnum) % (y - x + 1)) + x);

}

void sysdate (sdate)
char *sdate;
{
  time_t tp;
  struct tm *tmptr;

  time (&tp);
  tmptr = localtime (&tp);
  strftime (sdate, 29, "%d-%b-%Y", tmptr);
}

int ocierror(fname, lineno, errhp, status)
char *fname;
int lineno;
OCIError *errhp;
sword status;
{
```

```
text errbuf[512];
sb4 errcode;
sb4 lstat;
ub4 recno=2;

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

dpbcore.h

```
/* Copyright (c) Oracle Corporation 1993, 1992.  All Rights Reserved. */

/*
NAME    DPBCORE.H

DESCRIPTION
  Header for CORE function

NOTES
  Desktop Performance Group

MODIFIED      (MM/DD/YY)
  B Moriarty   06/02/95 -  add dpbetime() for accurate elapsed time measure
  B Moriarty   05/26/95 -  add dpboradt() for new reporting
  B Moriarty   05/10/95 -  add dpbcpu() for tpcc
  C Kelly      04/21/94 -  add dpbinpgm() and dpbxtpgm() for Netware NLMs
  C Kelly      02/24/93 -  add dpbfsync()
  B Moriarty   11/12/93 -  add dpbgetprty()
  R Keller     10/18/93 -  add dpbprty()
  R Keller     03/06/92 -  initial version

*/

#ifndef __dpbcore__
# define __dpbcore__
```

```
#include <stdio.h>
#include "dpbpcntl.h"

#ifdef __STDC__                         /* ANSI C               */
int         dpbfsync(FILE *);           /* fsync for ACID       */
int         dpbgetprty(char *,char *,int);  /* get O/S priority     */
void        dpbinpgm(void);             /* pgm. init. function  */
unsigned long dpbpchk(pcntl *);         /* check on forked process  */
unsigned long dpbproc(char *[], pcntl *);   /* spawn/fork new process   */
int         dpbprty(char *);            /* set O/S priority     */
clock_t     dpbtimef(void);             /* get time             */
clock_t     dpbcpu(void);               /* get CPU time         */
void        dpbwait(clock_t);           /* wait routine in millisec */
void        dpbxtpgm(void);             /* pgm exit routine     */
int         dpboradt(char *);           /* sys date time in ora form*/
clock_t     dpbetime(void);             /* elapsed time         */
#else                                   /* K&R C                */
int         dpbfsync();                 /* fsync for ACID       */
int         dpbgetprty();               /* get O/S priority     */
void        dpbinpgm();                 /* pgm. init. function  */
unsigned long dpbpchk();                /* check on forked process  */
unsigned long dpbproc();                /* spawn/fork new process   */
int         dpbprty();                  /* set O/S priority     */
clock_t     dpbtimef();                 /* get time             */
clock_t     dpbcpu();                   /* get cpu time         */
void        dpbwait();                  /* wait routine in millisec */
void        dpbxtpgm();                 /* pgm exit routine     */
int         dpboradt();                 /* sys date time in ora form*/
clock_t     dpbetime();                 /* elapsed time         */
#endif /* __STDC__ */


#endif /* __dpbcore__ */
```

dpbpcntl.h

```
/* Copyright (c) Oracle Corporation 1993, 1992.  All Rights Reserved. */

/*
NAME    DPBPCNTL.H

DESCRIPTION
  OSD structures for process control

NOTES
  Desktop Performance Group

MODIFIED      (MM/DD/YY)
  R Keller    02/03/93 -  initial version

*/

#ifndef __dpbpcntl__
# define __dpbpcntl__



#ifdef ORA_OS2                          /* IBM OS/2 2.x         */
# define INCL_DOSPROCESS
# include <os2.h>
typedef struct _pcntl
{
  RESULTCODES rcodes;
} pcntl;
#endif /* ORA_OS2 */                    /* IBM OS/2 2.x         */



#ifdef ORA_NT                           /* Microsoft Windows NT     */
# include <windows.h>                   /*                      */
typedef struct _pcntl
{
  PROCESS_INFORMATION proc_info;
} pcntl;
#endif /* ORA_NT */                     /* Microsoft Windows NT     */



#ifdef ORA_AUX                          /* Apple A/UX           */
typedef struct _pcntl
{
  int dummy;
} pcntl;
#endif /* ORA_AUX */                    /* Apple A/UX           */
```

```
#ifdef ORA_NW                           /* Novell Netware          */
typedef struct _pcntl
{
  int dummy;
} pcntl;
#endif /* ORA_NW */                     /* Novell Netware          */



#endif /* __dpbpcntl__ */
```

Makefile.linux

```
#gcc=/usr/local/bin/gcc
#============================================================
======+
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                  |
#                    All Rights Reserved                         |
#============================================================
======+
# FILENAME
#     Makefile
# DESCRIPTION
#     Makefile for batch driver, load program and tx testing.
#============================================================
======
#
#  Programs:
#
#     tpcc.exe       : OCI TPC-C generator
#     tpccload.exe   : Database loader for TPC-C
#     single_txn.exe : OCI program to test the TPC-C transactions
#     getrand.exe    :
#     90per.exe      :
#     runtpb.exe     :
#     sleep.exe      :
#     press_return.exe     :
#     runid.exe      :
#

all: compile load

#include $(ORACLE_HOME)/bench/buildtools/prefix.mk

#I_SYM=-I
#include $(ORACLE_HOME)/rdbms/lib/env_rdbms.mk

#LINK=/opt/SunProd/SUNWspro6.1/bin/../WS6U1/bin/cc
#CC=/opt/SunProd/SUNWspro6.1/bin/../WS6U1/bin/cc
LINK=/usr/bin/gcc
CC=/usr/bin/gcc

#LDFLAGS=-L$(ORACLE_HOME)/rdbms/lib/ -L$(ORACLE_HOME)/lib/ -dy \
#-L$(ORACLE_HOME)/rdbms/lib/ -L$(ORACLE_HOME)/lib/ \
#$(ORACLE_HOME)/rdbms/lib/defopt.o -lclntsh \
#`cat $(ORACLE_HOME)/lib/sysliblist` \
#-o $@

LDFLAGS=-L$(ORACLE_HOME)/rdbms/lib/ -L$(ORACLE_HOME)/lib/ -dy \
-L$(ORACLE_HOME)/rdbms/lib/ -L$(ORACLE_HOME)/lib/ \
$(ORACLE_HOME)/rdbms/lib/defopt.o -lclntsh \
`cat $(ORACLE_HOME)/lib/sysliblist` \
-L$(ORACLE_HOME)/lib -o $@

#-R$(ORACLE_HOME)/lib -laio -lthread -lposix4 -lkstat -lm -o $@

I_SYM=-I

TARGS=compile cleanup
REMOVE=rm

DPB_LIB_DIR=../lib
DPB_LIB=$(DPB_LIB_DIR)/dpblibunix.o

TPCBIN=../../bin
INCLUDE=$(I_SYM). $(I_SYM)$(ORACLE_HOME)/rdbms/demo \
 $(I_SYM)$(ORACLE_HOME)/rdbms/public \
 $(I_SYM)$(ORACLE_HOME)/rdbms/include \
 $(I_SYM)$(ORACLE_HOME)/plsql/public \
 $(I_SYM)$(ORACLE_HOME)/network/public \
 $(I_SYM)$(DPB_LIB_DIR)
ITUX=$(I_SYM)$(ROOTDIR)/include

MEMBS=
```

```
OBJS=tpccload.o c_trans.o c_drv_o7.o c_dump.o tpccpl.o getrand.o 90per.o report.o
errrpt.o sleep.o press_return.o runid.o
CTRAN_OBJS=plnew.o plpay.o plord.o pldel.o plsto.o
CTRANPOCI_OBJS=plnew.o plpay_oci.o plord.o pldel.o plsto.o
CTRANTUX_OBJS=plnew_tux.o plpay.o plord.o pldel_tux.o plsto.o
OTHER_OBJS=c_drv_val.o test_drv.o test_sample.o test_tran.o single_txn_ran.o
TUX_OBJS=c_drv_tux.o tpccpl_tux.o tpccsvr.o

files:

compile: $(OBJS) $(DPB_LIB)
        @-$(DOTARGS)

load: $(TPCBIN)/tpcc.exe $(TPCBIN)/tpccload.exe \
        $(TPCBIN)/single_txn.exe $(TPCBIN)/90per.exe \
        $(TPCBIN)/runtpb.exe $(TPCBIN)/getrand.exe \
        $(TPCBIN)/sleep.exe $(TPCBIN)/runid.exe \
        $(TPCBIN)/single_txn_ran.exe \
        $(TPCBIN)/press_return.exe
        @-$(DOTARGS)

cleanup:
        $(REMOVE) $(OBJS) $(CTRAN_OBJS) $(CTRANTUX_OBJS)
$(OTHER_OBJS) \
        $(TPCBIN)/tpcc.exe $(TPCBIN)/tpccload.exe \
        $(TPCBIN)/single_txn.exe $(TPCBIN)/90per.exe \
        $(TPCBIN)/runtpb.exe $(TPCBIN)/getrand.exe \
        $(TPCBIN)/sleep.exe $(TPCBIN)/runid.exe \
        $(TPCBIN)/single_txn_ran.exe \
        $(TPCBIN)/press_return.exe \
        $(TUX_OBJS)
        @-$(DOTARGS)

$(DPB_LIB):
        ( cd $(DPB_LIB_DIR); $(MAKE) -f Makefile.linux )

report.o: report.c results.h
        $(CC) $(CFLAGS) $(INCLUDE) -c report.c

errrpt.o: errrpt.c results.h
        $(CC) $(CFLAGS) $(INCLUDE) -c errrpt.c

sleep.o: sleep.c
        $(CC) $(CFLAGS) $(INCLUDE) -c sleep.c

press_return.o: press_return.c
        $(CC) $(CFLAGS) $(INCLUDE) -c press_return.c

runid.o: runid.c
        $(CC) $(CFLAGS) $(INCLUDE) -c runid.c

tpccload.o: tpccload.c tpcc.h
        $(CC) $(CFLAGS) $(INCLUDE) -c tpccload.c

c_drv_o7.o: c_drv_o7.c tpcc.h
        $(CC) $(CFLAGS) $(INCLUDE) -c c_drv_o7.c

c_drv_val.o: c_drv.c tpcc.h
        $(CP) c_drv.c c_drv_val.c
        -$(CC) $(CFLAGS) -DVALIDATE $(INCLUDE) -c c_drv_val.c
        $(REMOVE) c_drv_val.c

c_drv_tux.o: c_drv.c tpcc.h
        $(CP) c_drv.c c_drv_tux.c
        -$(CC) $(CFLAGS) -DTUX $(INCLUDE) $(ITUX) -c c_drv_tux.c
        $(REMOVE) c_drv_tux.c

c_dump.o: c_dump.c tpcc.h
        $(CC) $(CFLAGS) $(INCLUDE) -c c_dump.c

single_txn.o: single_txn.c tpcc.h
        $(CC) $(CFLAGS) $(INCLUDE) -c single_txn.c

single_txn_ran.o: single_txn_ran.c tpcc.h
        $(CC) $(CFLAGS) $(INCLUDE) -c single_txn_ran.c

runtpb.o: runtpb.c
        $(CC) $(CFLAGS) -DORA_AUX $(INCLUDE) -c runtpb.c

c_trans.o: $(CTRAN_OBJS)
        $(LD) -r -o $@ $(CTRAN_OBJS)

c_trans_tux.o: $(CTRANTUX_OBJS)
        $(LD) -r -o $@ $(CTRANTUX_OBJS)

tpccpl.o: tpccpl.c tpcc.h
```

```
	$(CC) $(CFLAGS) $(INCLUDE) -c tpccpl.c

tpccpl_tux.o: tpccpl.c tpcc.h
	$(CP) tpccpl.c tpccpl_tux.c
	-$(CC) $(CFLAGS) -DTUX $(INCLUDE) $(ITUX) -c tpccpl_tux.c
	$(REMOVE) tpccpl_tux.c

plnew_tux.o: plnew.c tpcc.h
	$(CP) plnew.c plnew_tux.c
	-$(CC) $(CFLAGS) -DTUX $(INCLUDE) $(ITUX) -c plnew_tux.c
	$(REMOVE) plnew_tux.c

plnew.o: plnew.c tpcc.h
	$(CC) $(CFLAGS) $(INCLUDE) -c plnew.c

plpay.o: plpay.c tpcc.h
	$(CC) $(CFLAGS) $(INCLUDE) -c plpay.c

plord.o: plord.c tpcc.h
	$(CC) $(CFLAGS) $(INCLUDE) -c plord.c

pldel_tux.o: pldel.c tpcc.h
	$(CP) pldel.c pldel_tux.c
	-$(CC) $(CFLAGS) -DTUX $(INCLUDE) $(ITUX) -c pldel_tux.c
	$(REMOVE) pldel_tux.c

pldel.o: pldel.c tpcc.h
	$(CC) $(CFLAGS) $(INCLUDE) -c pldel.c

plsto.o: plsto.c tpcc.h
	$(CC) $(CFLAGS) $(INCLUDE) -c plsto.c

tpccsvr.o: tpccsvr.c tpcc.h
	$(CC) $(CFLAGS) $(INCLUDE) $(ITUX) -c tpccsvr.c

getrand.o: getrand.c
	$(CC) $(CFLAGS) $(INCLUDE) -c getrand.c

90per.o: 90per.c
	$(CC) $(CFLAGS) $(INCLUDE) -c 90per.c

$(TPCBIN)/getrand.exe: getrand.o
	$(LINK) $(LDFLAGS) \
	getrand.o -lc

$(TPCBIN)/sleep.exe: sleep.o
	$(LINK) $(LDFLAGS) \
	sleep.o -lc

$(TPCBIN)/press_return.exe: press_return.o
	$(LINK) $(LDFLAGS) \
	press_return.o -lc

$(TPCBIN)/runid.exe: runid.o
	$(LINK) $(LDFLAGS) \
	runid.o $(DPB_LIB) -lc

$(TPCBIN)/90per.exe: 90per.o
	$(LINK) $(LDFLAGS) \
	90per.o -lc

$(TPCBIN)/tpccload.exe: tpccload.o $(DPB_LIB)
	$(LINK) $(LDFLAGS) \
	tpccload.o $(DPB_LIB) \
	$(SSABED) $(DEF_OPT) $(TTLIBS) -lc

$(TPCBIN)/runtpb.exe: runtpb.o $(DPB_LIB)
	$(LINK) $(LDFLAGS) \
	runtpb.o $(DPB_LIB) \
	$(SSABED) $(DEF_OPT) $(TTLIBS) -lc

$(TPCBIN)/tpcc.exe: c_drv_o7.o c_trans.o tpccpl.o c_dump.o report.o errrpt.o
$(DPB_LIB)
	$(LINK) $(LDFLAGS) \
	c_drv_o7.o c_trans.o tpccpl.o c_dump.o errrpt.o report.o $(DPB_LIB)\
    $(SSABED) $(DEF_OPT) $(TTLIBS) -lc

$(TPCBIN)/single_txn.exe: single_txn.o $(DPB_LIB) c_trans.o tpccpl.o c_dump.o
	$(LINK) $(LDFLAGS) \
	single_txn.o c_trans.o tpccpl.o c_dump.o $(DPB_LIB) \
	$(SSABED) $(DEF_OPT) $(TTLIBS) -lc

$(TPCBIN)/single_txn_ran.exe: single_txn_ran.o $(DPB_LIB) c_trans.o tpccpl.o
c_dump.o
	$(LINK) $(LDFLAGS) \
	single_txn_ran.o c_trans.o tpccpl.o c_dump.o $(DPB_LIB) \
```

```
	$(SSABED) $(DEF_OPT) $(TTLIBS) –lc
```

Makefile.linux

```
#=============================================================
======+
#       Copyright (c) 1996  Oracle Corp, Redwood Shores, CA      |
#                OPEN SYSTEMS PERFORMANCE GROUP                  |
#                   All Rights Reserved                          |
#=============================================================
======+
# FILENAME
#     Makefile
# DESCRIPTION
#     Makefile for lib for batch driver, load program and tx testing.
#=============================================================
======
#
#  Programs:
#
#     dpblibunix.o
#

all: compile dpblibunix.o

#include $(ORACLE_HOME)/bench/buildtools/prefix.mk
I_SYM=-I
#include $(ORACLE_HOME)/rdbms/lib/env_rdbms.mk
REMOVE=rm
#CC=/opt/SunProd/SUNWspro6.1/bin/../WS6U1/bin/cc
CC=/usr/bin/gcc

TARGS=compile cleanup

TPCBIN=.
INCLUDE=$(I_SYM). $(I_SYM)$(ORACLE_HOME)/rdbms/demo \
 $(I_SYM)$(ORACLE_HOME)/rdbms/public \
 $(I_SYM)$(ORACLE_HOME)/rdbms/include \
 $(I_SYM)$(ORACLE_HOME)/plsql/public \
 $(I_SYM)$(ORACLE_HOME)/network/public
ITUX=$(I_SYM)$(ROOTDIR)/include

MEMBS=

OBJS=gettime.o dpbproc.o dpbwait.o dpbpchk.o dpbtimef.o

CFLAGS=

files:

compile: $(OBJS)
	@-$(DOTARGS)

cleanup:
	$(REMOVE) $(OBJS) dpblibunix.o

dpbtimef.o: dpbtimef.c
	$(CC) $(CFLAGS) -DORA_PC $(INCLUDE) -c dpbtimef.c

dpbproc.o: dpbproc.c
	$(CC) $(CFLAGS) -DORA_AUX $(INCLUDE) -c dpbproc.c

dpbwait.o: dpbwait.c
	$(CC) $(CFLAGS) -DORA_AUX $(INCLUDE) -c dpbwait.c

dpbpchk.o: dpbpchk.c
	$(CC) $(CFLAGS) -DORA_AUX $(INCLUDE) -c dpbpchk.c

gettime.o: gettime.c
	$(CC) $(CFLAGS) $(INCLUDE) -c gettime.c

trigger.o: trigger.c

dpblibunix.o: $(OBJS)
	$(LD) -r -o $@ $(OBJS)

c_trans_tux.o: $(CTRANTUX_OBJS)
	$(LD) -r -o $@ $(CTRANTUX_OBJS)
```

Appendix C : Tunable Parameters

The procedure of a performance run

1. Boot up server, clients and RTEs.
2. Set IRQs on the server using irq.sh.
3. Start four Oracle listeners on the server.
4. Set affinity for these listeners using setaffi.sh.
5. Startup the database using run.ora.
6. Start Apache HTTP Server on the clients using httpd.conf.
7. Start Tuxedo on the clients using ubbconfig.
8. Set priority of Oracle processes using setrrpri.sh.
9. Start the RTEs.

Server Configuration

rr.c

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

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

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

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

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

        exit(0);
}
```

setrrpri.sh

```
#!/bin/sh

/root/priority/rr -p 48 $(ps aux | grep oracle_bin | grep -v grep | awk '{print $2}')
/root/priority/rr -p 48 $(ps aux | grep oracletpcc | grep -v grep | awk '{print $2}')
/root/priority/rr -p 48 $(ps aux | grep ora_ | grep -v grep | awk '{print $2}')

/root/priority/rr -p 49 $(grep 'LGWR started'
/home/oracle/031014/rdbms/log/alert_tpcc.log | tail -n 1 | awk '{print substr($6,4)}')

/home/oracle/tpcc-kit/oracleaffi/setaffi $(grep 'LGWR started'
/home/oracle/031014/rdbms/log/alert_tpcc.log | tail -n 1 | awk '{print substr($6,4)}') 4
```

irq.sh

```
#!/bin/sh
echo "00000004" > /proc/irq/49/smp_affinity
echo "00000004" > /proc/irq/50/smp_affinity
echo "00000001" > /proc/irq/51/smp_affinity
echo "00000008" > /proc/irq/52/smp_affinity
echo "00000010" > /proc/irq/53/smp_affinity
echo "00000040" > /proc/irq/54/smp_affinity
echo "00000001" > /proc/irq/55/smp_affinity
echo "00000008" > /proc/irq/56/smp_affinity
echo "00000020" > /proc/irq/57/smp_affinity
echo "00000080" > /proc/irq/58/smp_affinity
echo "00000001" > /proc/irq/59/smp_affinity
echo "00000008" > /proc/irq/60/smp_affinity
echo "00000010" > /proc/irq/61/smp_affinity
echo "00000040" > /proc/irq/62/smp_affinity
echo "00000002" > /proc/irq/63/smp_affinity
```

setaffi.c

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

```
#include <sched.h>
#include <errno.h>

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

/*
 * provide the proper syscall information if our libc
 * is not yet updated.
 */

#undef __NR_sched_setaffinity
#ifndef __NR_sched_setaffinity

#define __NR_sched_setaffinity     1231
#define __NR_sched_getaffinity     1232

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

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

#endif

int main(int argc, char * argv[])
{
            unsigned long new_mask;
            unsigned long junk[10];
            unsigned long cur_mask;
            unsigned long junk2[10];
            unsigned int len = sizeof(new_mask);
            pid_t pid;

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

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

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

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

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

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

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

            return 0;
}
```

listener.ora

```
LISTENER1=
 (ADDRESS_LIST=
  (ADDRESS=(PROTOCOL=tcp)(HOST=10.1.1.222)(PORT=2521))
 )

SID_LIST_LISTENER1=
 (SID_LIST=
  (SID_DESC=(SID_NAME=tpcc)(ORACLE_HOME=/home/oracle/031014))
 )

LISTENER2=
 (ADDRESS_LIST=
```

```
   (ADDRESS=(PROTOCOL=tcp)(HOST=10.1.1.222)(PORT=2522))
  )

SID_LIST_LISTENER2=
  (SID_LIST=
   (SID_DESC=(SID_NAME=tpcc)(ORACLE_HOME=/home/oracle/031014))
  )

LISTENER3=
  (ADDRESS_LIST=
   (ADDRESS=(PROTOCOL=tcp)(HOST=10.1.1.222)(PORT=2523))
  )

SID_LIST_LISTENER3=
  (SID_LIST=
   (SID_DESC=(SID_NAME=tpcc)(ORACLE_HOME=/home/oracle/031014))
  )

LISTENER4=
  (ADDRESS_LIST=
   (ADDRESS=(PROTOCOL=tcp)(HOST=10.1.1.222)(PORT=2524))
  )

SID_LIST_LISTENER4=
  (SID_LIST=
   (SID_DESC=(SID_NAME=tpcc)(ORACLE_HOME=/home/oracle/031014))
  )
```

listener_start.sh

```
#!/bin/sh
lsnrctl start listener1
lsnrctl start listener2
lsnrctl start listener3
lsnrctl start listener4
```

lsnraffi.sh

```
#!/bin/sh
ls1=`ps aux | grep listener1 | grep -v grep | awk '{print $2}'`
./setaffi $ls1 3
ls2=`ps aux | grep listener2 | grep -v grep | awk '{print $2}'`
./setaffi $ls2 c
ls3=`ps aux | grep listener3 | grep -v grep | awk '{print $2}'`
./setaffi $ls3 30
ls4=`ps aux | grep listener4 | grep -v grep | awk '{print $2}'`
./setaffi $ls4 c0
```

boot.local

```
#! /bin/sh
#
# Copyright (c) 2002 SuSE Linux AG Nuernberg, Germany.  All rights reserved.
#
# Author: Werner Fink <werner@suse.de>, 1996
#         Burchard Steinbild, 1996
#
# /etc/init.d/boot.local
#
# script with local commands to be executed from init on system startup
#
# Here you should add things, that should happen directly after booting
# before we're going to the first run level.
#

echo "512 32000 512 128" > /proc/sys/kernel/sem
echo 0x8000000000 > /proc/sys/kernel/shmmax
echo 0x20000000 > /proc/sys/kernel/shmall
echo 5242880 > /proc/sys/fs/aio-max-nr
echo 1000 > /proc/sys/vm/nr_hugepages
```

~oracle/.bashrc

```
# .bashrc

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

alias ls='ls -aF'
export ORACLE_BASE=/home/oracle
export ORACLE_HOME=$ORACLE_BASE/031014
```

```
export ORACLE_SID=tpcc
export
LD_LIBRARY_PATH=$ORACLE_HOME/lib:$ORACLE_HOME/rdbms/lib:/lib:/usr/lib
export PATH=$ORACLE_HOME/bin:$PATH
export FOREGROUND_TIMEOUT=1
```

Client Configuration

httpd.conf

```
# this file shows Apache settings that need to be added/modified to support  TPC-C module

#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
# Apache issue: This have to be as long as the TPC-C run if you use keep-alive user connections from RTE

Timeout 14400

KeepAlive On
KeepAliveTimeout 15000
MaxKeepAliveRequests 15000

HostnameLookups Off
UseCanonicalName Off

# worker MPM
# StartServers: initial number of server processes to start
# MaxClients: maximum number of simultaneous client connections
# MinSpareThreads: minimum number of worker threads which are kept spare
# MaxSpareThreads: maximum number of worker threads which are kept spare
# ThreadsPerChild: constant number of worker threads in each server process
# MaxRequestsPerChild: maximum number of requests a server process serves
<IfModule worker.c>
StartServers       270
ServerLimit           300
ThreadLimit        100
MaxClients         13000
MinSpareThreads     25
MaxSpareThreads    13000
ThreadsPerChild     50
MaxRequestsPerChild  0
</IfModule>

<IfModule !mpm_winnt.c>
<IfModule !mpm_netware.c>
#
# If you wish httpd to run as a different user or group, you must run
# httpd as root initially and it will switch.
#
# User/Group: The name (or #number) of the user/group to run httpd as.
#  . On SCO (ODT 3) use "User nouser" and "Group nogroup".
#  . On HPUX you may not be able to use shared memory as nobody, and the
#    suggested workaround is to create a user www and use that user.
#  NOTE that some kernels refuse to setgid(Group) or semctl(IPC_SET)
#  when the value of (unsigned)Group is above 60000;
#  don't use Group #-1 on these systems!
#
#User nobody
#Group #-1
#  oracle:dba is a user that have access to tuexdo server application


User oracle
Group    dba
</IfModule>
</IfModule>

ServerName acl33
Listen 8080

#
# LoadModule foo_module modules/mod_foo.so
LoadModule tpcc_module       modules/libmodtpcc.so

#
# tpcc module (ported from TPCC ISAPI extension)
#
<IfModule mod_tpcc.c>
            MaxConnections 13000
```

```
        MaximumWarehouses 50000
        Path      /tmp/
    Shm_Path /tmp/
        Log       OFF
        Debug     OFF
</IfModule>

<Location /tpcc.dll>
   SetHandler tpcc
</Location>
```

tnsnames.ora

```
tpcc =
  (DESCRIPTION=
    (ADDRESS=(PROTOCOL=tcp)(HOST=10.1.1.222)(PORT=2521))
    (CONNECT_DATA=(SID=tpcc))
  )
```

~oracle/.bashrc

```
# .bashrc

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

export LANG=C
export TUXDIR=/home/bea/tuxedo8.1
export APPDIR=/home/oracle/tpccrun/tmdir
export TUXCONFIG=/home/bea/tuxedo8.1/tuxconfig
export COBCPY=$TUXDIR/cobinclude
export COBOPT="-C ANS85 -C ALIGN=8 -C NOIBMCOMP -C TRUNC=ANSI -C
OSEXT=cbl"
export WEBJAVADIR=$TUXDIR/udataobj/webgui/java
export ORACLE_BASE=/home/oracle
export ORACLE_HOME=$ORACLE_BASE/031027
export
LD_LIBRARY_PATH=$ORACLE_HOME/lib:$TUXDIR/lib:LD_LIBRARY_PAT
H
export ORACLE_SID=tpcc
export PATH=$ORACLE_HOME/bin:$TUXDIR/bin:/usr/local/apache2/bin:$PATH
export LIBPATH=$ORACLE_HOME/lib:$TUXDIR/lib:$LIBPATH
export SHLIB_PATH=$ORACLE_HOME/lib32:TUXDIR/lib:$SHLIB_PATH
export TWO_TASK=tpcc

ulimit -u 30000
```

ubbconfig

```
#       Copyright (c) 1994 Unix System Laboratories, Inc.
#       All rights reserved
#
#       THIS IS UNPUBLISHED PROPRIETARY
#       SOURCE CODE OF Unix System Laboratories, Inc.
#       The copyright notice above does not
#       evidence any actual or intended
#       publication of such source code.
#
#ident  "@(#) apps/rpcsimp/ubbconfig      $Revision: 1.1 $"
*RESOURCES
IPCKEY 200042
MAXACCESSERS 15000
MAXGTT 2048
MAXSERVERS 1200
MAXSERVICES 1200
MODEL SHM
MASTER acl33
LDBAL Y
SCANUNIT 15
SANITYSCAN 20
BLOCKTIME 60
DBBLWAIT  4
BBLQUERY 120

*MACHINES
DEFAULT:
"acl33"
LMID=acl33
TUXDIR="/home/bea/tuxedo8.1"
APPDIR="/home/oracle/tpccrun/tmdir"
TUXCONFIG="/home/bea/tuxedo8.1/tuxconfig"
```

```
ULOGPFX="/home/bea/tuxedo8.1/ulog"
#UID= 0
#GID= 0


*GROUPS
GROUPGEN1
LMID=acl33
GRPNO=1
OPENINFO=NONE

GROUPGEN2
LMID=acl33
GRPNO=2
OPENINFO=NONE

GROUPGEN3
LMID=acl33
GRPNO=3
OPENINFO=NONE

GROUPGEN4
LMID=acl33
GRPNO=4
OPENINFO=NONE

GROUPGEN5
LMID=acl33
GRPNO=5
OPENINFO=NONE

GROUPGEN6
LMID=acl33
GRPNO=6
OPENINFO=NONE

GROUPGEN7
LMID=acl33
GRPNO=7
OPENINFO=NONE

GROUPGEN8
LMID=acl33
GRPNO=8
OPENINFO=NONE

GROUPGEN9
LMID=acl33
GRPNO=9
OPENINFO=NONE

*SERVERS
DEFAULT:
generic SRVGRP=GROUPGEN1
SRVID=100
MIN=1 MAX=1
REPLYQ=Y
RQADDR=OPSTUXSERVER1
CLOPT="-s OPSTUXSERVER:OPSTUXSERVER"

generic SRVGRP=GROUPGEN2
SRVID=200
MIN=1 MAX=1
REPLYQ=Y
RQADDR=OPSTUXSERVER2
CLOPT="-s OPSTUXSERVER:OPSTUXSERVER"

generic SRVGRP=GROUPGEN3
SRVID=300
MIN=1 MAX=1
REPLYQ=Y
RQADDR=OPSTUXSERVER3
CLOPT="-s OPSTUXSERVER:OPSTUXSERVER"

generic SRVGRP=GROUPGEN4
SRVID=400
MIN=1 MAX=1
REPLYQ=Y
RQADDR=OPSTUXSERVER4
CLOPT="-s OPSTUXSERVER:OPSTUXSERVER"

generic SRVGRP=GROUPGEN5
SRVID=500
MIN=1 MAX=1
REPLYQ=Y
RQADDR=OPSTUXSERVER5
```

```
CLOPT="-s OPSTUXSERVER:OPSTUXSERVER"

generic SRVGRP=GROUPGEN6
SRVID=600
MIN=1 MAX=1
REPLYQ=Y
RQADDR=OPSTUXSERVER6
CLOPT="-s OPSTUXSERVER:OPSTUXSERVER"

generic SRVGRP=GROUPGEN7
SRVID=700
MIN=1 MAX=1
REPLYQ=Y
RQADDR=OPSTUXSERVER7
CLOPT="-s OPSTUXSERVER:OPSTUXSERVER"

generic SRVGRP=GROUPGEN8
SRVID=800
MIN=1 MAX=1
REPLYQ=Y
RQADDR=OPSTUXSERVER8
CLOPT="-s OPSTUXSERVER:OPSTUXSERVER"

generic SRVGRP=GROUPGEN9
SRVID=900
MIN=1 MAX=1
REPLYQ=Y
RQADDR=OPSTUXSERVER9
CLOPT="-s OPSTUXSERVER2:OPSTUXSERVER"

*SERVICES
#OPSTUXSERVER
```

RTE input parameter

The following parameters were used with NEC proprietary RTE.

RTE master driver

Help message:

```
Parameter                                   Default
-------------------------------------------------------------
General test parameters
-u Ramp Up Time (sec)                        600
-s Steady State Time (sec)                  1200
-d Ramp Down Time (sec)                      120
-c Number of Connections                     10
-w Number of Warehouses                      1
-n C value for NURand                        66

Reporting/Logging options
-R Name of Report File (txt format)          report.txt
-X Name of Report File (xlt format)          report.xlt
-P Name of NewOrder Trouguput Curve File     thrput.xlt
-l Enable Statistics Logging.
-L Name of Statistics Log File               slog
-o Generate 'success' File for Durability Test
-O Name of 'success' File                    succ

Misc
-C Connect Rate of a Client Driver (users/min)   1000
-S Start Rate of a Client Driver (users/min)     1000
-v Verbose Mode (experimental)

Tunable parameters
-W Comma Separated List of Distribution Weights
                                        4480 (NewOrder)
                                        4308 (Payment)
                                        404 (OrderStatus)
                                        404 (Delivery)
                                        404 (StockLevel)
-T Comma Separated List of Think Times (sec)
                                        12.05 (NewOrder)
                                        12.05 (Payment)
                                        10.05 (OrderStatus)
                                        5.05 (Delivery)
                                        5.05 (StockLevel)
-K Comma Separated List of Keying Times (sec)
                                        18.01 (NewOrder)
                                        3.01 (Payment)
                                        2.01 (OrderStatus)
                                        2.01 (Delivery)
                                        2.01 (StockLevel)
```

```
        -D Web Browser Delay Time (sec)                 0.10
```

Command-line parameters:

```
        master -u14400 -s10800 -d300 -c202400 -w20240 -C1500 -S50 -n66 -
T12.07,12.07,10.07,5.07,5.07
```

During the run, we adjusted the measurement interval dynamically.
Followings are the master.exe console output messages.

```
[Tue Dec 27 13:03:43] Start test with 202400/202400 user(s)
[Tue Dec 27 13:03:43] Start ramp up (CTRL-C to start M.I. immediately.)
[Tue Dec 27 13:03:43] Ramp up period: 0/14400 secs
[Tue Dec 27 13:03:44] Ramp up period: 1/14400 secs
[Tue Dec 27 13:03:45] Ramp up period: 2/14400 secs
[Tue Dec 27 13:03:46] Ramp up period: 3/14400 secs
                :
[Tue Dec 27 15:42:48] Ramp up period: 9545/14400 secs
[Tue Dec 27 15:42:49] Ramp up period: 9546/14400 secs
[Tue Dec 27 15:42:50] Ramp up period: 9547/14400 secs
[Tue Dec 27 15:42:50] Catch CTRL-C event. Go to next phase in 10 secs.
[Tue Dec 27 15:42:50] Sent duration changing message (9557, 10800, 300).
[Tue Dec 27 15:42:51] Ramp up period: 9548/9557 secs
[Tue Dec 27 15:42:52] Ramp up period: 9549/9557 secs
[Tue Dec 27 15:42:53] Ramp up period: 9550/9557 secs
[Tue Dec 27 15:42:54] Ramp up period: 9551/9557 secs
[Tue Dec 27 15:42:55] Ramp up period: 9552/9557 secs
[Tue Dec 27 15:42:56] Ramp up period: 9553/9557 secs
[Tue Dec 27 15:42:57] Ramp up period: 9554/9557secs
[Tue Dec 27 15:42:58] Ramp up period: 9555/9557 secs
[Tue Dec 27 15:42:59] Ramp up period: 9556/9557 secs
[Tue Dec 27 15:43:00] Start steady state (CTRL-C to stop M.I. immediately.)
[Tue Dec 27 15:43:00] Steady state period: 0/10800 secs
[Tue Dec 27 15:43:01] Steady state period: 1/10800 secs
[Tue Dec 27 15:43:02] Steady state period: 2/10800 secs
[Tue Dec 27 15:43:03] Steady state period: 3/10800 secs
                :
[Tue Dec 27 17:43:18] Steady state period: 7218/10800 secs
[Tue Dec 27 17:43:19] Steady state period: 7219/10800 secs
[Tue Dec 27 17:43:20] Steady state period: 7220/10800 secs
[Tue Dec 27 17:43:20] Catch CTRL-C event. Go to next phase in 10 secs.
[Tue Dec 27 17:43:20] Sent duration changing message (9557, 7230, 300).
[Tue Dec 27 17:43:21] Steady state period: 7221/7230 secs
[Tue Dec 27 17:43:22] Steady state period: 7222/7230 secs
[Tue Dec 27 17:43:23] Steady state period: 7223/7230 secs
[Tue Dec 27 17:43:24] Steady state period: 7224/7230 secs
[Tue Dec 27 17:43:25] Steady state period: 7225/7230 secs
[Tue Dec 27 17:43:26] Steady state period: 7226/7230 secs
[Tue Dec 27 17:43:27] Steady state period: 7227/7230 secs
[Tue Dec 27 17:43:28] Steady state period: 7228/7230 secs
[Tue Dec 27 17:43:29] Steady state period: 7229/7230 secs
[Tue Dec 27 17:43:30] Start ramp down (CTRL-C to stop test immediately.)
[Tue Dec 27 17:43:30] Ramp down period: 0/300 secs
[Tue Dec 27 17:43:31] Ramp down period: 1/300 secs
[Tue Dec 27 17:43:32] Ramp down period: 2/300 secs
                :
[Tue Dec 27 17:48:27] Ramp down period: 297/300 secs
[Tue Dec 27 17:48:28] Ramp down period: 298/300 secs
[Tue Dec 27 17:48:29] Ramp down period: 299/300 secs
[Tue Dec 27 17:48:30] Waiting for 160 client driver(s) to finish.
[Tue Dec 27 17:48:33] Waiting for 119 client driver(s) to finish.
[Tue Dec 27 17:48:36] Waiting for 79 client driver(s) to finish.
[Tue Dec 27 17:48:39] Waiting for 39 client driver(s) to finish.
[Tue Dec 27 17:48:42] Test complete. See report.txt for details.
```

RTE slave driver

Help message:

```
        Parameter                       Default
        ---------------------------------------
        -M Master Driver Servername     .
        -I HTTP Servername
        -p HTTP Server Port             80
        -c Number of Connections        10
        -o Starting User ID             1
        -d Disable HTTP Keep-Alive
```

Following is the script file used to launch all 202,400 users
(= 1265 users * 10 slave drivers * 16 RTE machines)

```
if %COMPUTERNAME%==RTE33 (
start cmd /k c:\client7 -Mbmaster -Iacl33 -o1 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl332 -o1266 -c1265 -p8080
sleep 80
```

```
start cmd /k c:\client7 -Mbmaster -Iacl33 -o2531 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl332 -o3796 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl33 -o5061 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl332 -o6326 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl33 -o7591 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl332 -o8856 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl33 -o10121 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl332 -o11386 -c1265 -p8080
sleep 80
)
if %COMPUTERNAME%==RTE34 (
start cmd /k c:\client7 -Mbmaster -Iacl34 -o12651 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl342 -o13916 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl34 -o15181 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl342 -o16446 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl34 -o17711 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl342 -o18976 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl34 -o20241 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl342 -o21506 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl34 -o22771 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl342 -o24036 -c1265 -p8080
sleep 80
)
if %COMPUTERNAME%==RTE35 (
start cmd /k c:\client7 -Mbmaster -Iacl35 -o25301 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl352 -o26566 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl35 -o27831 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl352 -o29096 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl35 -o30361 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl352 -o31626 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl35 -o32891 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl352 -o34156 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl35 -o35421 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl352 -o36686 -c1265 -p8080
sleep 80
)
if %COMPUTERNAME%==RTE36 (
start cmd /k c:\client7 -Mbmaster -Iacl36 -o37951 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl362 -o39216 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl36 -o40481 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl362 -o41746 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl36 -o43011 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl362 -o44276 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl36 -o45541 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl362 -o46806 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl36 -o48071 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl362 -o49336 -c1265 -p8080
sleep 80
)
if %COMPUTERNAME%==RTE39 (
start cmd /k c:\client7 -Mbmaster -Iacl39 -o50601 -c1265 -p8080
sleep 80
```

```
start cmd /k c:\client7 -Mbmaster -Iacl392 -o51866 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl39 -o53131 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl392 -o54396 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl39 -o55661 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl392 -o56926 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl39 -o58191 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl392 -o59456 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl39 -o60721 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl392 -o61986 -c1265 -p8080
sleep 80
)
if %COMPUTERNAME%==RTE40 (
start cmd /k c:\client7 -Mbmaster -Iacl40 -o63251 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl402 -o64516 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl40 -o65781 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl402 -o67046 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl40 -o68311 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl402 -o69576 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl40 -o70841 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl402 -o72106 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl40 -o73371 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl402 -o74636 -c1265 -p8080
sleep 80
)
if %COMPUTERNAME%==RTE41 (
start cmd /k c:\client7 -Mbmaster -Iacl41 -o75901 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl412 -o77166 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl41 -o78431 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl412 -o79696 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl41 -o80961 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl412 -o82226 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl41 -o83491 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl412 -o84756 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl41 -o86021 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl412 -o87286 -c1265 -p8080
sleep 80
)
if %COMPUTERNAME%==RTE42 (
start cmd /k c:\client7 -Mbmaster -Iacl42 -o88551 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl422 -o89816 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl42 -o91081 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl422 -o92346 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl42 -o93611 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl422 -o94876 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl42 -o96141 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl422 -o97406 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl42 -o98671 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl422 -o99936 -c1265 -p8080
sleep 80
)
if %COMPUTERNAME%==RTE45 (
```

```
start cmd /k c:\client7 -Mbmaster -Iacl45 -o101201 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl452 -o102466 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl45 -o103731 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl452 -o104996 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl45 -o106261 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl452 -o107526 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl45 -o108791 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl452 -o110056 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl45 -o111321 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl452 -o112586 -c1265 -p8080
sleep 80
)
if %COMPUTERNAME%==RTE46 (
start cmd /k c:\client7 -Mbmaster -Iacl46 -o113851 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl462 -o115116 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl46 -o116381 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl462 -o117646 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl46 -o118911 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl462 -o120176 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl46 -o121441 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl462 -o122706 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl46 -o123971 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl462 -o125236 -c1265 -p8080
sleep 80
)
if %COMPUTERNAME%==RTE47 (
start cmd /k c:\client7 -Mbmaster -Iacl47 -o126501 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl472 -o127766 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl47 -o129031 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl472 -o130296 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl47 -o131561 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl472 -o132826 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl47 -o134091 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl472 -o135356 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl47 -o136621 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl472 -o137886 -c1265 -p8080
sleep 80
)
if %COMPUTERNAME%==RTE48 (
start cmd /k c:\client7 -Mbmaster -Iacl48 -o139151 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl482 -o140416 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl48 -o141681 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl482 -o142946 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl48 -o144211 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl482 -o145476 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl48 -o146741 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl482 -o148006 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl48 -o149271 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl482 -o150536 -c1265 -p8080
sleep 80
```

```
)
if %COMPUTERNAME%==RTE51 (
start cmd /k c:\client7 -Mbmaster -Iacl51 -o151801 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl512 -o153066 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl51 -o154331 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl512 -o155596 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl51 -o156861 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl512 -o158126 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl51 -o159391 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl512 -o160656 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl51 -o161921 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl512 -o163186 -c1265 -p8080
sleep 80
)
if %COMPUTERNAME%==RTE52 (
start cmd /k c:\client7 -Mbmaster -Iacl52 -o164451 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl522 -o165716 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl52 -o166981 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl522 -o168246 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl52 -o169511 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl522 -o170776 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl52 -o172041 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl522 -o173306 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl52 -o174571 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl522 -o175836 -c1265 -p8080
sleep 80
)
if %COMPUTERNAME%==RTE53 (
start cmd /k c:\client7 -Mbmaster -Iacl53 -o177101 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl532 -o178366 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl53 -o179631 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl532 -o180896 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl53 -o182161 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl532 -o183426 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl53 -o184691 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl532 -o185956 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl53 -o187221 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl532 -o188486 -c1265 -p8080
sleep 80
)
if %COMPUTERNAME%==RTE54 (
start cmd /k c:\client7 -Mbmaster -Iacl54 -o189751 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl542 -o191016 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl54 -o192281 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl542 -o193546 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl54 -o194811 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl542 -o196076 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl54 -o197341 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl542 -o198606 -c1265 -p8080
sleep 80
start cmd /k c:\client7 -Mbmaster -Iacl54 -o199871 -c1265 -p8080
sleep 80
```

```
start cmd /k c:\client7 -Mbmaster -Iacl542 -o201136 -c1265 -p8080
sleep 80
)
```

Appendix D : Space Calculation

60 Day Space

TPM	254472
Populated Warehouses	21600
Warehouses	20240

SEGMENT	TYPE	BLOCK_SIZE	BLOCKS	FIVE_PCT	DAILY_GROW	**BLOCKS (MB)**	**FIVE_PCT (MB)**	**DAILY_GROW (MB)**
CUSTCLUSTER	CLUSTER	2048	227054744	11352737	0	443,466	22,173	0
DB_STAT	SYS	2048	1048576	0	0	2,048	0	0
DISTCLUSTER	CLUSTER	2048	225152	11258	0	440	22	0
HIST	TABLE	2048	35750005	0	6,738,796	69,824	0	13,162
ICUST1	INDEX	16384	884616	44231	0	13,822	691	0
ICUST2	INDEX	2048	17531776	876589	0	34,242	1,712	0
IDIST	INDEX	2048	54512	2726	0	106	5	0
IITEM	INDEX	2048	5632	282	0	11	1	0
IORDR2	INDEX	2048	20648064	1032403	0	40,328	2,016	0
ISTOK	INDEX	16384	2658776	132939	0	41,543	2,077	0
ITEMCLUSTER	CLUSTER	2048	8446	422	0	16	1	0
IWARE	INDEX	2048	14012	701	0	27	1	0
NORDCLUSTER_QUEUE	CLUSTER	2048	2999296	149965	0	5,858	293	0
ORDRCLUSTER_QUEUE	CLUSTER	16384	63893700	0	12,043,820	998,339	0	188,185
ROLL_SEG	SYS	8192	9326592	0	0	72,864	0	0
STOKCLUSTER	CLUSTER	2048	308913333	15445667	0	603,346	30,167	0
SYSAUX	SYS	2048	61440	0	0	120	0	0
SYSTEM	SYS	2048	204800	0	0	400	0	0
SYS_IQ0000008805$$	INDEX	16384	141900	7095	0	2,217	111	0
SYS_IQ0000009134$$	INDEX	2048	749824	37491	0	1,465	73	0
WARECLUSTER	CLUSTER	2048	22976	1149	0	45	2	0
Total						2330529.086	59346.69922	201346.4067

Dynamic space (MB)	1068163.29	Initial MB for (Hist+Ordrcluster)
Static space (MB)	1321712.49	
Free space (MB)	201346.41	
Daily growth (MB)	201346.41	
Daily spread (MB)	0.00	Oracle may be configured such that daily spread is 0
60-day space (MB)	13402496.89	
60-day space (GB)	13088.38	
Log size (MB)	940000.00	
Log used (KB)	301934462.71	
Log used/N_O txn. (KB)	4.97	Log used per New-Order transactions
8-hour log (GB)	578.59	

os,file,swap	33.87

	Disk size(GB)	Priced Qty	Priced(GB)	Needed(GB)	Extra(GB)
Database, sys	33.26	525	17461.50	13088.38	4373.12
	33.87	1	33.87		
RAID5 log	33.26	45	1496.70	619.92	876.78

Appendix E : Price Quotation

Date: Fri, 06 Jan 2006 10:29:36 -0800

Subject: Price quote

Product	Price	Quantity	Extended price
Oracle Database 10g Enterprise Edition, Processor License for 3 years	20,000	8	160,000
Oracle Database Server Support Package for 3 years	6,000	1	6,000
Oracle Mandatory E-Business Discount (license and support)			<24,900>

Oracle Corp. pricing contact: MaryBeth Pierantoni, mary.beth.pierantoni@oracle.com, 916-315-5081

redhat

Red Hat Store
Shopping Cart
Contracts

	List	Contract price	Change quantity	Remove	Line total
New Contract	Jan. 10, 2006	Jan. 9, 2007			
Red Hat Enterprise Linux AS (Standard for x86, AMD64 & Intel EM64T, Intel Itanium, and POWER)	$1,499.00	$1,499.00	3	☐	$4,497.00
Red Hat Enterprise Linux ES (Standard for x86, AMD64 & Intel EM64T, and Intel Itanium)	$799.00	$799.00	48	☐	$38,352.00
Add to this contract:		Add			

Browse store

Subtotal: $42,849.00

License Agreement

Required - You must agree to the following before proceeding to checkout.

☐ I have read and agreed to the terms of the Red Hat Enterprise Linux AS (Standard for x86, AMD64 & Intel EM64T, Intel Itanium, and POWER) agreement.

☐ I have read and agreed to the terms of the Red Hat Enterprise Linux ES (Standard for x86, AMD64 & Intel EM64T, and Intel Itanium) agreement.

Install discs and documentation

Optional - You may add the following to your order.

☐ RHEL AS (v.4 for 32-bit x86 and 64-bit AMD64 & Intel EM64T) Media Kit $0

☐ RHEL ES (v.4 for 32-bit x86 and 64-bit AMD64 & Intel EM64T) Media Kit $0

☐ RHEL AS (v.4 for Intel Itanium) Media Kit $0

☐ RHEL ES (v.4 for Intel Itanium) Media Kit $0

☐ RHEL AS (v.4 for POWER) Media Kit $0

Add Promotion Code:

Apply changes | Continue shopping | Cancel order | Checkout

Order History | Order Methods | About Security | Shipping Info | Return Policy | European Store

THE ECOMMERCE TRANSACTION PLATFORM

January 9, 2006

Masato Kataoka
Middle Software Division,
NEC Corporation
4-14-22 Shibaura,Minato-ku
Tokyo,108-8558,Japan

Per your request I am enclosing the pricing information regarding TUXEDO 8.1 that you requested. This pricing applies to Tuxedo 6.4, 6.5, 7.1,8.0,8.1, and 9.0. Please note that Tuxedo 8.1 is our most recent version of Tuxedo. Core functionality services (CFS)-R pricing is appropriate for your activities. NEC systems are classified as either a Tier 1, 2, 3, 4 or 5 systems depending on the performance and CPU capacity of the system. The NEC Express5800/120Rg-2 are Tier 1 machines – price is $1,200 per server (License), eligible for a 5% discount = $1,140 per server + $252 per server (7x24) for support – support is non discountable. This quote is valid for 60 days from the date of this letter.

Tuxedo Core Functionality Services (CFS-R) Program Product Pricing and Description

TUX-CFS-R provides a basic level of middleware support for distributed computing, and is best used by organizations with substantial resources and knowledge for advanced distributed computing implementations.

TUX-CFS-R prices are server only and are based on the overall performance characteristics of the server and uses the same five tier computer classification as TUXEDO 6.4, 6.5, 7.1,8.0, 8.1,and 9.0. Prices range from $1,200 for Tier 1 to $100,000 for Tier 5. Under this pricing option EVERY system running TUX-CFS-R at the user site must have a TUXEDO license installed and pay the appropriate per server license fees.

Very Truly Yours,

Robert J. Gieringer

Rob Gieringer,
Worldwide Pricing Manager

1

BEA Tux/CFS-R Unlimited User License Fees Per Server

Unlimited User License fees per server	Number of Users	Dollar Amount	Maintenance (5 x 9) per year	Maintenance (7 x 24) per year
Tier 1 -- PC Servers with 1 or 2 CPUs, entry level RISC Uni-processor workstations and servers	Unlimited	$1,200.00	$216	$252
Tier 2 - PC Servers with 3 or 4 CPUs, Midrange RISC Uni-processor servers and workstations with up to 2 CPUs	Unlimited	$4,800.00	$864	$1,008
Tier 3 - Midrange Multiprocessors, up to 8 CPUs per system capacity	Unlimited	$12,000.00	$2,160	$2,520
Tier 4 - Large (more than 8, less than 32 CPUs)	Unlimited	$40,000.00	$7,200	$8,400
Tier 5 - Massively Parallel Systems, > 32 processors	Unlimited	$100,000.00	$18,000	$21,000

CDW 800.750.4239

SHOPPING CART

Your Saved Carts | Save This Cart | Edit Saved Carts | Send To An Associate

Continue to Checkout

Quantity	Product	CDW	Usually Ships	Price	Ext. Price
3	Allied Telesyn AT GS924 - switch - 24 ports	806699	1-3 Days	$344.99	$1,034.97
1	Intel Pro/100+ Adapter	191126	3-6 Days	$643.99	$643.99
6	NEC AccuSync 500	573757	Same Day	$119.95	$719.70
12	QLogic SANblade 2340 2Gb 133MHz PCI-X Single Channel HBA	408081	Same Day	$951.99	$11,423.88
36	Tripp Lite 10' Gray Cat5e or Cat5 Snagless Crossover Cable 10ft	324527	Same Day	$6.48	$233.28
21	Tripp Lite 25' Gray Cat5e or Cat5 RJ45 Molded 350mhz UTP Patch Cable	324516	Same Day	$7.49	$157.29

Click to remove an item from your cart Sub-Total $14,213.11

Update Clear Cart Continue to Checkout

Continue Shopping | Go to CDW.com Homepage

Shipping Calc:

Enter a postal code to quickly estimate shipping cost.

QuickCart:

Enter a **CDW part number** to quickly add it to your cart.

* Sample: CDW Part #

http://www.cdw.com/shop/cart/default.aspx?printable=1 1/10/2006