

**IBM RISC System/6000  
Enterprise Server S80 c/s**  
*using*  
**Oracle8 Enterprise Edition 8.1.6**  
**IBM TXSeries 4.2 for AIX**

---

**TPC Benchmark<sup>TM</sup> C**

**Full Disclosure Report**

IBM System Performance and Evaluation Center

Submitted For Review  
October 29, 1999



The Oracle logo, consisting of the word "ORACLE" in a red, sans-serif font, enclosed within a red rectangular border.

### *Special Notices*

The following terms used in this publication are trademarks of **International Business Machines** Corporation in the United States and/or other countries:

RISC System/6000  
AIX  
IBM

The following terms used in this publication are trademarks of other companies as follows:

TPC Benchmark	Trademark of the Transaction Processing Performance Council
ORACLE, SQL*Loader	Trademark of Oracle, Inc.
Oracle8,SQL*Net and SQL*Plus	Trademark of Oracle, Inc.

### **First Edition October 29, 1999**

The information contained in this document is distributed on an AS IS basis without any warranty either expressed or implied. The use of this information or the implementation of any of these techniques is a customer's responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment.

While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used.

It is possible that this material may contain references to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such products, programming, or services in your country.

All performance data contained in this publication was obtained in a controlled environment, and therefore the results which may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data in their specific environment.

Request for additional copies of this document should be sent to the following address:

TPC Benchmark Administrator  
IBM Commercial Performance  
Mail Stop 9571  
11400 Burnet Road  
Austin, TX 78758  
FAX Number (512) 838-1852

© **Copyright International Business Machines Corporation 1999. All rights reserved.**

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

**NOTE:** US. Government Users - Documentation related to restricted rights: Use, duplication, or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.



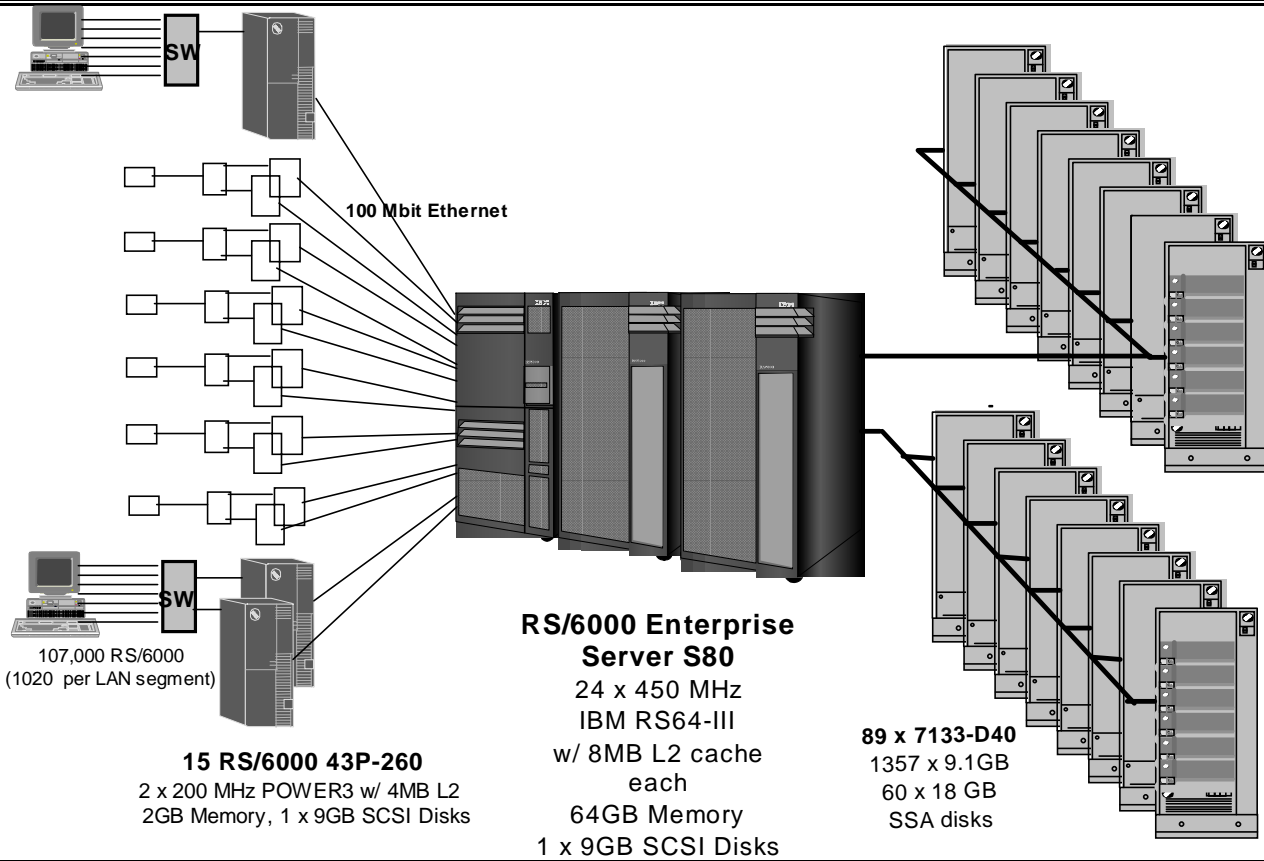
# IBM RS/6000 Enterprise Server S80 c/s

TPC-C Rev. 3.4

ORACLE

Report Date:  
Oct. 29, 1999

<b>Total System Cost</b>	<b>TPC-C Throughput</b>	<b>Price/Performance</b>	<b>Availability Date</b>	
<b>\$7,067,644</b>	<b>135,815.7 tpmC</b>	<b>\$52.04/tpmC</b>	<b>March 1, 2000</b>	
<b>Processors</b>	<b>Database Manager</b>	<b>Operating System</b>	<b>Other</b>	<b>No. Users</b>
24 x IBM RS64-III	Oracle8 Enterprise Edition 8.1.6	AIX 4.3.3 +APAR IY05431	TXSeries 4.2 for AIX	107,000



System Components	Clients		Server	
	Quantity	Description	Quantity	Description
<b>Processor</b>	15	2 x 200 MHz POWER3 w/ 4MB L2 cache each	24	450 MHz IBM RS64-III w/ 8MB L2 cache each
<b>Memory</b>		2 GB		64 GB
<b>Disk Controllers</b>	15	SCSI-2 Adapters	2	SCSI-2 Adapters
			18	SSA Adapters
<b>Disk Drives</b>	15	9.1 GB SCSI	1,357	9.1 GB SSA Disks
			60	18 GB Disk
			1	9.1 GB SCSI Disk
<b>Total Storage</b>		9.1 GB each client		12,358.4 GB
<b>Terminals</b>	15	System Console	1	System Console
<b>Term. Connect</b>	13,440	8-Port Ethernet Hubs		



# IBM RS/6000

TPC-C Rev. 3.4

ORACLE

## Enterprise Server S80 c/s

Report Date: Oct. 29, 1999

Description	Part No.	Source	Unit Price	Qty	Ext Price	Maint Price
<b>Server Hardware</b>						
RS/6000 Server Model S80	7017-S80		11,538	1	11,538	45,168
16bit PCISCSI cables	2447		75	1	75	0
32x CD-ROM drive	2624		375	1	375	0
9.1 GB 1" Ultra SCSI Hot Swap	2913		1,260	1	1,260	0
Async/Terminal Cable	2934		45	1	45	0
10/100 Mbps Ethernet PCI Adapter	2968		275	16	4,400	0
Remote I/O Cable (3142, 3143) set	3142		4,170	1	4,170	0
16384 MB (4x4096) R1 Memory	4194		147,456	4	589,824	0
6way RS64 III 450 Mhz, 8MB L2	5318		200,000	4	800,000	297,408
I/O Power Cable set (6006,7,8)	6000		685	1	685	0
1000 Watt AC Power Supply	6193		765	1	765	0
Power Regulators (6194, 6195)	6194		2,505	1	2,505	0
Ultra SCSI PCI-Bus Adapter	6206		395	2	790	0
Advanced Serial RAID Adapter	6225		3,000	18	54,000	0
32 MB FastWrite Cache Option	6235		575	3	1,725	0
SCSI I/O Drawer, 10 EIA	6320		23,766	4	95,064	0
Primary I/O Drawer Group	6321		600	1	600	0
Secondary I/O Drawer Group	6323		734	3	2,202	0
Support Processor Group	6326		2,700	1	2,700	0
Remote I/O Hub, Dual Loop	6503		8,000	1	8,000	0
SCSI 6-pack Hot Swap Back Plane	6547		600	1	600	0
I/O Rack	7000		3,500	2	7,000	0
Prestige 6000VA, 208-240V UPS	9910-EP8		5,505	1	5,505	0
Prestige EXT UPS, Rack Mount	9910-U33		3,822	2	7,644	0
System Rack Model S00	7014-S00		3,500	1	3,500	1,536
System Rack Model T00	7014-T00		3,550	10	35,500	15,360
Additional Power Distribution Unit	6171		1,000	21	21,000	0
SSA Disk Subsystem, Black Cover	7133-D40		13,000	89	1,157,000	786,048
50/60 Hz AC, 300 VDC Power Supply	8022		2,000	89	178,000	0
9.1 GB Disk Drive Modules	8209		2,760	1,357	3,745,320	0
18 GB Disk Drive Modules	8209		4,500	60	270,000	0
SSA Cables	8802		65	178	11,570	0
			<b>Subtotal</b>		<b>7,023,362</b>	<b>1,145,520</b>
<b>Server Software</b>						
AIX 4.3.3 for S80	5765-C34		50	1	50	40,920
IBM C for AIX V4.4	5765-C64		799	1	799	30,480
Oracle8 Enterprise Edition 8.1.6 64bit		1	695,970	1	695,970	695,970
			<b>Subtotal</b>		<b>696,819</b>	<b>767,370</b>
<b>Client Hardware</b>						
RS/6000 Model 43P-260	7043-260		3,762	15	56,430	101,520
32x CD-ROM	2624		375	15	5,625	0
Async Terminal/Printer Cable	2934		45	15	675	0
10/100 Mbps Ethernet Adapter, PCI	2968		275	30	8,250	0
9.1 GB SCSI Hard Disk	3029		1,260	30	37,800	0
Memory Expansion Feature	4098		1,038	15	15,570	0
512MB SDRAM DIMM Memory	4119		5,760	60	345,600	0
200 MHz Power3 Processor, 4MB L2	4342		8,000	30	240,000	0
IBM ASCII Terminal, Keyboard, Cable	3153-BG3		577	16	9,232	9,728
			<b>Subtotal</b>		<b>719,182</b>	<b>111,248</b>
<b>Client Software</b>						
AIX 4. Unlimited Users	5756-C34		50	15	750	24,900
IBM Websphere Application Server	41L3462		10,500	15	157,500	25,513
			<b>Subtotal</b>		<b>158,250</b>	<b>50,413</b>
<b>User Connectivity</b>						
8port Ethernet Switch + 2 Spare	ES-3108P	2	79	20	1,580	6,240
Ethernet Hub (8port) + 10% Spares	CT1008D/P	3	18.00	14,784	266,112	0
			<b>Subtotal</b>		<b>267,692</b>	<b>6,240</b>
			<b>Discounts</b>		<b>(3,555,739)</b>	<b>(322,713)</b>
			<b>Total</b>		<b>5,309,566</b>	<b>1,758,078</b>

Notes:

Pricing Sources:

Five-Year Cost of Ownership:

7,067,644

1=Oracle, 2=PC Connection, 3=NetCruiser Technologies, Inc

tpmC 135,815.70

Audited by: Francois Raab, Info Sizing

\$/tpmC 52.04

Prices used in TPC benchmarks reflect the actual prices a customer would pay for a one-time purchase of the stated components. Individually negotiated discounts are not permitted. Special prices based on assumptions about past or future purchases are not permitted. All discounts reflect standard pricing policies for the listed components. For complete details, see the pricing sections of the TPC benchmark specifications. If you find that the stated prices are not available according to these terms, please inform the TPC at [pricing@tpc.org](mailto:pricing@tpc.org). Thank you

## Numerical Quantities Summary for the IBM RS/6000 S80

MQTH, computed Maximum Qualified Throughput: 135,815.70 tpmC

Repeatability Run: 135,537.07 tpmC 0.2% difference

<u>Response Times (in seconds)</u>	<u>90th %</u>	<u>Average</u>	<u>Maximum</u>
New Order	0.54	0.26	7.55
Payment	0.48	0.22	7.29
Order-Status	0.55	0.26	5.11
Delivery (interactive portion)	0.19	0.11	1.61
Delivery (deferred portion)	0.22	0.13	3.52
Stock-Level	0.85	0.52	7.55
Menu	0.01	0.01	3.59

### Transaction Mix, in percent of total transactions

	<u>Percent</u>
New Order	44.96%
Payment	43.00%
Order-Status	4.01%
Delivery	4.00%
Stock-Level	4.01%

<u>Keying/Think Times (in seconds)</u>	<u>Min.</u>	<u>Average</u>	<u>Max.</u>
New Order	18.00/0.01	18.01/12.01	18.18/120.11
Payment	3.00/0.01	3.01/12.00	3.19/120.11
Order-Status	2.00/0.01	2.01/10.01	2.16/100.10
Delivery	2.00/0.01	2.01/5.02	2.17/50.10
Stock-Level	2.00/0.01	2.01/5.00	2.17/50.10

### Test Duration

Ramp-up Time	52 min 14 sec
Measurement interval	30 minutes
Transactions during measurement interval (all types)	9,061,152
Ramp-down time	15 minutes

### Checkpointing

Number of checkpoints	1
Checkpoint interval	29 min 51 sec

---

**Abstract**

This report documents the full disclosure information required by the TPC Benchmark™ C Standard Specification Revision 3.4 dated August 25, 1998, for measurements on the IBM RISC System/6000 Enterprise Server Model S80. The phrase RS/6000 will be substituted for RISC System/6000 for the remainder of this document.

The software used on the RS/6000 Enterprise Server S80 includes AIX Version 4.3.3 operating system, Oracle8 Server database manager, and TXSeries 4.2 for AIX transaction manager.

**IBM RISC System/6000 Enterprise SMP Server S80**

<b>Company Name</b>	<b>System Name</b>	<b>Data Base Software</b>	<b>Operating System Software</b>
IBM Corporation Oracle Corporation	RS/6000 Enterprise Server S80	Oracle8 Enterprise Edition 8.1.6	AIX Version 4.3.3 +APAR IY05431

<b>Total System Cost</b>	<b>TPC-C Throughput</b>	<b>Price/Performance</b>
-Hardware -Software -5 Years Maintenance	Sustained maximum throughput of system running TPC-C expressed in transactions per minute	Total system cost/tpmC
<b>\$7,067,644</b>	<b>135,815.70 tpm-C</b>	<b>\$52.04 per tpm-C</b>

---

## Preface

TPC Benchmark™ C Standard Specification was developed by the Transaction Processing Performance Council (TPC). It was released on August 13, 1992 and updated with revision 3.4 on August 25, 1998.

This is the full disclosure report for benchmark testing of the IBM RS/6000 Enterprise server S80 according to the TPC Benchmark™ C Standard Specification.

TPC Benchmark™ C exercises the system components necessary to perform tasks associated with that class of on-line transaction processing (OLTP) environments emphasizing a mixture of read-only and update intensive transactions. This is a complex OLTP application environment exercising a breadth of system components associated by such environments characterized by:

- v The simultaneous execution of multiple transaction types that span a breadth of complexity
- v On-line and deferred transaction execution modes
- v Multiple on-line terminal sessions
- v Moderate system and application execution time
- v Significant disk input/output
- v Transaction integrity (ACID properties)
- v Non-uniform distribution of data access through primary and secondary keys
- v Data bases consisting of many tables with a wide variety of sizes, attributes, and relationships
- v Contention on data access and update

This benchmark defines four on-line transactions and one deferred transaction, intended to emulate functions that are common to many OLTP applications. However, this benchmark does not reflect the entire range of OLTP requirements. 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 benchmarks when critical capacity planning and/or product evaluation decisions are contemplated.

The performance metric reported by TPC-C is a “business throughput” measuring the number of orders processed per minute. Multiple transactions are used to simulate the business activity of processing an order, and each transaction is subject to a response time constraint. The performance metric for this benchmark is expressed in transactions-per-minute-C (tpmC). To be compliant with the TPC-C standard, all references to tpmC results must include the tpmC rate, the associated price-per-tpmC, and the availability date of the priced configuration.

---

# 1. General Items

---

## 1.1 Application Code Disclosure

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

Appendix A contains the RS/6000 application code for the five TPC Benchmark™ C transactions. Appendix D contains the terminal functions and layouts.

---

## 1.2 Benchmark Sponsor

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

This benchmark was sponsored by **International Business Machines Corporation** and **Oracle Corporation**.

---

## 1.3 Parameter Settings

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

- √ Data Base tuning options*
- √ Recovery/commit options*
- √ Consistency/locking options*
- √ Operating system and application configuration parameters.*

Appendix B contains the system, data base, and application parameters changed from their default values used in these TPC Benchmark™ C tests.

---

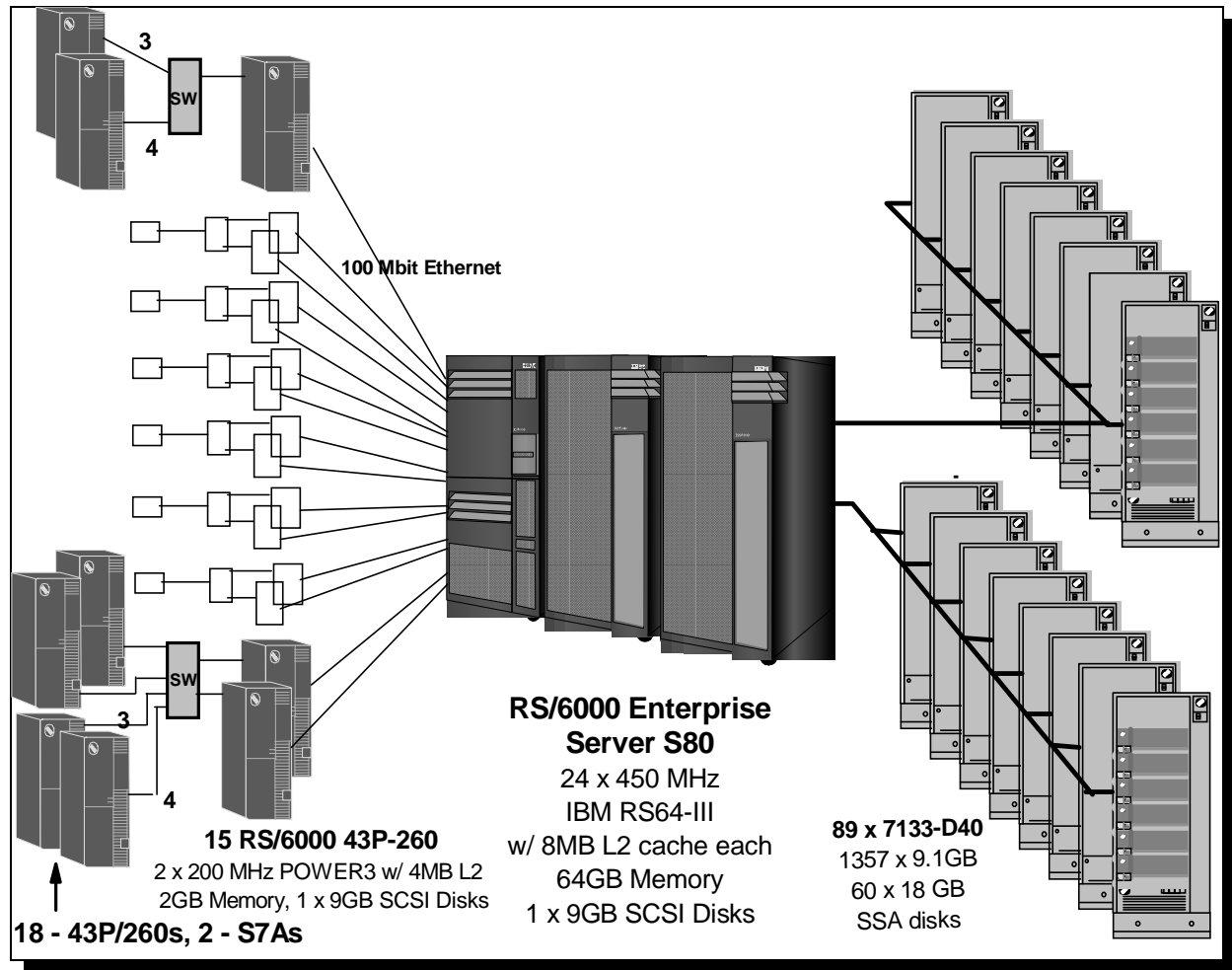
## 1.4 Configuration Diagrams

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

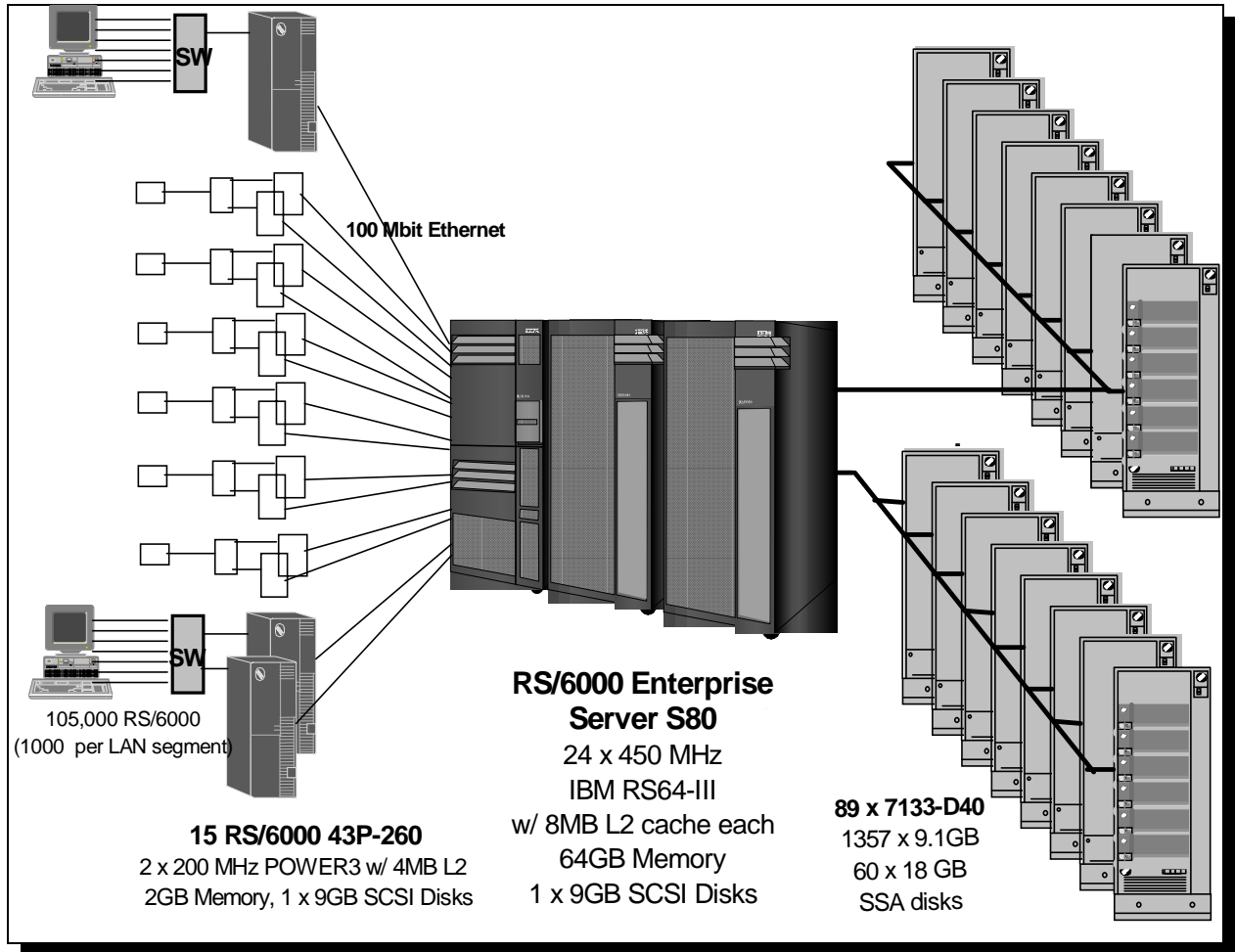
- √ Number and type of processors*
- √ Size of allocated memory, and any specific mapping/partitioning of memory unique to the test*
- √ Number and type of disk units (and controllers, if applicable)*
- √ Number of channels or bus connections to disk units, including the protocol type*
- √ Number of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



# RISC System/6000 Enterprise Server S80 Benchmark Configuration



# RISC System/6000 Enterprise Server S80 Priced Configuration



---

## 2. Clause 1: Logical Data Base Design Related Items

---

### 2.1 Table Definitions

*Listings must be provided for all table definition statements and all other statements used to setup the data base.*

Appendix C contains the table definitions and the database load programs used to build the data base.

---

### 2.2 Database Organization

*The physical organization of tables and indices, within the data base, must be disclosed.*

Physical space was allocated to Oracle8 Server on the server disks according to the details provided in Appendix C. The size of the space segments on each disk was calculated to provide even distribution of data across the disk subsystem.

---

### 2.3 Insert and/or 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 data base 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.*

There were no restrictions on insert and/or delete operations to any of the tables. The space required for an additional five percent of the initial table cardinality was allocated to Oracle8 Server and priced as static space.

---

### 2.4 Horizontal or Vertical Partitioning

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

Partitioning was not used for any of the measurement reported in this full disclosure.

---

## 3. Clause 2: Transaction and Terminal Profiles Related Items

---

### 3.1 Verification for the Random Number Generator

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

The `random()`, `getpid()` and `gettimeofday()` functions are used to produce unique random seeds for each driver. The drivers use these seeds to seed the `srand()`, `random()` and `srand48()` functions. Random numbers are produced using wrappers around the standard system random number generators.

The negative exponential distribution uses the following function to generate the distribution. This function has the property of producing a negative exponential curve with a specified average and a maximum value 4 times the average.

```
const double RANDOM_4_Z = 0.89837799236185
const double RANDOM_4_K = 0.97249842407114

double neg_exp_4(double average {
    return - average * (1/RANDOM_4_Z * log (1 - RANDOM_4_K * drand48()));
})
```

The random functions used by the driver system and the data base generation program were verified. The `C_LAST` column was queried to verify the random values produced by the database generation program. After a measurement, the `HISTORY`, `ORDER`, and `ORDER_LINE` tables were queried to verify the randomness of values generated by the driver. The rows were counted and grouped by customer and item numbers.

Here is an example of one SQL query used to verify the random number generation functions:

```
v create table TEMP (W_ID int, D_ID, C_LAST char(16), CNTR int);

v insert into TEMP select C_W_ID, C_D_ID, C_LAST, COUNT(*) from CUSTOMER group by C_W_ID,
    C_D_ID, C_LAST;

v select CNTR, COUNT(*) from TEMP group by CNTR order by 1;
```

---

### 3.2 Input/Output Screens

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

The screen layouts corresponds exactly to the layout corresponding in clauses 2.4.3, 2.5.3, 2.6.3, 2.7.3 and 2.8.3 of the TPC-C specifications.

---

### 3.3 Priced Terminal Features

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

The emulated workstations, IBM RS/6000 Model 43P-100s, are commercially available and support all of the requirements in Clause 2.2.2.4.

---

### **3.4 Presentation Managers**

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

The RS/6000 Model 43P-100 workstations did not involve screen presentations, message bundling or local storage of TPC-C rows. All screen processing was handled by the client system. All data manipulation was handled by the server system.

---

### **3.5 Home and Remote Order-lines**

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

Table 3-1 show the percentage of home and remote transactions that occurred during the measurement period for the New-Order transactions.

---

### **3.6 New-Order Rollback Transactions**

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

Table 3-1 show the percentage of New-Order transactions that were rolled back due to an illegal item being entered.

---

### **3.7 Number of Items per Order**

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

Table 3-1 show the average number of items ordered per New-Order transaction.

---

### **3.8 Home and Remote Payment Transactions**

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

Table 3-1 show the percentage of home and remote transactions that occurred during the measurement period for the Payment transactions.

---

### **3.9 Non-Primary Key Transactions**

*The percentage of Payment and Order-Status transactions that used non-primary key (C\_LAST) access to the data base must be disclosed.*

Table 3-1 show the percentage of non-primary key accesses to the data base by the Payment and Order-Status transactions.

---

### **3.10 Skipped Delivery Transactions**

*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 3-1 show the percentage of Delivery transactions missed due to a shortage of supply of rows in the NEW-ORDER table.

---

### **3.11 Mix of Transaction Types**

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

Table 3-1 show the mix percentage for each of the transaction types executed by the SUT.

---

### **3.12 Queuing Mechanism of Delivery**

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

The Delivery transaction was submitted using an RPC call to an IBM TXSeries version 4.2, Encina interface transaction manager (TM). TXSeries returns an immediate response to the calling program and schedules the work to be performed. This allows the Delivery transaction to be submitted, obtain an interactive response and queue the actual data base transaction for deferred execution. Please see the application code in Appendix A for details.

**Table 3-1 Numerical Quantities for Transaction and Terminal Profiles**

<b>New Order</b>	<b>RS/6000 Enterprise Server S80</b>
Percentage of Home order lines	99.0%
Percentage of Remote order lines	1.0%
Percentage of Rolled Back Transactions	1.00%
Average Number of Items per order	10
<b>Payment</b>	
Percentage of Home transactions	84.99%
Percentage of Remote transactions	15.01%
<b>Non-Primary Key Access</b>	
Percentage of Payment using C_LAST	60.01%
Percentage of Order-Status using C_LAST	59.97%
<b>Delivery</b>	
Delivery transactions skipped	0
<b>Transaction Mix</b>	
New-Order	44.96%
Payment	43.00%
Order-Status	4.01%
Delivery	4.00%
Stock-Level	4.01%

---

## 4. Clause 3: Transaction and System Properties

*The results of the ACID test must be disclosed along with a description of how the ACID requirements were met.*

All ACID tests were conducted according to specification.

---

### 4.1 Atomicity Requirements

*The system under test must guarantee that data base 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.*

---

#### 4.1.1 Atomicity of Completed Transaction

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

The following steps were performed to verify the Atomicity of completed transactions.

1. The balance was retrieved from the CUSTOMER table for a random Customer, District and Warehouse giving BALANCE\_1.
2. The Payment transaction was executed for the Customer, District and Warehouse used in step 1
3. The balance was retrieved again for the Customer used in step 1 and step 2 giving BALANCE\_2. It was verified that BALANCE\_1 was greater than BALANCE\_2 by AMT.

---

#### 4.1.2 Atomicity of Aborted Transactions

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

The following steps were performed to verify the Atomicity of the aborted Payment transaction:

1. The Payment application code was changed to execute a rollback of the transaction instead of performing the commit.
2. Using the balance, BALANCE\_2, from the CUSTOMER table retrieved for the completed transaction, the Payment transaction was executed for the Customer, District, and Warehouse used in step 1 of the section 4.1.1, using a payment amount (AMT) of 410.00. The transaction rolled back due to the change in the application code from step 1.
3. The balance was retrieved again for the Customer used for step 2 giving BALANCE\_3. It was verified that BALANCE\_2 was equal to BALANCE\_3.

---

## 4.2 Consistency Requirements

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

*Verify that the data base is initially consistent by verifying that it meets the consistency conditions defined in Clauses 3.3.2.1 to 3.3.2.4. Describe the steps used to do this in sufficient detail so that the steps are independently repeatable.*



---

### 4.2.1 Consistency Condition 1

Entries in the *WAREHOUSE* and *DISTRICT* tables must satisfy the relationship:

$$\forall W\_YTD = \text{sum}(D\_YTD)$$

for each warehouse defined by ( $W\_ID = D\_W\_ID$ )

---

### 4.2.2 Consistency Condition 2

Entries in the *DISTRICT*, *ORDER*, and *NEW-ORDER* tables must satisfy the relationship:

$$\forall D\_NEXT\_O\_ID - 1 = \text{max}(O\_ID) = \text{max}(NO\_O\_ID)$$

for each district defined by ( $D\_W\_ID = O\_W\_ID = NO\_W\_ID$ ) and ( $D\_ID = O\_D\_ID = NO\_D\_ID$ ). This condition does not apply to the *NEW-ORDER* table for any districts which have no outstanding new orders.

---

### 4.2.3 Consistency Condition 3

Entries in the *New-Order* table must satisfy the relationship:

$$\forall \text{max}(NO\_O\_ID) - \text{min}(NO\_O\_ID) + 1 = [\text{number of rows in the New-Order table for this district}]$$

for each district defined by  $NO\_W\_ID$  and  $NO\_D\_ID$ . This condition does not apply to any districts which have no outstanding new orders.

---

### 4.2.4 Consistency Condition 4

Entries in the *ORDER* and *ORDER-LINE* tables must satisfy the relationship:

$$\forall \text{sum}(O\_OL\_CNT) = [\text{number of rows in the ORDER-LINE table for this district}]$$

for each district defined by ( $O\_W\_ID = OL\_W\_ID$ ) and ( $O\_D\_ID = OL\_D\_ID$ ).

---

### 4.2.5 Consistency Tests

Verify that the data base is initially consistent by verifying that it meets the consistency conditions defined in Clauses 3.3.2.1 to 3.3.2.4. Describe the steps used to do this in sufficient detail so that the steps are independently repeatable.

The consistency conditions defined in 4.2.1 through 4.2.4 were tested using a shell script to issue queries to the database. All queries showed that the data base was in a consistent state.

After executing transactions at full load for approximately sixty minutes the shell script was executed again. All queries show that the database was still in a consistent state.

---

## 4.3 Isolation Requirements

Operations of concurrent data base transactions must yield results which are indistinguishable from the results which would be obtained by forcing each transaction to be serially executed to completion in some order.

---

### 4.3.1 Isolation Test 1

This test demonstrates isolation for read-write conflicts of *Order-Status* and *New-Order* transactions.

1. An Order status transaction T0 was executed for a randomly selected customer, and the order returned was as recorded. Transaction T0 was committed.
2. A new-order transaction T1 was started for the same customer used in T0. T1 was stopped immediately prior to commit.
3. An order-status transaction T2 was started for the same customer used in T1. Transaction T2 completed and was committed without being blocked by T1. T2 returned the same order that T0 had returned.
4. T1 completed and was committed.
5. An order-status transaction T3 was started for the same customer used in T1. T3 returned the order inserted by T1.

This result demonstrates serialization of T2 before T1. It has equivalent validity to the outcome specified in the Standard which supposes T1 to be serialized before T2.

---

### 4.3.2 Isolation Test 2

*This test demonstrates isolation for read-write conflicts of Order-Status and New-Order transactions when the New-Order transaction is rolled back.*

The following steps were performed to satisfy the test of isolation for Order-Status and a rolled back New-Order transactions:

1. An Order status transaction T0 was executed for a randomly selected customer, and the order returned was recorded. Transaction T0 was committed.
2. A new-order transaction T1 with an invalid item was started for the same customer used in T0. Transaction T1 was stopped prior to rollback.
3. An order-status transaction T2 was started for the same customer used in T1. T2 completed and was committed without being blocked by T1. Transaction T2 returned the same order that T0 had returned.
4. T1 was rollback.
5. An order-status transaction T3 was started for the same customer used in T1. T3 returned the same order that T0 returned.

---

### 4.3.3 Isolation Test 3

*This test demonstrates isolation for write-write conflicts of two New-Order transactions.*

The following steps were performed to verify isolation of two New-Order transactions:

1. The D\_NEXT\_O\_ID of a randomly selected district was retrieved.
2. A new-order transaction T1 was started for a randomly selected customer within the district used in step 1. T1 was stopped immediately prior to commit.
3. Another new-order transaction was started for the same customer used in T1. Transaction T2 waited.
4. T1 completed. T2 completed and was committed.
5. The order number returned by T1 was the same as the D\_NEXT\_O\_ID retrieved in step 1. The order number returned by T2 was one greater than the order number returned by T1.
6. The D\_NEXT\_O\_ID of the same district was retrieved again. It had been incremented by two (it was one greater than the order number returned by T2).

---

#### 4.3.4 Isolation Test 4

*This test demonstrates isolation for write-write conflicts of two New-Order transactions when one transaction is rolled back.*

The following steps were performed to verify the isolation of two New-Order transactions after one is rolled back:

1. The D\_NEXT\_O\_ID of a randomly selected district was retrieved.
2. A new-order transaction T1 with an invalid item was started for a randomly selected customer with the district used in step 1. T1 was stopped immediately prior to rollback.
3. Another new-order transaction was started for the same customer used in T1. T2 waited.
4. T1 was allowed to rollback. T2 completed and was committed.
5. The order number returned by T2 was the same as the D\_NEXT\_O\_ID retrieved in step 1.
6. The D\_NEXT\_O\_ID of the same district was retrieved again. It had been incremented by one (it was one greater than the order number returned by T2).

---

#### 4.3.5 Isolation Test 5

*This test demonstrates isolation for write-write conflicts of Payment and Delivery transactions.*

The following steps were performed to successfully conduct this test:

1. A query was executed to find out the customer who would be updated by the next delivery transaction for a randomly selected warehouse and district.
2. The C\_BALANCE of the customer found in step 1 is retrieved.
3. A delivery transaction T1 was started for the same warehouse used in step 1. T1 was stopped immediately prior to the commit of the database transaction corresponding to the district used in step 1.
4. A payment transaction T2 was started for the same customer found in step 1. T2 waited.
5. T1 was allowed to complete. T2 completed and was committed.
6. The C\_BALANCE of the customer found in step 1 was retrieved again. The C\_BALANCE reflected the results of both T1 and T2.

---

#### 4.3.6 Isolation Test 6

*This test demonstrates isolation for write-write conflicts of Payment and Delivery transactions when the Delivery transaction is rolled back.*

The following steps were performed to successfully conduct this test:

1. A query was executed to find out the customer who would be updated by the next delivery transaction for a randomly selected warehouse and district.
2. The C\_BALANCE of the customer found in step 1 is retrieved
3. A delivery transaction T1 was started for the same warehouse used in step 1. T1 was stopped immediately prior to the rollback of the database transaction corresponding to the district used in step 1.
4. A payment transaction T2 was started for the same customer found in step 1. Transaction T2 waited.
5. T1 was allowed to rollback. T2 completed and was committed.
6. The C\_BALANCE of the customer found in step 1 was retrieved again. The C\_BALANCE reflected the results of only Transaction T2.

---

### 4.3.7 Isolation Test 7

*This test demonstrates repeatable reads for the New-Order transaction while an interactive transaction updates the price of an item.*

The following steps were performed to successfully conduct this test:

1. The I\_PRICE of two randomly selected items were retrieved.
2. A new-order transaction T2 with a group of items X and Y was started. T2 was stopped immediately, after retrieving the prices of all items. The prices of items X and Y retrieved matched those values retrieved in step 1.
3. A transaction T3 was started to increase the price of items X and Y by 10%.
4. T3 did not stall and no transaction was rolled back. T3 was committed.
5. T2 was resumed, and the prices of all items were retrieved again within T2. The prices of items X and Y matched those retrieved in step 1.
6. T2 was committed.
7. The prices of items X and Y were retrieved again. The values matched the values set by T3.

---

### 4.3.8 Isolation Test 8

*This test demonstrates isolation for phantom protection between a Delivery and a New-Order transaction.*

The following steps were performed to successfully conduct this test:

1. The NO\_D\_ID of all new order rows for a randomly selected warehouse and district was changed. The changes were committed.
2. A delivery transaction T1 was started for the selected customer.
3. T1 was stopped immediately after reading the new order table for the selected warehouse and district . No qualifying rows were found.
4. A new order transaction T2 was started for the same warehouse and district. T2 completed and was committed without being blocked by T1.
5. T1 was resumed and the new order table was read again. No qualifying row was found.
6. T1 completed and was committed.
7. The NO\_D\_ID of all new order rows for the selected warehouse and district was restored to the original value. The changes were committed.

---

### 4.3.9 Isolation Test 9

*This test demonstrates isolation for phantom protection between an Order-Status and a New-Order transaction.*

The following steps were performed to successfully conduct this test:

1. An order status transaction T1 was started for a randomly selected customer.
2. T1 was stopped immediately after reading the order table for the selected customer The most recent order for that customer was found.

3. A new order transaction T2 was started for the same customer. T2 completed and was committed without being blocked by T1.
4. T1 was resumed and the order table was read again to determine the most recent order for the same customer. The order found was the same as the one found in step 2.
5. T1 completed and was committed.

---

## 4.4 Durability Requirements

*The tested system must guarantee durability: the ability to preserve the effects of committed transactions and insure data base consistency after recovery from any one of the failures listed in Clause 3.5.3*

---

### 4.4.1 Permanent Unrecoverable Failure of any Single Durable Medium

*Permanent irrecoverable failure of any single durable medium containing TPC-C data base tables or recovery log data.*

***Failure of Durable Medium containing recovery log data and Instantaneous Interruption and Memory Failure.***

This test was conducted on a fully scaled database. The following steps were performed successfully.

1. The current count of the total number of orders was determined by the sum of D\_NEXT\_O\_ID of all rows in the DISTRICT table giving SUM\_1.
2. A test was started and allowed to run for twelve minutes.
3. One of the disks containing the Oracle8 transaction log data was powered off. Since the log was on a raid disk, Oracle8 continued to process the transactions successfully.
4. The test continued for another 1 1/2 minutes.
5. The system was immediately shut down by switching the Emergency Power Off , thereby removing system power.
6. The disk from step 3 was powered back on.
7. The system was powered back on and rebooted.
8. Step 1 is performed returning the value for SUM\_2. It was verified that SUM\_2 was equal to SUM\_1 plus the completed New\_Order transactions recorded by the RTE and that no entries existed for rolled-back transactions.
9. Consistency condition 3 was verified.

***Failure of Durable Medium containing TPC-C data base tables.***

The following steps were successfully performed to pass the Durability test of failure of a disk unit with data base tables:

1. The contents of a disk containing a TPCC table was backed up by copying it to another disk.
2. The current count of the total number of orders was determined by the sum of D\_NEXT\_O\_ID of all rows in the DISTRICT table giving SUM\_1.
3. A scaled-down test was started and allowed to run until steady state.
4. The disk containing the TPCC table was powered off.
5. The run was stopped.
6. The disk from step 4 was powered back on and was restored from the backup copy in step 1.

7. Oracle8 was restarted and its transaction log was used to roll forward through the transactions that had completed since the run had started.
8. Step 2 was performed returning SUM\_2. It was verified that SUM\_2 was equal to SUM\_1 plus the completed New\_Order transactions recorded by the RTE and that no entries existed for rolled-back transactions.
9. Consistency condition 3 was verified.

### ***Failure of Durable Fast Write Cache on SSA Adapter for Redo Logs***

The following steps were successfully performed to pass the Durability test for failure of a durable medium that contains transient redo log transactions:

1. The SSA adapter for the Redo logs contains a cache that is powered by an onboard battery which will retain its contents if the adapter fails, or system power goes off. This test was performed in two parts:
  - A) failure of the adapter/system power
  - B) failure of the onboard battery
2. Test (A) was performed with the power-off test of the log above. After the system was powered off, the cache was removed from its current adapter and re-inserted into the system before rebooting.
3. Data on the cache was recovered successfully by meeting the requirements listed in the power-off test of the logs above.
4. Test (B). The current count of the total number of orders was determined by the sum of D\_NEXT\_O\_ID of all rows in the DISTRICT table, giving SUM\_1.
5. Test (B) was conducted by inducing a battery failure during a test run. Once the system had reached steady state, the battery was failed using a toggle switch. The system recorded the failure, flushed out its vram contents and quit using the cache. Error notices were posted into the system error log.
6. The run continued without the cache.
7. Step 4 was performed returning SUM\_2. It was verified that SUM\_2 was equal to SUM\_1 plus the completed New\_Order transactions recorded by the RTE and that no entries existed for rolled-back transactions.
8. Consistency condition 3 was verified.

---

## 5. Clause 4: Scaling and Data Base Population Related Items

---

### 5.1 Cardinality of Tables

The cardinality (e.g., the number of rows) of each table, as it existed at the start of the benchmark run, must be disclosed.

Table 5-1 portray the TPC Benchmark™ C defined tables and the number of rows for each table as they were built initially.

---

**Table 5-1 Initial Cardinality of Tables (RS/6000 S80)**

Table Name	Number of Rows
Warehouse	10,700
District	107,000
Customer	321,000,000
History	321,000,000
Order	321,000,000
New Order	96,300,000
Order Line	3,210,960,710
Stock	1,070,000,000
Item	100,000

---

### 5.2 Distribution of Tables and Logs

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

The following table depicts the data base configuration of the system tested.

---

**Table 5-2. RS/6000 S80 Data Distribution Benchmark Configuration**

Cntrl	Disk	Contents	Capacity
ssa	hdisk4	lvsys1	9GB
ssa	hdisk5	lvtpcc cntl1-3	9GB
ssa	hdisk6-7	lvware	9GB
ssa	hdisk8-11	lvhist1-10	9GB
ssa	hdisk12-19	lvroll1-2	9GB
ssa	hdisk22-51	lvcust1-10	9GB
ssa	hdisk52-81	lvstock1-9	9GB
ssa	hdisk82-86	lvi1cust1-5	9GB
ssa	hdisk88-99	lvi2cust1-15	9GB
ssa	hdisk100-129	lvcust11-20	9GB
ssa	hdisk130-159	lvstock10-18	9GB
ssa	hdisk160-189	lvstock136-144	9GB
ssa	hdisk190	lvitem	9GB

<b>Cntrl</b>	<b>Disk</b>	<b>Contents</b>	<b>Capacity</b>
ssa	hdisk191	paging00	9GB
ssa	hdisk192	paging01	9GB
ssa	hdisk193	paging02	9GB
ssa	hdisk194	paging03	9GB
ssa	hdisk195	paging04	9GB
ssa	hdisk196-225	lvcust21-30	9GB
ssa	hdisk226-255	lvstock19-27	9GB
ssa	hdisk256-285	lvstock145-152	9GB
ssa	hdisk286-290	lvnord1-4	9GB
ssa	hdisk291	lvord1-14	9GB
ssa	hdisk292-321	lvcust31-40	9GB
ssa	hdisk322-351	lvstock28-36	9GB
ssa	hdisk352-381	lvstock153-160	9GB
ssa	hdisk382-386	lvnord1-4	9GB
ssa	hdisk387	lvord1-14	9GB
ssa	hdisk468-497	lvcust41-49,141	9GB
ssa	hdisk498-527	lvstock37-45	9GB
ssa	hdisk528-557	lvstock161-168	9GB
ssa	hdisk558-563	lvi2cust1-15	9GB
ssa	hdisk564-593	lvcust50-58,142	9GB
ssa	hdisk594-623	lvstock46-54	9GB
ssa	hdisk624-653	lvstock169-176	9GB
ssa	hdisk654-658	lvord1121-140	9GB
ssa	hdisk659	paging05	9GB
ssa	hdisk660-689	lvcust59-67,143	9GB
ssa	hdisk690-719	lvstock55-63	9GB
ssa	hdisk720-749	lvstock177-184	9GB
ssa	hdisk750-755	lvord1-14	9GB
ssa	hdisk756-785	lvcust68-76,144	9GB
ssa	hdisk786-815	lvstock64-72	9GB
ssa	hdisk816-830	lvord11-20	9GB
ssa	hdisk831-835	lvord1121-140	9GB
ssa	hdisk836-844	lvi1ord1-4	9GB
ssa	hdisk845-849	lvstk1-15	9GB
ssa	hdisk850	paging06	9GB
ssa	hdisk851	paging07	9GB
ssa	hdisk852-881	lvcust77-85	9GB
ssa	hdisk882-911	lvstock73-81	9GB
ssa	hdisk912-941	lvstock185-192	9GB
ssa	hdisk942-946	lvord1121-140	9GB
ssa	hdisk948-977	lvcust86-95	9GB
ssa	hdisk978-1007	lvstock82-90	9GB
ssa	hdisk1008-1037	lvstock193-200	9GB
ssa	hdisk1038-1042	lvstk1-15	9GB
ssa	hdisk1044-1073	lvcust96-105	9GB
ssa	hdisk1074-1103	lvstock91-99	9GB
ssa	hdisk1104-1118	lvord121-40	9GB
ssa	hdisk1119-1127	lvi2ord4-6	9GB
ssa	hdisk1140-1169	lvcust106-115	9GB
ssa	hdisk1170-1199	lvstock100-108	9GB
ssa	hdisk1200-1214	lvord141-60	9GB



<b>Cntrl</b>	<b>Disk</b>	<b>Contents</b>	<b>Capacity</b>
ssa	hdisk1215-1223	lvi2ord7-9	9GB
ssa	hdisk1236-1265	lvcust116-125	9GB
ssa	hdisk1266-1295	lvstock109-117	9GB
ssa	hdisk1296-1310	lvordl61-80	9GB
ssa	hdisk1311-1319	lvi2ord10-12	9GB
ssa	hdisk1332-1361	lvcust126-135	9GB
ssa	hdisk1362-1391	lvstock118-126	9GB
ssa	hdisk1392-1406	lvordl81-100	9GB
ssa	hdisk1407-1415	lvi1ord5-8	9GB
ssa	hdisk1428-1437	lvcust136-140	9GB
ssa	hdisk1438-1467	lvstock127-135	9GB
ssa	hdisk1468-1482	lvordl101-120	9GB
ssa	hdisk1483-1491	lvi2ord1-3	9GB
ssa	hdisk388	lvlog	72.9GB
ssa	hdisk389	lvlog	72.9GB
ssa	hdisk390	lvlog	72.9GB
ssa	hdisk391	lvlog	72.9GB
ssa	hdisk392	lvlog	72.9GB
ssa	hdisk393	lvlog	72.9GB
ssa	hdisk394	lvlog	72.9GB
ssa	hdisk395	lvlog	72.9GB
ssa	hdisk396	lvlog	72.9GB
ssa	hdisk397	lvlog	72.9GB
ssa	hdisk398	lvlog	72.9GB
ssa	hdisk401	lvlog	72.9GB

---

### 5.3 Data Base Model Implemented

*A statement must be provided that describes the data base model implemented by the DBMS used.*

The database manager used for this testing was Oracle8 Enterprise Edition 8.1.6 from Oracle Inc. Oracle8 Enterprise Edition 8.1 is a relational DBMS.

---

### 5.4 Partitions/Replications Mapping

*The mapping of data base partitions/replications must be explicitly described.*

IBM did not implement horizontal or vertical partitioning for this TPC-C test.

## 5.5 180 day space calculations

### RS/6000 Enterprise Server S80

TPM		126,671.10						
Warehouses		10,700						
<b>SEGMENT</b>	<b>TYPE</b>	<b>TSPACE</b>	<b>'BLOCKS</b>	<b>FIVE_PCT</b>	<b>DAILY_GROW</b>	<b>TOTAL</b>		
CUSTOMER	TABLE	CUST	80250003.00	4012500	0	84262503.15		
DISTRICT	TABLE	WARE	53511.00	2676	0	56186.55		
HISTORY	TABLE	HIST	4587806.00	0	868998	5456804.03		
ICUSTOMER	INDEX	ICUST1	2304000.00	115200	0	2419200.00		
ICUSTOMER2	INDEX	ICUST2	3840000.00	192000	0	4032000.00		
IDISTRICT	INDEX	WARE	2560.00	128	0	2688.00		
ITEM	INDEX	ITEMS	5120.00	256	0	5376.00		
INORD	INDEX	NORD	402688.00	20134	0	422822.40		
ORDERS	INDEX	ORD1	2073600.00	103680	0	2177280.00		
ORDERS2	INDEX	ORD2	3072000.00	153600	0	3225600.00		
ORDL	INDEX	ORDL	58572800.00	0	11094551	69667351.04		
ISTOCK	INDEX	ISTR	5382144.00	269107	0	5651251.20		
ITEM	TABLE	ITEMS	3031.00	152	0	3182.55		
WAREHOUSE	INDEX	WARE	256.00	13	0	268.80		
ORDERS	TABLE	ORD	3180841.00	0	602498	3783339.14		
ROLL_SEG	SYS	ROLL	409600.00	0	0	409600.00		
STOCK	TABLE	STOCKS	97272729.00	4863636	0	102136365.45		
SYSTEM	SYS	SYSTEM	691200.00	0	0	691200.00		
WAREHOUSE	TABLE	WARE	5355.00	268	0	5622.75		
Total			262,109,244	9,733,350	12,566,047	284,408,641		
Dynamic space		66,341,447						
Static space		205,501,147						
Free space		12,566,047						
Daily growth		12,566,047						
Daily spread		0	Oracle may be configured such that daily spread is 0					
180-day space (blk.)		2,467,389,643						
Block size (bytes)		4,096						
180-day (GB)		9,412.34						
			new_order	9986434.00				
Log block size		512	Redo blocks written	276325322.00				
Log blocks/tpm C		27.67	Number of log blocks used in one tpm C					
8-hour log (GB)		802.23						
Disk	Disk	SUT	SUT	Priced	Priced	Space usage (GB)		
Type	Formatted	Capacity	# of disks	Capacity(GB)	# of disks	Capacity(GB)	180-day	9412.34
							RAID	874.80
							os+ paging	4.31
9.1	8672	1358	11500.56	1358	11500.56		Paging	66.88
RAID(5-18.1GB)	74649.60	12	874.80	12	874.80		Total Space	10358.33

## 6. Clause 5: Performance Metrics and Response Time Related Items

### 6.1 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 6-1 list the response times and the ninetieth percentiles for each of the transaction types for the measured system.

### 6.2 Keying and Think Times

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

Table 6-1 list the TPC-C keying and think times for the measured system.

**Table 6-1. RS/6000 Enterprise Server S80 Response, Think and Keying Times**

<b>Response Times</b>	<b>New Order</b>	<b>Payment</b>	<b>Order Status</b>	<b>Delivery (int./def.)</b>	<b>Stock Level</b>	<b>Menus</b>
<b>90 %</b>	0.54	0.48	0.55	0.19/0.xx	0.85	0.01
<b>Average</b>	0.26	0.22	0.26	0.11/0.xx	0.52	0.01
<b>Maximum</b>	7.55	7.29	5.11	1.61/3.xx	7.55	3.59
			<b>Think Times</b>			
<b>Minimum</b>	0.01	0.01	0.01	0.01	0.01	N/A
<b>Average</b>	12.01	12	10.01	5.02	5	N/A
<b>Maximum</b>	120.11	120.11	100.1	50.1	50.1	N/A
			<b>Keying Times</b>			
<b>Minimum</b>	18	3	2	2	2	N/A
<b>Average</b>	18.01	3.01	2.01	2.01	2.01	N/A
<b>Maximum</b>	18.18	3.19	2.16	2.17	2.17	N/A

### 6.3 Response Time Frequency Distribution

Response time frequency distribution curves must be reported for each transaction type.

Figure 6-3-1. RS/6000 S80 New-Order Response Time Distribution

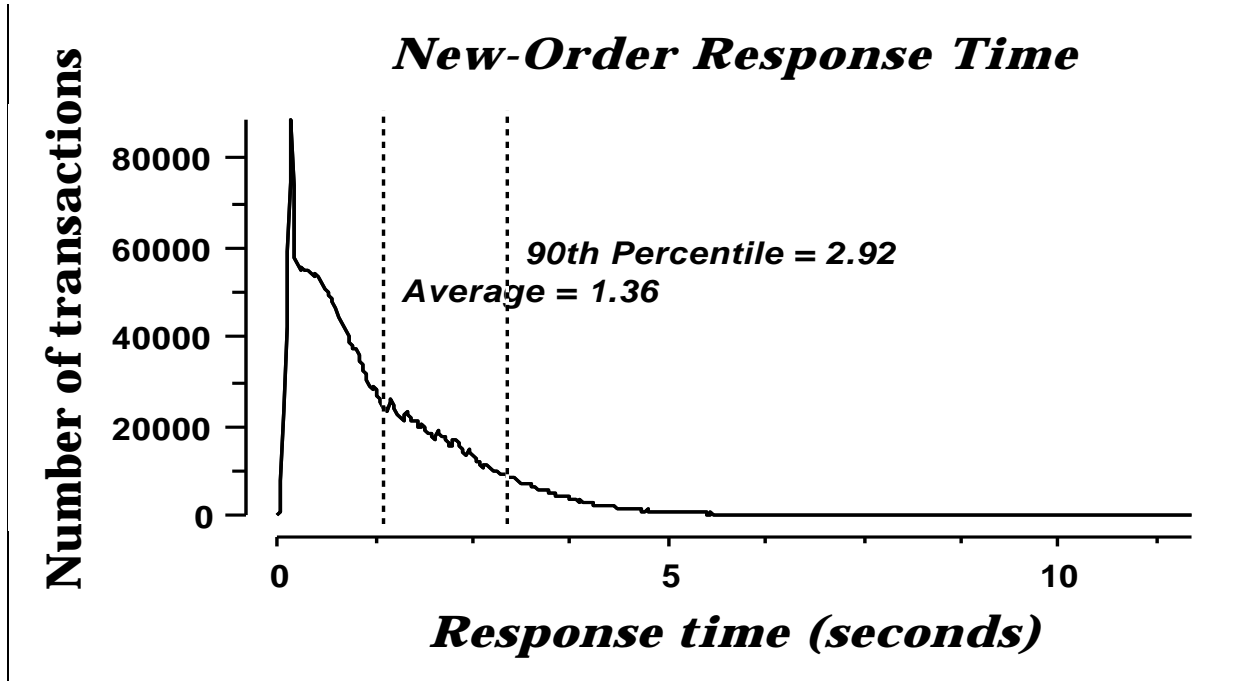


Figure 6-3-2. RS/6000 S80 Payment Response Time Distribution

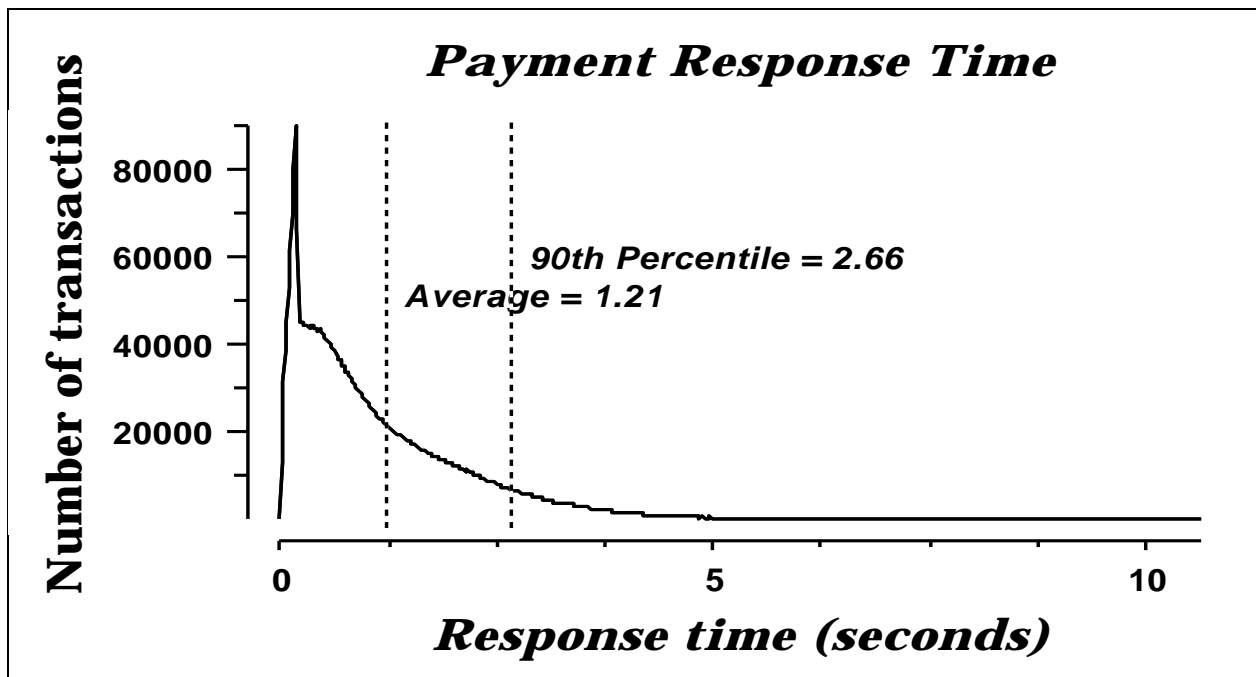


Figure 6-3-3. RS/6000 S80 Order-Status Response Time Distribution

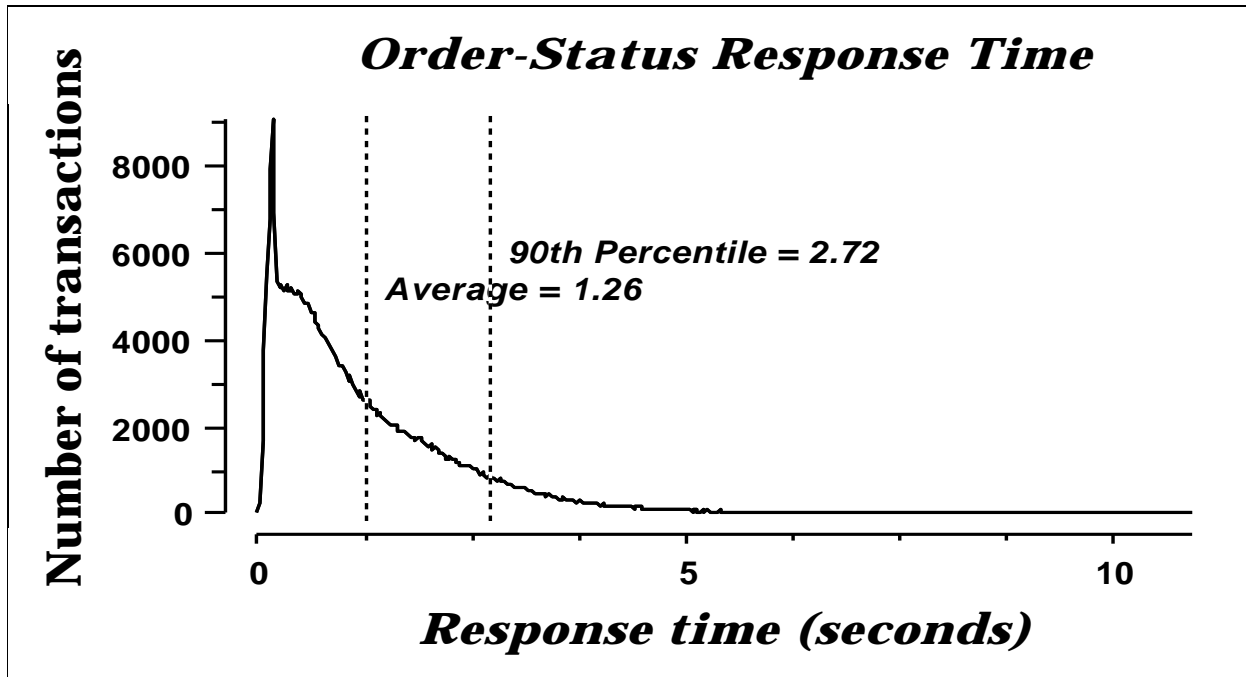


Figure 6-3-4. RS/6000 S80 Delivery (Interactive) Response Time Distribution

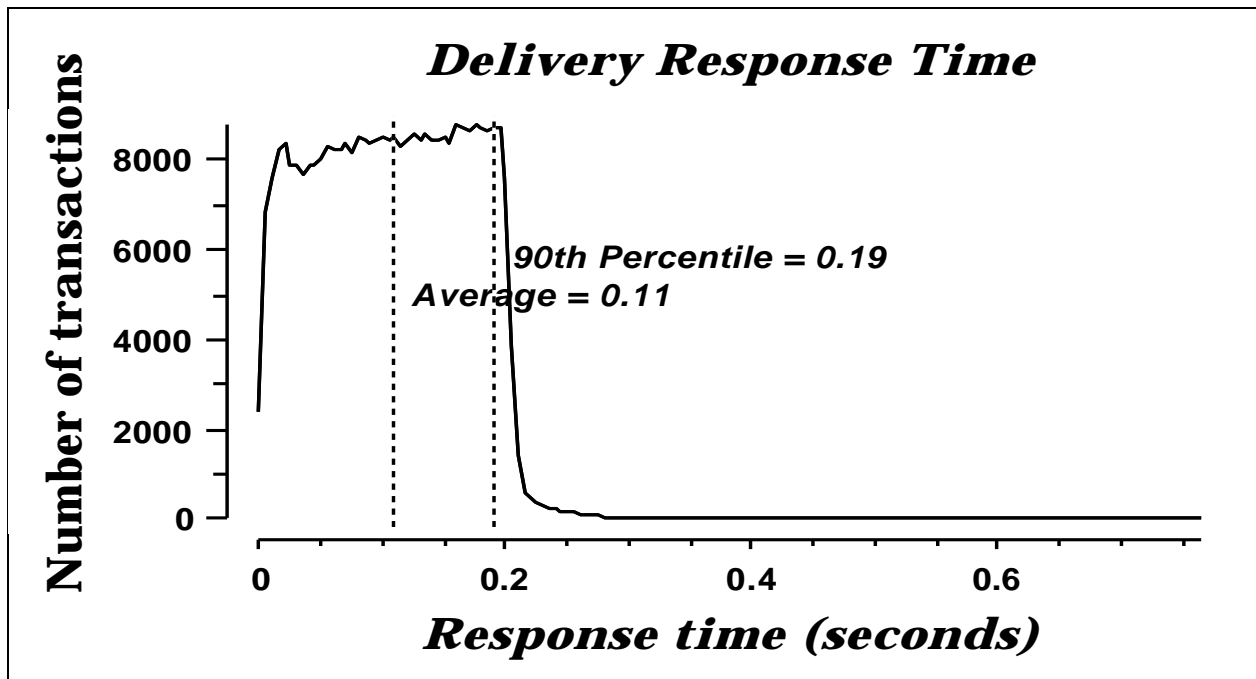


Figure 6-3-5. RS/6000 S80 Delivery (Deferred) Response Time Distribution

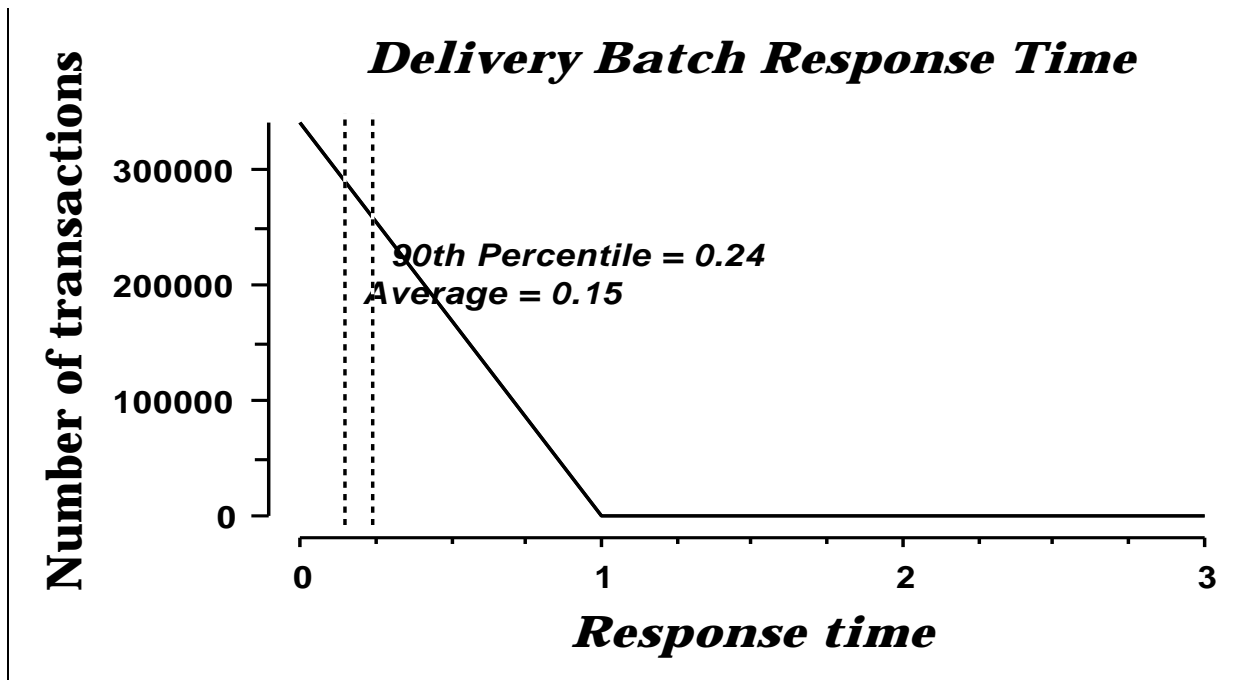
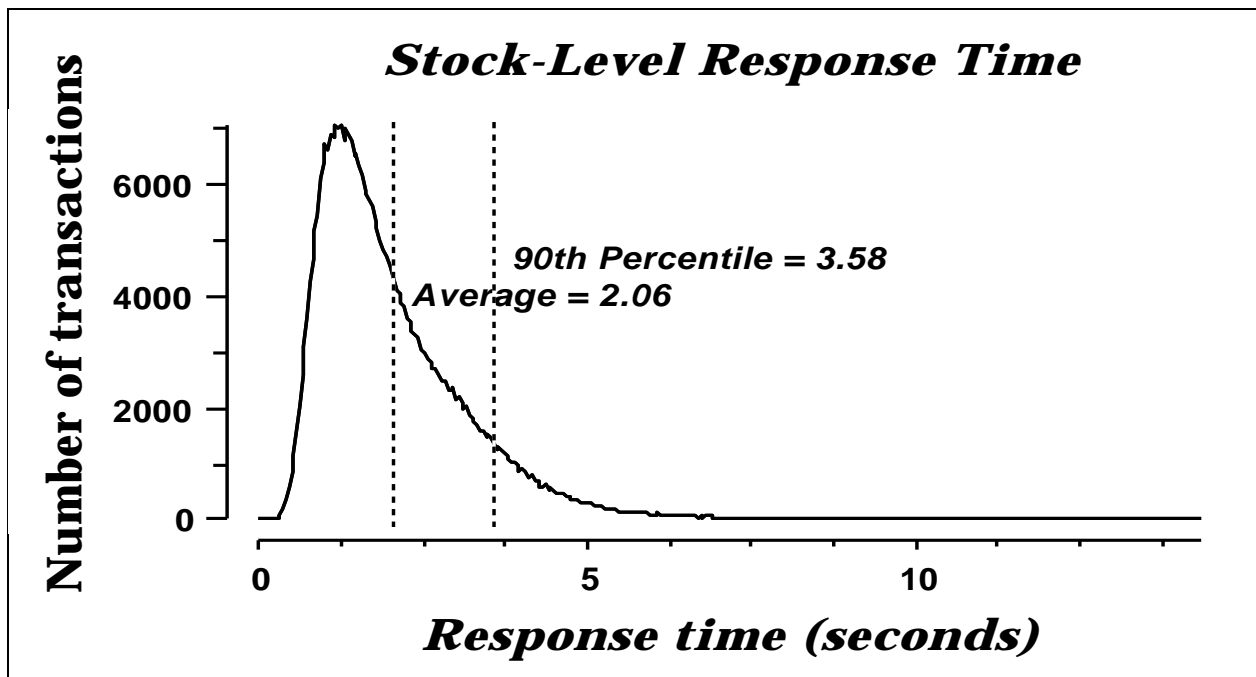


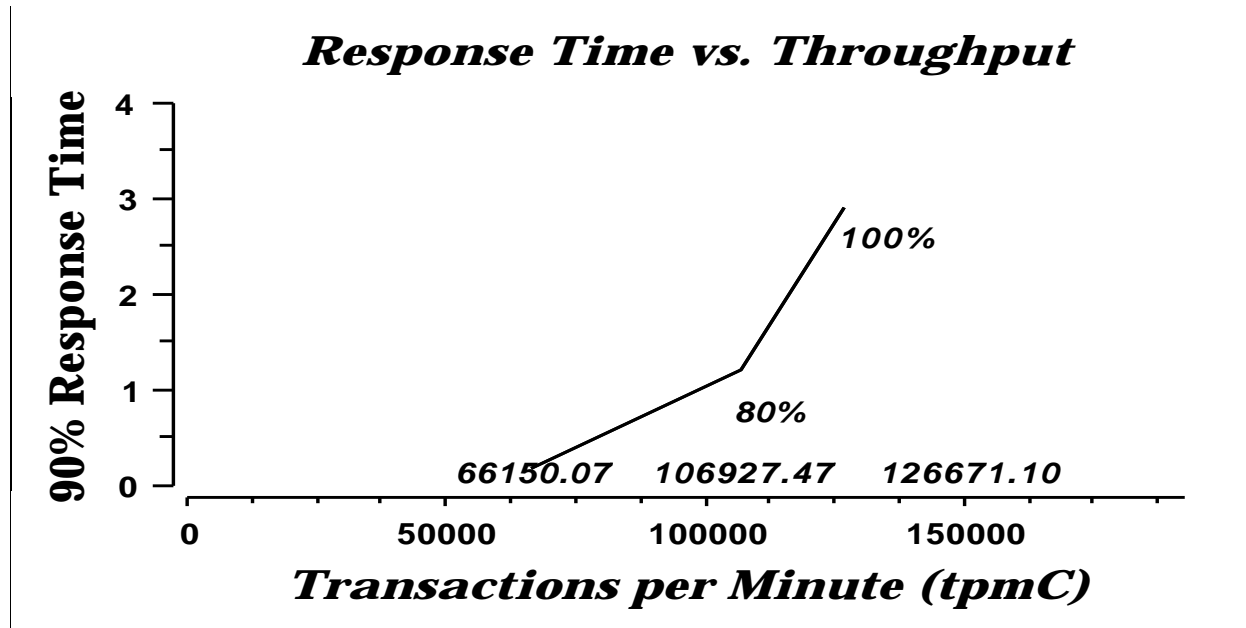
Figure 6-3-6. RS/6000 S80 Stock Level Response Time Distribution



## 6.4 Performance Curve for Response Time versus Throughput

The performance curve for response times versus throughput must be reported for the New-Order transaction.

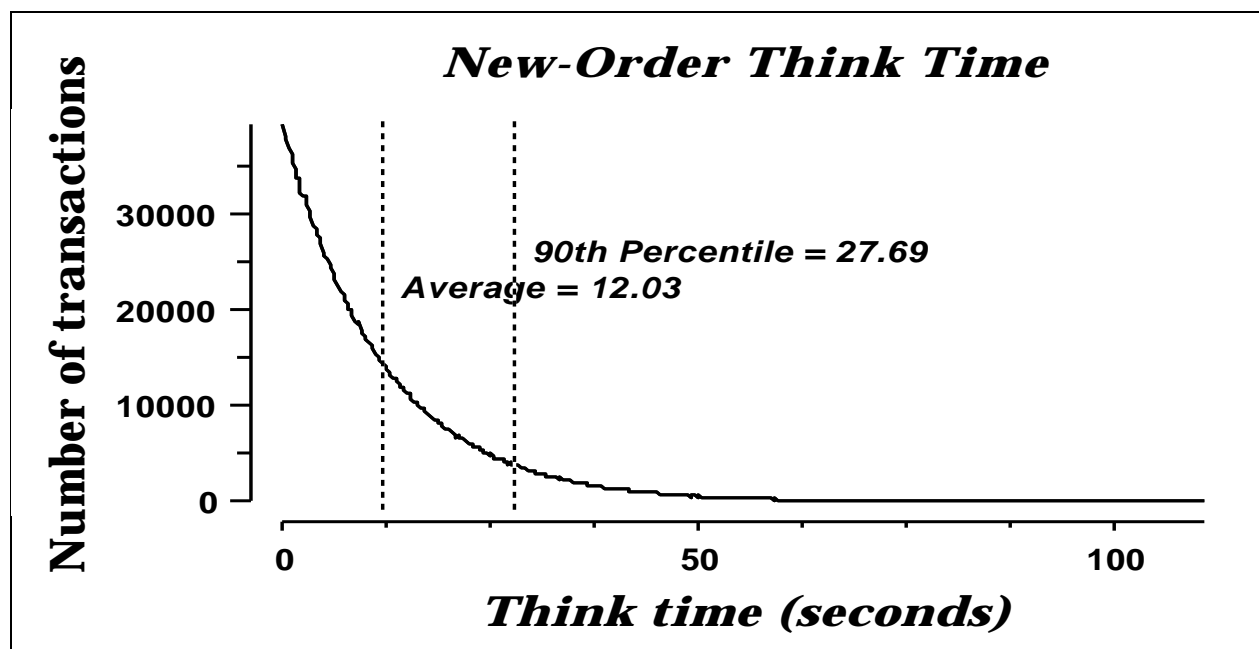
Figure 6-4-1. RS/6000 S80 New-Order Response Time vs. Throughput



## 6.5 Think Time Frequency Distribution

A graph of the think time frequency distribution must be reported for the New-Order transaction.

Figure 6-5-1. RS/6000 S80 New-Order Think Time Distribution



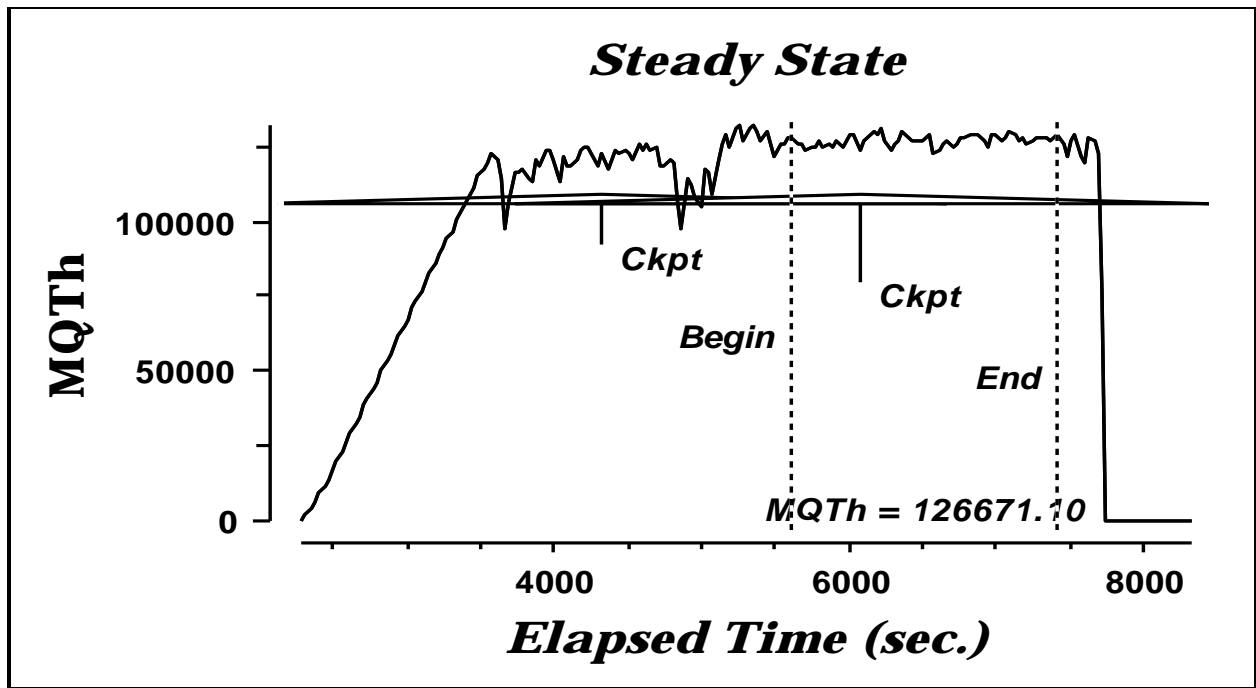
---

## 6.6 Throughput versus Elapsed Time

A graph of throughput versus elapsed time must be reported for the New-Order transaction.

---

**Figure 6-6-1. New-Order Throughput vs. Elapsed Time**



### 6.7 Steady State Determination

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

All the emulated users were allowed to logon and do transactions. The time stamping interval was set to start after several minutes of rampup. Refer to the Numerical Quantities Summary pages for the rampup time. Figure 6.7.1 New-Order throughput versus Elapsed Time graph show that the system was in steady state at the beginning of the Measurement Interval.

---

## 6.8 Work Performed During Steady State

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

---

### 6.8.1 Transaction Flow

For each of the TPC Benchmark™C transaction types, the following steps are executed:

IBM TXSeries version 4.2, Encina interface, was used as a transaction manager (TM). Each transaction was divided into three programs. The front end program handled all screen I/O, a database client program which connected to the database and served as a TXSeries server (a back end program), and a database server program which handled all database operations at the SUT. Both the front end and back end programs ran on the client system. The front end program communicates with the database client program through DCE RPCs. The database client program communicates with the Server system over Ethernet using SQL\*Net calls. Besides calling TXSeries Encina initialization code during startup, all other functions are transparent to the application code. Encina routes the TPC Benchmark™ C Full Disclosure Report - IBM RS/6000



transaction and balances the load according to the options defined in the configuration file in appendix B.2, The transaction flow is described below.

- Each client machine is a node in an Encina Cell.
- Two servers are configured in each node: one processes the delivery transactions and one all other transaction.
- The delivery server is configured with one processing agent with 2 server manager DCE threads, and 8 background threads to process deferred deliveries. Each background thread has one connection to the database.
- The other server is configured with 44 processing agents. Each processing agent has 1 server manager DCE threads. Each thread has one connection to the database.
- When the Encina clients are started, they connect to Encina cell.
- When terminals are started, each terminal connects to the Encina client. The client spawns a thread for each connection to handle that connection. The thread executes the 'process\_terminal' routine. The process\_terminal displays the TPC-C transaction menu on the user terminal.
- The TPC-C user chooses the transaction type and proceeds to fill the screen fields required for transaction.
- The process\_terminal accepts all values entered by the user and transmits those values to one of the TPC\_C backend programs. The transaction is performed through a DCE RPC. There is an interface for each TPC-C transaction type and each TPC-C backend program exports one or more of these interfaces. (The delivery servers export only the delivery interface, the other servers export the other four interfaces, and only those). Encina transparently routes the RPC to one of the servers exporting the corresponding interface.
- A TPC-C backend server program receives an RPC and proceeds to execute all database operations related to the request. All information entered on the user terminal is contained in the RPC.
- Once the transaction is committed, the server program fills in the output parameters. The RPC is then sent back to the client program.
- When the RPC returns to the client, the process\_terminal routine writes the transaction out on the user terminal.

---

## 6.8.2 Database Transaction

All database operations are performed by the TPC-C back-end programs. The process is described below:

Using SQL\*Net calls, the TPC-C back-end program interacts with Oracle8 Server to perform SQL data manipulations such as update, select, delete and insert, as required by the transaction. After all database operations are performed for a transaction, the transaction is committed.

Oracle8 Server proceeds to update the database as follows:

When Oracle8 Server changes a database table with an update, insert, or delete operation, the change is initially made in memory, not on disk. When there is not enough space in the memory buffer to read in or write additional data pages, Oracle8 Server will make space by flushing some modified pages to disk. Modified pages are also written to disk when a checkpoint occurs. Before a change is made to the database, it is first recorded in the transaction log. This ensures that the database can be recovered completely in the event of a failure. Using the transaction log, transactions that started but did not complete prior to a failure

can be undone, and transactions recorded as complete in the transaction log but not yet written to disk can be redone.

---

### **6.8.3 Checkpoints**

A checkpoint is the process of writing all modified data pages to disk. The TPC-C benchmark was setup to automatically checkpoint every 30 minutes. One checkpoint occurs during the rampup period, with another occurring during the measurement interval.

---

### **6.9 Reproducibility**

*A description of the method used to determine the reproducibility of the measurement results must be reported.*

A repeatability measurement was taken for the same length of time as the measured run. The repeatability measurement was 125,412.93 tpmC.

---

### **6.10 Measurement Interval**

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

A thirty minute Measurement Interval was used. Further, the measurement interval is a multiple of the checkpoint interval, and the checkpoints fall outside the protected zones of either edge of the measurement interval (as required by Clause 5.5.2.2). This demonstrates that a different measurement interval over the eight hour period would yield similar throughput results.

---

## 7. Clause 6: SUT, Driver, and Communication Definition Related Items

---

### 7.1 RTE Availability

*If the RTE is commercially available, then its inputs must be specified. Otherwise, a description must be supplied of what inputs to the RTE had been used.*

IBM used an internally developed RTE for these tests. Appendix D contains the scripts used in the testing.

---

### 7.2 Functionality and Performance of 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.*

In the benchmark configuration the Remote Terminal Emulator (RTE) communicates with the client system over Ethernet. One RS/6000 Model 43P-260 emulates a network of 7,000 RS/6000 Model 43P-140 workstations. The communications mechanism used in the benchmark and priced configurations are the same. In the benchmark configuration a separate Ethernet LAN was used to connect two driver systems to a client system. In other words, there was a separate LAN segment every two drivers to a client. Each LAN segment in the priced configuration is used to connect 1,000 workstations.

---

### 7.3 Network Bandwidth

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

The Ethernet used in the LAN complies with the IEEE 802.3 standard and has a bandwidth of 10 Megabits per second. Each LAN segment in the RS/6000 Enterprise Server S80 configuration connected 1,000 workstations.

---

### 7.4 Operator Intervention

*If the configuration requires operator intervention, the mechanism and the frequency of this intervention must be disclosed.*

No operator intervention is required to sustain the reported throughput during the eight hour period.

---

## 8. Clause 7: Pricing Related Items

---

### 8.1 Hardware and Programs Used

*A detailed list of the hardware and software used in the priced system must be reported. Each 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, contents of the package must be disclosed. Pricing source(s) and effective date(s) must also be reported.*

The detailed list of all hardware and programs for the priced configuration is listed in the pricing sheets (please refer to Section 8.2 for details) for each system reported. The prices for all products and features that are provided by IBM are available the same day as product or feature availability.

Pricing for IBM Websphere Application Server Enterprise Edition is for Txseries License only.

Pricing for 8 port Ethernet Hubs was quoted by Netercruiser Tech. Inc..

Pricing for the 8 port Ethernet Switches was quoted by PC Connection. These are warranted (Labor and Product) for 5 years from purchase.

---

### 8.2 Five Year Cost of System Configuration

*The total 5-year price of the entire configuration must be reported, including: hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.*

The price sheets for the RS/6000 are contained on the following page. The basis for the discounts used are:

#### **5-year Term Maintenance Contract Discount**

This discount is available for customers who sign a 5-year maintenance agreement on the hardware. A discount of 8% is available for customers when they sign a 5-year maintenance agreement

#### **Scope incentive**

A 2% discount is applied for a ServiceElect contract that combines hardware maintenance with one or more services, which in this pricing report the selected service is Support Line.

#### **5-year Maintenance Prepay Discount**

This is a discount for prepayment of maintenance costs. A discount of 15.05% is available for this configuration based on payment for five years maintenance at time of purchase. This discount is applied to the balance after the 5-year term maintenance contract discount and Scope discount is applied.

For this configuration the discount is about 35% for the hardware and software totals.

---

### 8.3 Availability Dates

*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.*

All products are generally available today except the following:

<b>Product</b>	<b>Availability Date</b>
Oracle 8.1.6	March 1, 2000

---

## 8.4 Statement of tpmC and Price/Performance

*A statement of the measured tpmC, as well as the respective calculations for 5-year pricing, price/performance (price/tpmC), and the availability date must be disclosed.*

<b>System</b>	<b>tpmC</b>	<b>5-year System Cost</b>	<b>\$/tpmC</b>	<b>Availability Date</b>
RS/6000 Enterprise Server S80	135,815.7	\$7,067,644	\$52.04	All HS/SW available as shown in Section 8.3

## RS/6000 Enterprise Server S80 Five Year System Price Configuration

Description	Part No.	Source	Unit Price	Qty	Ext Price	Maint Price
<b>Server Hardware</b>						
RS/6000 Server Model S80	7017-S80		11,538	1	11,538	45,168
16bit PCI SCSI cables	2447		75	1	75	0
32x CD-ROM drive	2624		375	1	375	0
9.1 GB 1" Ultra SCSI Hot Swap	2913		1,260	1	1,260	0
Async/Terminal Cable	2934		45	1	45	0
10/100 Mbs Ethernet PCI Adapter	2968		275	16	4,400	0
Remote I/O Cable (3142, 3143) set	3142		4,170	1	4,170	0
16384 MB (4x4096) R1 Memory	4194		147,456	4	589,824	0
6way RS64 III 450 Mhz, 8MB L2	5318		200,000	4	800,000	297,408
I/O Power Cable set (6006,7,8)	6000		685	1	685	0
1000 Watt AC Power Supply	6193		765	1	765	0
Power Regulators (6194, 6195)	6194		2,505	1	2,505	0
Ultra SCSI PCI-Bus Adapter	6206		395	2	790	0
Advanced Serial RAID Adapter	6225		3,000	18	54,000	0
32 MB Fast-Write Cache Option	6235		575	3	1,725	0
SCSI I/O Drawer, 10 EIA	6320		23,766	4	95,064	0
Primary I/O Drawer Group	6321		600	1	600	0
Secondary I/O Drawer Group	6323		734	3	2,202	0
Support Processor Group	6326		2,700	1	2,700	0
Remote I/O Hub, Dual Loop	6503		8,000	1	8,000	0
SCSI 6-pack Hot Swap Back Plane	6547		600	1	600	0
I/O Rack	7000		3,500	2	7,000	0
Prestige 6000VA, 208-240V UPS	9910-EP8		5,505	1	5,505	0
Prestige EXT UPS, Rack Mount	9910-U33		3,822	2	7,644	0
System Rack Model S00	7014-S00		3,500	1	3,500	1,536
System Rack Model T00	7014-T00		3,550	10	35,500	15,360
Additional Power Distribution Unit	6171		1,000	21	21,000	0
SSA Disk Subsystem, Black Cover	7133-D40		13,000	89	1,157,000	786,048
50/60 Hz AC, 300 VDC Power Supply	8022		2,000	89	178,000	0
9.1 GB Disk Drive Modules	8209		2,760	1,357	3,745,320	0
18 GB Disk Drive Modules	8209		4,500	60	270,000	0
SSA Cables	8802		65	178	11,570	0
			<b>Subtotal</b>		<b>7,023,362</b>	<b>1,145,520</b>
<b>Server Software</b>						
AIX 4.3.3 for S80	5765-C34		50	1	50	40,920
IBM C for AIX V4.4	5765-C64		799	1	799	30,480
Oracle8 Enterprise Edition 8.1.6 64bit		1	695,970	1	695,970	695,970
			<b>Subtotal</b>		<b>696,819</b>	<b>767,370</b>
<b>Client Hardware</b>						
RS/6000 Model 43P-260	7043-260		3,762	15	56,430	101,520
32x CD-ROM	2624		375	15	5,625	0
Async Terminal/Printer Cable	2934		45	15	675	0
10/100 Mbps Ethernet Adapter, PCI	2968		275	30	8,250	0
9.1 GB SCSI Hard Disk	3029		1,260	30	37,800	0
Memory Expansion Feature	4098		1,038	15	15,570	0
512MB SDRAM DIMM Memory	4119		5,760	60	345,600	0
200 MHz Power3 Processor, 4MB L2	4342		8,000	30	240,000	0
IBM ASCII Terminal, Keyboard, Cable	3153-BG3		577	16	9,232	9,728
			<b>Subtotal</b>		<b>719,182</b>	<b>111,248</b>
<b>Client Software</b>						
AIX 4. Unlimited Users	5756-C34		50	15	750	24,900
IBM Websphere Application Server	41L3462		10,500	15	157,500	25,513
			<b>Subtotal</b>		<b>158,250</b>	<b>50,413</b>
<b>User Connectivity</b>						
8port Ethernet Switch + 2 Spare	ES-3108P	2	79	20	1,580	6,240
Ethernet Hub (8port) + 10% Spares	CT1008D/P	3	18.00	14,784	266,112	0
			<b>Subtotal</b>		<b>267,692</b>	<b>6,240</b>
			<b>Discounts</b>		<b>(3,555,739)</b>	<b>(322,713)</b>
			<b>Total</b>		<b>5,309,566</b>	<b>1,758,078</b>
Notes:						
Pricing Sources:			<b>Five-Year Cost of Ownership:</b>		<b>7,067,644</b>	
1=Oracle, 2=PC Connection, 3=NetCruiser Technologies, Inc					<b>tpmC</b>	<b>135,815.70</b>
Audited by: Francois Raab, Info Sizing					<b>\$/tpmC</b>	<b>52.04</b>

---

## **9. Clause 9: Audit Related Items**

*If the benchmark has been independently audited, then the auditor's name, address, phone number, and a brief audit summary report indicating compliance must be included in the Full Disclosure Report. A statement should be included, specifying when the complete audit report will become available and who to contact in order to obtain a copy.*

The auditor's attestation letter is included in this section of this report.

# Appendix A: TPC-C Application

## Source

### A.1 Client/Terminal Handler code

#### callora.c

```
/*
 * callora.c
 *
 * $Revision: 1.3 $
 * $Date: 1999/05/06 21:28:29 $
 * $Log: callora.c,v $
 *
 * $TALog: callora.c,v $
 * Revision 1.3 1999/05/06 21:28:29 oz
 * - Removed all the .. from the includes
 * - Added -I. to the makefiles instead
 * - Moved all the thread related code and connection
 *   selection to serverMon.c
 *
 * - get_db_ready now does not take the number of connections
 * - Export create_connection() and clean_connection(void *)
 * - All the transactions take a connection pointer as a first param
 * [from r1.2 by delta oz-24309-TPCC-add-oracle8.1-code, r1.5]
 *
 * Revision 1.2 1999/04/19 20:14:48 oz
 * - Moved all the simulated code to server.c
 * - Created nulldb.c for compilation with no DB
 * [from r1.1 by delta oz-24331-TPCC-move-sim-code-to-common-file, r1.1]
 *
 * Revision 1.1 1999/04/19 14:37:27 oz
 * [added by delta wenjian-23742-TPCC-update-with-Ralieggh-code, r1.3]
 *
 * Revision 1.15 1998/10/22 20:51:00 wenjian
 * [merge of changes from 1.6 to 1.14 into 1.12]
 *
 * Revision 1.14 1998/10/08 14:17:57 dongfeng
 * Add codes for doing web-based tpcc.
 * [from r1.6 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.1]
 *
 * Revision 1.12 1998/09/04 19:17:54 wenjian
 * Add new variables: more_srv_work, period_to_add_rt, and
 * period_to_check_tran to replace the original constants in
 * order to control the increment of server RT.
 * [from r1.11 by delta wenjian-23183-TPCC-sync-AIX-code-with-Austin, r1.5]
 *
 * Revision 1.11 1998/08/28 18:29:56 wenjian
 * This delta sync the TPCC code with Austin.
 *
 * Modify get_wait_time():
 * - add rt_increment so that the wait time is increased in a certain time;
 * - rt_increment is reset to 0 at the beginning of each run
 * - the waiting time is different for different tran type.
 * [from r1.8 by delta wenjian-23183-TPCC-sync-AIX-code-with-Austin, r1.1]
 *
 * Revision 1.8 1998/08/18 14:38:37 wenjian
 * Change the wait time for NewOrder to 0.23 second
 * [from r1.6 by delta wenjian-21750-TPCC-changes-for-porting-on-NT, r1.4]
 *
 * Revision 1.6 1998/06/17 15:28:50 wenjian
 * - Reduce matrix size
 * - In get_wait_time(), the waiting time is decided by transaction type.
 * [from r1.5 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients, r1.2]
 *
 * Revision 1.5 1998/02/17 22:06:58 wenjian
 * Define macro RANDOM as rand on NT and random on other platforms
 * [from r1.4 by delta wenjian-21750-TPCC-changes-for-porting-on-NT, r1.1]
 *
 * Revision 1.4 1998/01/23 15:07:42 oz
 * - Updated the SP TPCC directory to the latest files used
 * - during the SP tpcc audit.
 * [from r1.3 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
 *
 * Revision 1.1 1997/07/22 21:17:14 radha
 * [added by delta radha-20360-TPCC-integrate-with-Oracle-7322-drivers, r1.1]
 *
 */

#if 1
#define NULL_WITH_SLEEP
#endif

#include <stdio.h>
#include <time.h>
#include <string.h>
#include "serverDebug.h"
```

```
#if defined(solaris)
#include <dce/pthread.h>
#else /* solaris */
#include <pthread.h>
#endif /* solaris */

#ifdef MULTIPLE_INTERFACE
#include "common/neworder.h"
#include "common/payment.h"
#include "common/stocklevel.h"
#include "common/orderstatus.h"
#else
#include "common/tpcc_trans.h"
#endif
#include "common/databuf.h"
#include "server.h"

#ifdef WIN32
#include <winsock.h>
#endif

#include "tpcc_info.h"

#ifdef WIN32
#define RANDOM rand
#else
#define RANDOM random
#endif

extern int server_null_test;
extern void *create_ora_connection();

#ifdef DEBUG_SERVER
#define PRINT_NEW_IN(a, b) fprintf(stderr, "%s\n", b); print_new_in(a)
#define PRINT_NEW_ORDER(a, b) fprintf(stderr, "%s\n", b); print_new_order(a)
#define PRINT_NEW_RES(rc, a) \
    fprintf(stderr, "<R do_new_order, rc=%d, transtatus=%d, duplicates=%d, all_local=%d\n", \
        rc, (a)->s_transtatus, (a)->s_all_local, (a)->duplicate_items)
#else
#define PRINT_NEW_RES(rc, a)
#define PRINT_NEW_ORDER(a, b)
#define PRINT_NEW_IN(a, b)
#define PRINT_DIST_NEW_ORDER(a, b)
#endif

#define TPCC_RET_SCP(a,b,len) \
    strncpy((char *)dataP->b, (char *)oraStruct.a, len); \
    (char *)dataP->b[(len)-1] = '\0'
#define TPCC_CP(a,b) oraStruct.a = dataP->b
#define TPCC_SCP(a,b,len) strncpy((char *)oraStruct.a, (char *)dataP->b, len)
#define TPCC_RET_CP(a,b) dataP->b = oraStruct.a

#define TPCCP_RET_SCP(a,b,len) \
    strncpy((char *)dataP->b, (char *)oraStructP->a, len); \
    dataP->b[(len)-1] = '\0'
#define TPCCP_CP(a,b) oraStructP->a = dataP->b
#define TPCCP_SCP(a,b,len) strncpy((char *)oraStructP->a, (char *)dataP->b, len)
#define TPCCP_RET_CP(a,b) dataP->b = oraStructP->a

/*
 * Talk to Oracle
 */
int get_db_ready(char *dbName, int flag)
{
    int rc;
    char dvryFileName[100];
    extern char *tpcc_serverName;

    AUDITLOG(("> get_db_ready to %s flag %d\n", dbName, flag));
    if (server_null_test) return(0);

    fprintf(stderr, ">> get_db_ready, db: %s, flag %d\n", dbName, flag);

    sprintf(dvryFileName, "home/encina/runs/deliveries/%s",
        tpcc_serverName);
    rc = TPCinit(serverIdNumber, "tpcc", "tpcc", dvryFileName);
    err_printf("TPCinit(%d, tpcc, tpcc, %s) returned %d\n",
        serverIdNumber, dvryFileName, rc);
    if (rc) {
        fprintf(stderr, "TPCinit(%d, tpcc, tpcc, %s) returned %d\n",
            serverIdNumber, dvryFileName, rc);
    }

    AUDITLOG(("< get_db_ready rc %d\n", rc));
    return(rc);
}

void *create_connection() {
    return create_ora_connection();
}

void do_delivery(enP, dataP)
void *cnP;
delivery_data_t *dataP;
{
```



```

struct delstruct oraStruct;
int rc;

AUDITLOG("> do_delivery\n");

TPCC_CP(delin.w_id, w_id);
TPCC_CP(delin.o_carrier_id, o_carrier_id);
TPCC_CP(delin.qtime, start_queue);
TPCC_CP(delin.in_timing_int, queued_time);

DPRINT(("Calling TPCdel: w_id %d, o_carrier_id %d, %f qtime, %d in_timing_int",
oraStruct.delin.w_id, oraStruct.delin.o_carrier_id,
oraStruct.delin.qtime, oraStruct.delin.in_timing_int));

rc = TPCdel(cnP, &oraStruct);
if (rc != 0) && (rc != -666) {
    err_printf("Error TPCdel: terror %d, rc %d, retry %d, w_id %d, o_carrier_id %d, %f qtime,
%d in_timing_int\n",
oraStruct.delout.terror, rc, oraStruct.delout.retry,
oraStruct.delin.w_id, oraStruct.delin.o_carrier_id,
oraStruct.delin.qtime, oraStruct.delin.in_timing_int);
}
dataP->header.returncode = rc == 0 ? TPCC_SUCCESS : oraStruct.delout.terror;
AUDITLOG("< do_delivery rc %d\n", rc);
}

void copyout_order_status(orderStatus_data_t *dataP,
struct ordstruct *oraStructP)
{
    int i;
    TPCCP_RET_CP(ordout.c_balance, c_balance);
    TPCCP_RET_CP(ordout.o_id, o_id);
    TPCCP_RET_CP(ordout.o_carrier_id, o_carrier_id);
    TPCCP_RET_CP(ordout.o_ol_cnt, o_ol_cnt);
    TPCCP_RET_CP(ordout.c_id, c_id);
#define I_CP(ind, a, b) dataP->item[ind].b = oraStructP->ordout.a[ind]
#define I_SCP(ind, a, b, len) \
strcpy((char *)dataP->item[ind].b, (char *)oraStructP->ordout.a[ind], len); \
dataP->item[ind].b[(len) - 1] = '\0'
    for (i=0; i<oraStructP->ordout.o_ol_cnt && i < 15; i++) {
        I_CP(i, ol_amount, ol_amount);
        I_CP(i, ol_i_id, ol_i_id);
        I_CP(i, ol_supply_w_id, ol_supply_w_id);
        I_CP(i, ol_quantity, ol_quantity);
        I_SCP(i, ol_delivery_d, delivery_date, 11);
    }
#undef I_CP
#undef I_SCP
    TPCCP_RET_SCP(ordout.c_first, c_first, 17);
    TPCCP_RET_SCP(ordout.c_middle, c_middle, 3);
    TPCCP_RET_SCP(ordout.c_last, c_last, 17);
    TPCCP_RET_SCP(ordout.o_entry_d, entry_date, 20);
}

void do_order_status(cnP, dataP)
void *cnP;
orderStatus_data_t *dataP;
{
    struct ordstruct oraStruct;
    int i, rc;

    AUDITLOG("> do_order_status\n");

    TPCC_CP(ordin.w_id, w_id);
    TPCC_CP(ordin.d_id, d_id);
    TPCC_CP(ordin.c_id, c_id);
    oraStruct.ordin.bylastname = ((dataP->c_id == 0) ? 1 : 0);
    TPCC_SCP(ordin.c_last, c_last, 17);

    DEBUGP(("Calling TPCord: w_id %d, d_id %d, c_id %d, bylastname %d, c_last %s\n",
oraStruct.ordin.w_id, oraStruct.ordin.d_id, oraStruct.ordin.c_id, oraStruct.ordin.bylastname,
oraStruct.ordin.c_last));

    rc = TPCord(cnP, &oraStruct);
    if (rc != 0) {
        err_printf("Error TPCord: terror %d, rc %d, retry %d, w_id %d, d_id %d, c_id %d, bylastname %d,
c_last %s\n",
oraStruct.ordout.terror, rc, oraStruct.ordout.retry,
oraStruct.ordin.w_id, oraStruct.ordin.d_id, oraStruct.ordin.c_id,
oraStruct.ordin.bylastname, oraStruct.ordin.c_last);
    }

    copyout_order_status(dataP, &oraStruct);

    dataP->header.returncode = rc == 0 ? TPCC_SUCCESS : oraStruct.ordout.terror;
    AUDITLOG("< do_order_stats rc %d\n", dataP->header.returncode);
}

void do_stock_level(cnP, dataP)
void *cnP;
stockLevel_data_t *dataP;
{
    struct stostruct oraStruct;
    /* What's this comment?? -- srs: i only did this one to check the links */
    int rc;

    AUDITLOG("> do_stock_level\n");

    TPCC_CP(stoin.w_id, w_id);
    TPCC_CP(stoin.d_id, d_id);
    TPCC_CP(stoin.threshold, threshold);

    DEBUGP(("Calling TPCsto: w_id %d, d_id %d, threshold %d\n",
oraStruct.stoin.w_id, oraStruct.stoin.d_id,
oraStruct.stoin.threshold));

    rc = TPCsto(cnP, &oraStruct);
    if (rc != 0) {
        err_printf("Error TPCsto: terror %d, rc %d, retry %d, w_id %d, d_id %d, threshold %d\n",
oraStruct.stout.terror, rc, oraStruct.stout.retry,
oraStruct.stoin.w_id, oraStruct.stoin.d_id,
oraStruct.stoin.threshold);
    }

    TPCC_RET_CP(stout.low_stock, stock_count);

    dataP->header.returncode = rc == 0 ? TPCC_SUCCESS : oraStruct.stout.terror;

    DEBUGP(("do_stock_lev returning %d\n", dataP->header.returncode));
    AUDITLOG("< do_stock_level rc %d\n", dataP->header.returncode);
}

void copyin_payment(dataP, oraStructP)
payment_data_t *dataP;
struct paystruct *oraStructP;
{
    TPCCP_CP(payin.w_id, w_id);
    TPCCP_CP(payin.d_id, d_id);
    TPCCP_CP(payin.c_w_id, c_w_id);
    TPCCP_CP(payin.c_d_id, c_d_id);
    TPCCP_CP(payin.c_id, c_id);
    oraStructP->payin.bylastname = ((dataP->c_id == 0) ? 1 : 0);
    TPCCP_CP(payin.h_amount, h_amount);
    TPCCP_SCP(payin.c_last, c_last, 17);
}

void copyout_payment(dataP, oraStructP)
payment_data_t *dataP;
struct paystruct *oraStructP;
{
    TPCCP_RET_SCP(payout.w_street_1, w_street_1, 21);
    TPCCP_RET_SCP(payout.w_street_2, w_street_2, 21);
    TPCCP_RET_SCP(payout.w_city, w_city, 21);
    TPCCP_RET_SCP(payout.w_state, w_state, 3);
    TPCCP_RET_SCP(payout.w_zip, w_zip, 10);
    TPCCP_RET_SCP(payout.d_street_1, d_street_1, 21);
    TPCCP_RET_SCP(payout.d_street_2, d_street_2, 21);
    TPCCP_RET_SCP(payout.d_city, d_city, 21);
    TPCCP_RET_SCP(payout.d_state, d_state, 3);
    TPCCP_RET_SCP(payout.d_zip, d_zip, 10);
    TPCCP_RET_CP(payout.c_id, c_id);
    TPCCP_RET_SCP(payout.c_first, c_first, 17);
    TPCCP_RET_SCP(payout.c_middle, c_middle, 3);
    TPCCP_RET_SCP(payout.c_last, c_last, 17);
    TPCCP_RET_SCP(payout.c_street_1, c_street_1, 21);
    TPCCP_RET_SCP(payout.c_street_2, c_street_2, 21);
    TPCCP_RET_SCP(payout.c_city, c_city, 21);
    TPCCP_RET_SCP(payout.c_state, c_state, 3);
    TPCCP_RET_SCP(payout.c_zip, c_zip, 10);
    TPCCP_RET_SCP(payout.c_phone, c_phone, 17);
    TPCCP_RET_SCP(payout.c_since, c_date, 11);
    TPCCP_RET_SCP(payout.c_credit, c_credit, 3);
    TPCCP_RET_CP(payout.c_credit_lim, c_credit_lim);
    TPCCP_RET_CP(payout.c_discount, c_discount);
    TPCCP_RET_CP(payout.c_balance, c_balance);
    TPCCP_RET_SCP(payout.c_data, c_data, 201);
    TPCCP_RET_SCP(payout.h_date, pay_date, 20);
    strcpy((char *)dataP->w_name, "W_NAME");
    strcpy((char *)dataP->d_name, "D_NAME");
    /* Ignore c_ytd_payment, c_payment_cnt */
}

void do_payment(cnP, dataP)
void *cnP;
payment_data_t *dataP;
{
    struct paystruct oraStruct;
    int firstWh, secondWh;
    int rc;

    AUDITLOG("> do_payment\n");

    copyin_payment(dataP, &oraStruct);

    #if 0
    err_printf("TPCpay: w_id %d, D_id %d, C_w_id %d, c_id %d, bylastname %d, amount %.2f, c_last %s
(%s)\n",
oraStruct.payin.w_id,
oraStruct.payin.d_id,
oraStruct.payin.c_w_id,
oraStruct.payin.c_id,
oraStruct.payin.bylastname,
oraStruct.payin.h_amount,
oraStruct.payin.c_last,
dataP->c_last);
    #endif
}

```

```

rc = TPCpay(cnP, &oraStruct);

#if 0
err_printf("< TPCpay terror %d, rc %d, retry %d\n",
oraStruct.payout.terror, rc, oraStruct.payout.retry);
#endif

dataP->header.num_rms = 1;
if (rc != 0) {
err_printf("Error TPCpay: terror %d, rc %d, retry %d, w_id %d, D_id %d, C_w_id %d, c_id %d,
bylastname %d, amount %.2f, c_last %s (%s)\n",
oraStruct.payout.terror, rc, oraStruct.payout.retry,
oraStruct.payout.w_id,
oraStruct.payout.d_id,
oraStruct.payout.c_w_id,
oraStruct.payout.c_id,
oraStruct.payout.bylastname,
oraStruct.payout.h_amount,
oraStruct.payout.c_last ? oraStruct.payout.c_last : "-NULL-",
(char *)dataP->c_last ? (char *)dataP->c_last : "-NULL-");
}

copyout_payment(dataP, &oraStruct);

dataP->header.returncode = rc == 0 ? TPCC_SUCCESS : oraStruct.payout.terror;
AUDITLOG("< do_payment rc %d\n", dataP->header.returncode);

static void copyin_new_order(dataP, oraStructP)
newOrder_data_t *dataP;
struct newstruct *oraStructP;
{
int i;

TPCCP_CP(newin.w_id, w_id);
TPCCP_CP(newin.d_id, d_id);
TPCCP_CP(newin.c_id, c_id);

#define NO_I_CP(ind,a,b) oraStructP->a[ind] = dataP->item[ind].b
#define NO_I_SCP(ind,a,b,len) strcpy((char *)oraStructP->a[ind], (char *)dataP->item[ind].b, len)

/* tpccpl.c loops over 15 items, we do the same */
for (i=0; i<15; i++) {
NO_I_CP(i, newin.ol_i_id, ol_i_id);
NO_I_CP(i, newin.ol_supply_w_id, ol_supply_w_id);
NO_I_CP(i, newin.ol_quantity, ol_quantity);
#ifdef DEBUG_SERVER
fprintf(stderr, "NewOrder: Item %d, supplyWh %d (local %d)\n",
i,
oraStructP->newin.ol_supply_w_id[i],
oraStructP->newin.w_id);
#endif
}
/* Ignore all local field, total_items,
* tpccpl.c doesnt use them
*/
#undef NO_I_CP
#undef NO_I_SCP
}

void copyout_new_order(dataP, oraStructP)
newOrder_data_t *dataP;
struct newstruct *oraStructP;
{
int i;

TPCCP_RET_CP(newout.o_id, o_id);
TPCCP_RET_CP(newout.ol_cnt, ol_cnt);
TPCCP_RET_SCP(newout.c_last, c_last, 17);
TPCCP_RET_SCP(newout.c_credit, c_credit, 3);
TPCCP_RET_CP(newout.c_discount, c_discount);
TPCCP_RET_CP(newout.w_tax, w_tax);
TPCCP_RET_CP(newout.d_tax, d_tax);
TPCCP_RET_SCP(newout.o_entry_d, entry_date, 20);
TPCCP_RET_CP(newout.total_amount, total);
TPCCP_RET_SCP(newout.status, statusline, 26);

#define NO_RET_CP(ind,a,b) dataP->item[ind].b = oraStructP->newout.a[ind]
#define NO_RET_SCP(ind,a,b,len) strncpy((char *)dataP->item[ind].b, (char *)oraStructP->newout.a[ind], len)

for (i=0; i<oraStructP->newout.ol_cnt && i<15; i++) {
NO_RET_SCP(i, i_name, name_i, 25);
NO_RET_CP(i, s_quantity, s_quantity);
dataP->item[i].brand_generic[0] = oraStructP->newout.brand_generic[i];
dataP->item[i].brand_generic[1] = '0';
NO_RET_CP(i, i_price, price);
NO_RET_CP(i, ol_amount, ol_amount);
/* Ignore s_idx and s_dist */
}
if (oraStructP->newout.status[0] != '0') {
DEBUGP(("TPCnew: status -- %s\n", oraStructP->newout.status));
dataP->items_valid = 0;
} else {
dataP->items_valid = 1;
}
}

#undef NO_RET_CP
#undef NO_RET_SCP
}

void do_new_order(cnP, dataP)
void *cnP;
newOrder_data_t *dataP;
{
static int num_calls = 0;
int i;
struct newstruct oraStruct;
int rc;

AUDITLOG("> do_new_order\n");

/* Copy the structure into the TPCC structure. */
copyin_new_order(dataP, &oraStruct);

DEBUGP("> TPCnew %d items to wh %d\n",
dataP->o_ol_cnt, dataP->w_id);
dataP->header.num_rms = 1;

#if 0
err_printf("Error TPCnew : w_id %d, d_id %d, c_id %d, o_ol_cnt %d (out cnt %d)\n",
oraStruct.newin.w_id, oraStruct.newin.d_id,
oraStruct.newin.c_id, dataP->o_ol_cnt, oraStruct.newout.o_ol_cnt);
for (i=0; i<15; i++) {
err_printf("ol_i_id %d, ol_supply_w_id %d, ol_quantity %d\n",
oraStruct.newin.ol_i_id[i], oraStruct.newin.ol_supply_w_id[i],
oraStruct.newin.ol_quantity[i]);
}
#endif

rc = TPCnew(cnP, &oraStruct);

#if 0
err_printf("< TPCnew terror %d, rc %d, retry %d\n",
oraStruct.newout.terror, rc, oraStruct.newout.retry);
#endif

if (rc != 0) {
err_printf("Error TPCnew : terror %d, rc %d, retry %d, w_id %d, d_id %d, c_id %d, o_ol_cnt %d (out
cnt %d)\n",
oraStruct.newout.terror, rc, oraStruct.newout.retry,
oraStruct.newin.w_id, oraStruct.newin.d_id,
oraStruct.newin.c_id, dataP->o_ol_cnt, oraStruct.newout.o_ol_cnt);
for (i=0; i<15; i++) {
err_printf("ol_i_id %d, ol_supply_w_id %d, ol_quantity %d\n",
oraStruct.newin.ol_i_id[i], oraStruct.newin.ol_supply_w_id[i],
oraStruct.newin.ol_quantity[i]);
}
}
DEBUGP("< TPCnew %d\n", rc);

/* copy out results */
copyout_new_order(dataP, &oraStruct);

if (rc == 0) {
dataP->header.returncode =
dataP->items_valid ? TPCC_SUCCESS : INVALID_NEWO;
}

#if 0
if (dataP->items_valid && (++num_calls % 500) == 0) {
int i;
err_printf("TPCnew Success: w_id %d, d_id %d, c_id %d, o_ol_cnt %d, Oid %d\n",
oraStruct.newin.w_id, oraStruct.newin.d_id,
oraStruct.newin.c_id, oraStruct.newout.o_ol_cnt,
oraStruct.newout.o_id);
for (i=0; i<15 && i<oraStruct.newout.o_ol_cnt; i++) {
err_printf(" %2d: i_id %i5d, sw_id %4d, qty %d, price %.2f amt %.2f\n",
i, oraStruct.newin.ol_i_id[i],
oraStruct.newin.ol_supply_w_id[i],
oraStruct.newin.ol_quantity[i],
oraStruct.newout.i_price[i],
oraStruct.newout.ol_amount[i]);
}
}
#endif
} else {
dataP->header.returncode = oraStruct.newout.terror;
}

AUDITLOG("< do_new_order rc %d\n", dataP->header.returncode);
}

/* (C)1997 IBM Corporation */
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <ctype.h>
#include <string.h>
#include <math.h>

```

**client.C**

```

#include "screen.h"
#include "encina.h"

extern "C" void set_client_debug_state(void *contextP, int state, int tran);

Encina encina;

extern "C" int client_login(int infd, int outfd, int *w_idP, int *d_idP)
{
    Thread_data thread(infd, outfd, NULL);
    User_data user_data;
    Login log(&user_data, &thread);
    log.handle();
    *w_idP = user_data.warehouse;
    *d_idP = user_data.district;
    return 0;
}

extern "C" int client_init(int infd, int outfd, int w_id, int d_id,
                          void *contextP) {
    int rc = 0;
    Thread_data *threadP = new Thread_data(infd, outfd, contextP);
    Field *menuField = new IntField(threadP, 8);
    User_data user_data;
    Menu menu(&user_data, threadP);

    user_data.warehouse = w_id;
    user_data.district = d_id;
    menu.present();

    Payment pay(&user_data, threadP);
    Delivery del(&user_data, threadP);
    OrderStatus os(&user_data, threadP);
    StockLevel sl(&user_data, threadP);
    NewOrder no(&user_data, threadP);

    while (rc == 0) {
        int key = menuField->get_key();
        set_client_debug_state(contextP, 1, key - '0');
        switch (key) {
            case EOF: rc = -1; break;
            case '1': case 'N': case 'n': rc = no.handle(); break;
            case '2': case 'P': case 'p': rc = pay.handle(); break;
            case '3': case 'O': case 'o': rc = os.handle(); break;
            case '4': case 'D': case 'd': rc = del.handle(); break;
            case '5': case 'S': case 's': rc = sl.handle(); break;
            case '030':
                position(threadP, 1, 1);
                threadP->flush(); break;
            case '9': case 'Q': case 'q': case 'E': case 'e':
                return(0);
            default: threadP->write("a", 1); break;
        }
        set_client_debug_state(contextP, 4, key - '0');
    }
    return 0;
}

```

## client.h

```

*
* client.h
*
* SRevision: 1.6 $
* SDate: 1998/1/24 21:45:58 $
* SLog: $
*
* STALog: client.h.v $
* Revision 1.6 1998/11/24 21:45:58 wenjian
* - Add MULTIPLE_INTERFACE compile flag
* [from r1.5 by delta wenjian-23742-TPCC-update-with-Raleigh-code, r1.1]
*
* Revision 1.5 1998/11/09 14:48:13 wenjian
* In an effort to make a new directory structure for TPCC, this delta
* creates two directories: tpcc/client and tpcc/server. All the files
* for this revision are copied from tpcc/sp-tpcc without any change.
* Further change may be needed for some files due to the change of
* the directory structure.
* [added by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.1]
*
* Revision 1.10 1998/10/26 14:41:34 dongfeng
* Add Init command in web client so when something bad happens during
* initialization web client sends back error information and allows
* reinitialization instead of killing IIS server.
*
* Define values of initialization errors.
* [from r1.9 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.3]
*
* Revision 1.9 1998/10/22 20:55:53 wenjian
* [merge of changes from 1.5 to 1.8 into 1.6]
*
* Revision 1.8 1998/10/22 19:18:31 dongfeng
* [from r1.7 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.2]
*
* Revision 1.7 1998/10/08 14:17:59 dongfeng
* Add codes for doing web-based tpcc.
* [from r1.5 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.1]

```

```

*
* Revision 1.6 1998/09/26 10:56:25 oz
* - renamed thread_init and thread_done to clnt_thread_init and
* clnt_thread_done respectively because of name conflicts on AIIX4.3
* [from r1.5 by delta oz-23339-TPCC-update-for-NT, r1.2]
*
* Revision 1.5 1998/01/26 16:19:22 oz
* - moved all the code pertaining to the background
* thread to its own file and all the data structures
* to client_utils.h
* [from r1.4 by delta oz-21689-TPCC-move-client-bg-thread-to-separate-file, r1.1]
*
* Revision 1.4 1998/01/23 15:07:43 oz
* - Updated the SP TPCC directory to the latest files used
* during the SP tpcc audit.
* [from r1.3 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
*
*
*
*
*/

#ifdef TPCC_CLIENT_H
#define TPCC_CLIENT_H

#ifdef (solaris)
#include <dce/ptthread.h>
#else /* solaris */
#include <pthread.h>
#endif /* solaris */

#ifdef WEB_TPCC_CLIENT
#include <windows.h>

#define MUTEX_T CRITICAL_SECTION
#define MUTEX_LOCK(a) EnterCriticalSection(a)
#define MUTEX_UNLOCK(a) LeaveCriticalSection(a)
#define MUTEX_INIT(mut) InitializeCriticalSection(mut)
#define MUTEX_DESTROY(mut) DeleteCriticalSection(mut)
#define ERROUT errtpcc

/* initialization status */
#define INIT_SUCCESS 0
#define INIT_FAILED 1
#define CELL_NAME_UNAVAILABLE 2
#define MON_RETRIEVEENABLE_FAILED 3
#define MON_INITCLIENT_FAILED 4
#define MON_SECURITYSET_FAILED 5
#define MON_SETREFRESHINTERVAL_FAILED 6
#define NOINFO_TRPC_ERROR 7
#define ENROLL_CLIENT_EXCEPTION 8
#define ERROUT_FILE_NOT_FOUND 9
#define LOG_FILE_NOT_FOUND 10
#define TPCC_KEY_NOT_FOUND 11
#define TERM_ALLOC_FAILED 12

#else
#define MUTEX_T pthread_mutex_t
#define COND_T pthread_cond_t
#define MUTEX_LOCK(a) pthread_mutex_lock(a)
#define MUTEX_UNLOCK(a) pthread_mutex_unlock(a)
#define COND_WAIT(cond,mut) pthread_cond_wait(cond,mut)
#define COND_SIGNAL(cond) pthread_cond_signal(cond)
#define COND_BROADCAST(cond) pthread_cond_broadcast(cond)
#define MUTEX_INIT(mut) pthread_mutex_init(mut, pthread_mutexattr_default)
#define COND_INIT(cond) pthread_cond_init(cond, pthread_condattr_default)
#define MUTEX_DESTROY(mut) pthread_mutex_destroy(mut)
#define COND_DESTROY(cond) pthread_cond_destroy(cond)
#define ERROUT stderr
#endif

/*
* Routines and declarations that are common to all clients
*/
void *clnt_thread_init(void);

void clnt_thread_done(void *);

void *terminal_context_init(int);

void send_new_order(void *, newOrder_data_t *);

void send_payment(void *, payment_data_t *);

void send_order_status(void *, orderStatus_data_t *);

void send_delivery(void *, delivery_data_t *);

void send_stock_level(void *, stockLevel_data_t *);

void send_batch_request(void *contextP, int num, tpcc_data_t *dataP);

void send_unmarshalled(void *contextP,
                        int tran_type,
                        int size,
                        ndr_byte *dataP);

void enroll_client(int id);

```

```
#endif /* TPCC_CLIENT_H */
```

## client\_bg\_thread.c

```
*
* client_bg_thread.c
*
* $Revision: 1.17 $
* $Date: 1999/05/06 21:28:25 $
* $Log: $
*
*
* $TALog: client_bg_thread.c.v $
* Revision 1.17 1999/05/06 21:28:25 oz
* - Removed all the .. from the includes
* - Added -L. to the makefiles instead
* - Moved all the thread related code and connection
* selection to serverMon.c
* [from r1.13 by delta oz-24309-TPCC-add-oracle8.1-code, r1.5]
*
* Revision 1.13 1999/01/29 20:16:31 wenjian
* Small changes to make check_threads more robust
* [from r1.12 by delta wenjian-23787-TPCC-integrate-code-for-AIX-and-NT, r1.7]
*
* Revision 1.12 1998/12/28 20:07:11 wenjian
* - Change client_info to a pointer pClientInfo for flexibility.
* [from r1.11 by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.5]
*
* Revision 1.11 1998/12/14 20:27:53 wenjian
* Made corresponding changes due to data structure change of tran_info_t.
* [from r1.10 by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.3]
*
* Revision 1.10 1998/12/11 16:14:18 wenjian
* Add code for checking statistic data in a single variable and collecting
* statistic data based on iStatsFrequency.
*
* - Add new check_threads() in order to use the single statistic var.
* [from r1.9 by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.1]
*
* Revision 1.9 1998/12/08 23:03:48 wenjian
* Add (or rename) Makefile for each platform (AIX and NT). Reorganize the
* files a little bit.
*
* - Change path for tran_stat.h
* - Add #ifdef before getStatsForm since it is only used by NT
* [from r1.8 by delta wenjian-23787-TPCC-integrate-code-for-AIX-and-NT, r1.1]
*
* Revision 1.8 1998/12/07 20:04:11 wenjian
* Clean up
* [from r1.7 by delta wenjian-23742-TPCC-update-with-Raleigh-code, r1.2]
*
* Revision 1.7 1998/11/24 21:45:58 wenjian
* - Add #ifdef MULTIPLE_INTERFACE
* - Take care special case for check_threads
* [from r1.6 by delta wenjian-23742-TPCC-update-with-Raleigh-code, r1.1]
*
* Revision 1.6 1998/11/09 16:59:35 wenjian
* In this revision, most of the changes are related to the directory of header
* files after directory reorganization. Other changes include adding or removing
* files to put them in the right directories. Makefiles are written for NT
* platform so that nmake is working on NT now. Need a top level Makefile for all
* the directories.
* [from r1.5 by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.2]
*
* Revision 1.5 1998/11/09 14:48:14 wenjian
* In an effort to make a new directory structure for TPCC, this delta
* creates two directories: tpcc/client and tpcc/server. All the files
* for this revision are copied from tpcc/sp-tpcc without any change.
* Further change may be needed for some files due to the change of
* the directory structure.
* [added by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.1]
*
* Revision 1.29 1998/11/06 16:10:54 wenjian
* - Move gettimeofday() in check_threads out of the for loop
* - Minor change for cleanup
* [from r1.28 by delta wenjian-23646-TPCC-clean-up-source-code, r1.1]
*
* Revision 1.28 1998/10/26 15:17:53 dongfeng
* remove #include <sys/times.h> from NT tests
* [from r1.27 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.4]
*
* Revision 1.27 1998/10/22 21:05:37 wenjian
* [merge of changes from 1.12 to 1.26 into 1.25]
*
* Revision 1.26 1998/10/22 19:18:31 dongfeng
* [from r1.24 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.2]
*
* Revision 1.24 1998/10/08 14:17:59 dongfeng
* Add codes for doing web-based tpcc.
* [from r1.12 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.1]
*
* Revision 1.25 1998/10/08 18:03:00 gerstl
* Changes to allow configurations where some servers only service
```

```
* specific transaction types. Split transaction interfaces by type.
* [from r1.23 by delta gerstl-23515-TPCC-allow-separate-online-transaction-interfaces, r1.1]
*
* Revision 1.23 1998/09/03 16:07:11 wenjian
* Change GET_USER_SYS_TIME to GET_CLIENT_USAGE
* [from r1.19 by delta wenjian-23183-TPCC-sync-AIX-code-with-Austin, r1.3]
*
* Revision 1.19 1998/08/18 13:35:41 wenjian
* - Clean up including header files
* - Remove client_report() and socket_print_rt_avg()
* - Use #ifdef for the call of getUserSysTime()
* [from r1.16 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients, r1.7]
*
* Revision 1.16 1998/07/08 18:15:42 wenjian
* Add getUserSysTime().
* [from r1.15 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients, r1.6]
*
* Revision 1.15 1998/07/02 18:28:51 wenjian
* Change client_status_report to send more information of the
* client process. These changes are matched with changes in
* tpcc_monitor.c.
* [from r1.9 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients, r1.5]
*
* Revision 1.7 1998/04/29 19:47:40 wenjian
* - Add client_status_report to communicate with tpcc_monitor
* - Add socket_print_rt_avg
* [from r1.6 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients, r1.1]
*
* Revision 1.6 1998/02/17 22:12:40 wenjian
* [merge of changes from 1.3 to 1.4 into 1.5]
*
* Revision 1.4 1998/02/17 16:04:40 oz
* - Split the login into two parts to allow for special logins
* - If the warehouse ID is 0, this is a special login to
* query the client for status
*
* - check_threads: Return the number of threads
* - New function: client_report
* [from r1.3 by delta oz-21864-TPCC-split-client-login-screen, r1.1]
*
* Revision 1.5 1998/02/17 22:06:59 wenjian
* Add necessary head files for win32
* [from r1.3 by delta wenjian-21750-TPCC-changes-for-porting-on-NT, r1.1]
*
* Revision 1.3 1998/01/29 22:53:34 oz
* - Use pthread delay instead of sleep
* [from r1.2 by delta oz-21749-TPCC-use-pthread-delay-for-bg-thread, r1.1]
*
* Revision 1.2 1998/01/26 20:37:34 oz
* - Remove all the code associated with explicit binding
*
* - Removed include of mon_client_utils.h
* [from r1.1 by delta oz-21697-TPCC-remove-explicit-binding-code, r1.1]
*
* Revision 1.1 1998/01/26 16:19:22 oz
* - moved all the code pertaining to the background
* thread to its own file and all the data structures
* to client_utils.h
* [added by delta oz-21689-TPCC-move-client-bg-thread-to-separate-file, r1.1]
*
*/
/*
/* client_bg_thread
/* A file used for debug purposes only.
*
* It implements a background thread that once a minute checks the
* state of all the threads and reports the state of the client.
*
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include <time.h>
#ifdef WIN32
#include <sys/times.h> /* for getUserSysTime */
#endif
#ifdef defined (solaris)
#include <dlc/pthread.h>
#else
#include <pthread.h>
#endif /* solaris */
#include <tpm/mon/mon.h>
#include <utils/trace.h>
#include "common/delivery.h"
#ifdef MULTIPLE_INTERFACE
#include "common/neworder.h"
#include "common/payment.h"
#include "common/stocklevel.h"
#include "common/orderstatus.h"
#else
#include "common/tpcc_trans.h"
#endif
#include "common/utilities.h"
#include "client_utils.h"
```

```

#include "common/do_tpcc.h"
#include "client.h"
#include "encina_client.h"
#include "tools/tran_stat.h"
#include "common/get_local_time.h"

#ifdef WIN32
#define read(A,B,C) recv(A,B,C,0)
#define write(A,B,C) send(A,B,C,0)
#endif

#if 1
#define PRINT_AV(total, num, str) \
{ \
    if((num) > 0) { \
        fprintf(ERROUT, " %s %.0f", str, (double)(total)/(num)); \
    } \
}
#else
#define PRINT_AV(a,b,c)
#endif

static void check_threads(total_tran_count_t *tran_ctP, int *numP, int *numInitP);
static struct timeval *client_last_time(thread_descr_t *descrP);
void getUserSysTime(struct timeval *user_time, struct timeval *sys_time);

/*
 * client_last_time
 *
 * Each thread maintains the current state it is in and the time
 * it entered this state.
 * This routine returns a pointer to the structure in the thread
 * data that contains the time corresponding to the threads current
 * state.
 * Typical use:
 * - Set the state, then call gettimeofday on the pointer
 * returned by this function.
 */
static struct timeval *client_last_time(thread_descr_t *descrP)
{
    struct timeval *lastTimeP = &descrP->done;
    switch (descrP->state) {
        case thread_state_init: /* Thread is initializing - no trans yet */
            lastTimeP = &descrP->init;
            break;
        case thread_state_called: /* Tran type was sent by the RTE */
            lastTimeP = &descrP->called;
            break;
        case thread_state_returned: /* Final screen sent to RTE */
            lastTimeP = &descrP->returned;
            break;
        case thread_state_sent: /* Sent to server */
            lastTimeP = &descrP->sent;
            break;
        case thread_state_received: /* Received reply from server */
            lastTimeP = &descrP->received;
            break;
        case thread_state_done: /* The thread exited */
            lastTimeP = &descrP->done;
            break;
        default:
            err_printf("client_last_time: bad state: %d\n", descrP->state);
            lastTimeP = &descrP->done;
            break;
    }
    return(lastTimeP);
}

void set_client_debug_state(void *contextP, int state, int tran)
{
    thread_info_t *thread_context = (thread_info_t *)contextP;
    struct timezone tz;
    thread_descr_t *descrP = &thread_context->descr;

    descrP->state = state;

    gettimeofday(client_last_time(descrP), &tz);
    if (state == thread_state_called) descrP->tran = tran;
}

/* How often to report the state of a thread:
 * If it is in the thread_state_init phase: report if it has been in
 * that state for more than 5 minutes.
 * Report if it takes the terminal more than 3 minutes to generate the next
 * transaction. Otherwise, report if anything takes longer than 60 seconds.
 */
#define THREAD_STATE_REPORT_DELTA(state) \
((state) == thread_state_init ? 300 : \
(state) == thread_state_returned ? 180 : 60)

static char *thread_state_to_str(int state)
{
    char *ret_val = "-Unknown-";
    switch(state) {
        case thread_state_init: ret_val = "state_init"; break;
        case thread_state_called: ret_val = "state_called"; break;
        case thread_state_returned: ret_val = "state_returned"; break;
        case thread_state_sent: ret_val = "state_sent"; break;
        case thread_state_received: ret_val = "state_received"; break;
        case thread_state_done: ret_val = "state_done"; break;
        case thread_state_returned: ret_val = "state_returned"; break;
    }
    return(ret_val);
}

static void print_rt_avg(total_tran_count_t *curP,
                        total_tran_count_t *prevP,
                        int type)
{
    int i;
    static char *names[] = {"0", "no", "pa", "os", "dl", "sl"};
    err_printf("%s RT avg: ", type ? "server" : "client");

    for (i=1; i<=MAX_TRAN_TYPE; i++) {
        int num_trans = curP->tran[i].RTcount - prevP->tran[i].RTcount;
        double rt_diff = curP->tran[i].RTtotal[type] - prevP->tran[i].RTtotal[type];
        PRINT_AV(rt_diff, num_trans, names[i]);
    }
    fprintf(ERROUT, "\n");
}

/*
 * A background thread that keeps tabs on the state of all the
 * threads of the client. (For Debug)
 */
static void *bg_thread(void *argP)
{
    static int sleep_time = 60000; /* in ms */
    static struct timespec time_wait = {60, 0};
    struct timespec sleep_end;

    total_tran_count_t tran_ct, tran_reported[2];
    int total_newo, total_tran_err;
    struct timeval cur_time;
    struct timezone tz;
    struct timeval time_reported[2];

    gettimeofday(&time_reported[0], &tz);
    time_reported[1] = time_reported[0];

    memset(&tran_reported[0], ^0, 2 * sizeof(tran_reported[0]));

    while (1) {
        double time_diff1, time_diff2;
        double tran_diff1, tran_diff2;
        double prev_newo1, prev_newo2;
        double err_diff1, err_diff2;

        check_threads(&tran_ct, NULL, NULL);

        total_tran_err = tran_ct.errors;
        total_newo = tran_ct.tran[NEWO_TRANS].num-tran_ct.tran[NEWO_TRANS].errs;

        gettimeofday(&cur_time, &tz);
        time_diff1 = time_diff_ms(&cur_time, &time_reported[0]);
        prev_newo1 = tran_reported[0].tran[NEWO_TRANS].num -
        tran_reported[0].tran[NEWO_TRANS].errs;
        tran_diff1 = total_newo - prev_newo1;
        err_diff1 = total_tran_err - tran_reported[0].errors;

        time_diff2 = time_diff_ms(&cur_time, &time_reported[1]);
        prev_newo2 = tran_reported[1].tran[NEWO_TRANS].num -
        tran_reported[1].tran[NEWO_TRANS].errs;
        tran_diff2 = total_newo - prev_newo2;
        err_diff2 = total_tran_err - tran_reported[1].errors;
        if (total_newo != 0 && tran_diff2 > 0) {
            err_printf("bg_thread: TPM: %.0f (last %.0f sec), %.0f (last %.0f sec)\n",
                tran_diff1 / time_diff1 * 60000, time_diff1 / 1000.,
                tran_diff2 / time_diff2 * 60000, time_diff2 / 1000.);
            /* print av server response time for all transactions */
            print_rt_avg(&tran_ct, &tran_reported[1], 0);
            print_rt_avg(&tran_ct, &tran_reported[1], 1);
        }

        if (err_diff2 != 0) {
            err_printf("bg_thread: errPM %.1f (last %.0f sec)\n",
                err_diff2 / time_diff2 * 60000, time_diff2 / 1000.);
        }

        tran_reported[0] = tran_reported[1];
        tran_reported[1] = tran_ct;
        time_reported[0] = time_reported[1];
        time_reported[1] = cur_time;

        pthread_delay_np(&time_wait);
    }
}

#ifdef KEEP_TERMINAL_INFO
static void check_threads(total_tran_count_t *tran_ctP, int *num_threadsP, int *num_threadsInitP)
{
    struct timezone tz;
    int num_per_state[NUM_STATES];
    int total_stuck = 0;
    static int init_printed = 0;
    int total_tran_err;
    int num_active = 0;
}
#endif

```

<pre> MUTEX_LOCK(&amp;init_lock);  if (info_list == NULL    (info_list_len &lt; 1)) {     if (num_threadsP)         *num_threadsP = 0;     if (num_threadsInitP)         *num_threadsInitP = 0;     memset(tran_ctP, '0', sizeof(*tran_ctP)); } else {     int ij;     struct timeval cur_time;     int num_init = 0, num_done = 0;      for (i=0; i&lt;NUM_STATES; i++) num_per_state[i] = 0;      gettimeofday(&amp;cur_time, &amp;tz);     memset(tran_ctP, '0', sizeof(*tran_ctP));     for (i=0; i&lt;info_list_len; i++) {         struct timeval *client_timeP;         int time_diff;         thread_descr_t *descrP;         int delta;         if (info_list[i] == NULL    !info_list[i]-&gt;initialized) {             continue;         }         if (!info_list[i]-&gt;done) num_active++;         descrP = &amp;info_list[i]-&gt;descr;         delta = THREAD_STATE_REPORT_DELTA(descrP-&gt;state);         client_timeP = client_last_time(descrP);          for (j=1; j&lt;=MAX_TRAN_TYPE; j++) {             tran_ctP-&gt;tran[j].num += info_list[i]-&gt;tran[j].num;             tran_ctP-&gt;tran[j].errs += info_list[i]-&gt;tran[j].errs;             tran_ctP-&gt;tran[j].RTcount += info_list[i]-&gt;tran[j].RTcount;             tran_ctP-&gt;tran[j].RTtotal[0] += info_list[i]-&gt;tran[j].RTtotal[0];             tran_ctP-&gt;tran[j].RTtotal[1] += info_list[i]-&gt;tran[j].RTtotal[1];             tran_ctP-&gt;errors += info_list[i]-&gt;tran[j].errs;         }          time_diff = cur_time.tv_sec - client_timeP-&gt;tv_sec;         DPRINT(("bg_thread: thread %d (index %d) state %s tran %d for %d sec\n",             info_list[i]-&gt;thread_id, i,             thread_state_to_str(descrP-&gt;state),             descrP-&gt;tran,             time_diff));         if (descrP-&gt;state == thread_state_init) {             num_init++;         } else if (descrP-&gt;state == thread_state_done) {             num_done++;         } else if (time_diff &gt; delta) {             num_per_state[descrP-&gt;state]++;             total_stuck++;             if (!descrP-&gt;printed) {                 err_printf("bg_thread: thread %d (index %d) state %s tran %d stuck for %d                 sec\n",                     info_list[i]-&gt;thread_id, i,                     thread_state_to_str(descrP-&gt;state),                     descrP-&gt;tran,                     time_diff);                 descrP-&gt;printed = 1;             }         } else if (descrP-&gt;printed) {             err_printf("bg_thread: thread %d (index %d) state %s tran %d unstuck.\n",                 info_list[i]-&gt;thread_id, i,                 thread_state_to_str(descrP-&gt;state),                 descrP-&gt;tran);             descrP-&gt;printed = 0;         }     } }  if (num_threadsP)     *num_threadsP = num_active; if (num_threadsInitP)     *num_threadsInitP = num_init;  if (num_init &gt; 0) {     err_printf("bg_thread: %d threads still in the init state\n",         num_init); } else if (!init_printed) {     err_printf("bg_thread: All %d threads are running\n",         info_list_len);     init_printed = 1; }  if (num_active != info_list_len)     err_printf("%d threads of %d are still active\n",         num_active, info_list_len);  if (num_done &gt; 0) {     err_printf("bg_thread: %d threads done so far.\n", num_done); }  if (total_stuck &gt; 0) {     err_printf("bg_thread: Summary %d stuck: ", total_stuck);     for (i=0; i&lt;NUM_STATES; i++) {         if (num_per_state[i] &gt; 0) {             fprintf(ERROUT, "%d %s, ",                 num_per_state[i], thread_state_to_str(i));         }     } } </pre>	<pre>         }         fprintf(ERROUT, "\n");     }     total_tran_err = 0;     for (i=0; i&lt;=MAX_TRAN_TYPE; i++)         total_tran_err += tran_ctP-&gt;tran[i].errs;     if (total_tran_err &gt; 0) {         err_printf("bg_thread: %d errs: %d no, %d pa, %d os, %d sl\n",             total_tran_err,             tran_ctP-&gt;tran[NEWO_TRANS].errs,             tran_ctP-&gt;tran[PAYMENT_TRANS].errs,             tran_ctP-&gt;tran[ORDER_STAT_TRANS].errs,             tran_ctP-&gt;tran[STOCK_TRANS].errs);     } } } MUTEX_UNLOCK(&amp;init_lock); }  #else static void check_threads(total_tran_count_t *tran_ctP, int *num_threadsP, int *num_threadsInitP) {     int i;     extern total_tran_count_t *pClientInfo;      if (num_threadsP != NULL)         *num_threadsP = 0;     if (num_threadsInitP != NULL)         *num_threadsInitP = 0;      memcpy(tran_ctP, pClientInfo, sizeof(total_tran_count_t));      /* report error info */     if (pClientInfo-&gt;errors &gt; 0) {         err_printf("bg_thread: %d errs: %d no, %d pa, %d os, %d sl\n",             pClientInfo-&gt;errors,             pClientInfo-&gt;tran[NEWO_TRANS].errs,             pClientInfo-&gt;tran[PAYMENT_TRANS].errs,             pClientInfo-&gt;tran[ORDER_STAT_TRANS].errs,             pClientInfo-&gt;tran[STOCK_TRANS].errs);     } }  #endif  void start_bg_debug_thread() {     int rc;     pthread_attr_t attr;     pthread_t thread;      if (rc = pthread_attr_create(&amp;attr)) {         err_printf("start_bg_debug_thread: pthread_attr_create failed: %d\n", rc);         return;     }     if ((rc = pthread_create(&amp;thread,         attr,         bg_thread,         (pthread_addr_t) NULL)) != 0) {         err_printf("start_bg_debug_thread: pthread_create failed: %d\n", rc);         return;     }     if (rc = pthread_detach(&amp;thread) != 0) {         err_printf("start_bg_debug_thread: pthread_detach failed %d\n", rc);         return;     } }  /* client_status_report:  * mainly copied from bg_thread  */ void *client_status_report(int fileno) {     static struct timespec time_wait = {60, 0};      total_tran_count_t tran_ct;     tran_info_t *curP;     struct timeval cur_time;     struct timezone tz;     char buf[1024], cmd='\0';     int i, cnt=0;      /* a loop for communication with tpcc_monitor */     while (cmd != 'q') {         struct timeval cur_time;         struct timeval user_time={0,0}, sys_time={0,0};         struct timezone tz;         int num_threads, num_threadsInit;          memset(&amp;tran_ct, 0, sizeof(tran_ct));          /* read next cmd from the socket */         read(fileno, buf, 1);         cmd = buf[0];         /* DPRINT(("%"c\n",cmd)); */         if (cmd=='q') {             break;         }     } } </pre>
---	---

```

    }

    check_threads(&tran_ct, &num_threads, &num_threadsInit);
    gettimeofday(&cur_time, &tz);
#ifdef GET_CLIENT_USAGE
    getUserSysTime(&user_time, &sys_time);
#endif

    /* The tpcc_monitor has to read the data in the same order
    * to get the correct data.
    */
    prefix_sprintf(buf, "n");
    write(fileno, buf, strlen(buf));
    sprintf(buf, "%d %d %d %d %d\n",
            cur_time.tv_sec, cur_time.tv_usec,
            tran_ct.tran[NEWO_TRANS].num-tran_ct.tran[NEWO_TRANS].errs,
            tran_ct.errors, num_threads, num_threadsInit);
    write(fileno, buf, strlen(buf));
    sprintf(buf, "%d %d %d %d\n", user_time.tv_sec, user_time.tv_usec,
            sys_time.tv_sec, sys_time.tv_usec);
    write(fileno, buf, strlen(buf));
    for (i=0, curP=tran_ct.tran; i<=MAX_TRAN_TYPE; i++, curP++) {
        if (i==0) continue;
        sprintf(buf, "%d %d %d %d %d %d %d %d\n",
                i, curP->RTtotal, curP->errs, curP->RTtotal[0],
                curP->RTtotal[1]);
        write(fileno, buf, strlen(buf));
    }
    write(fileno, ENDMSG, strlen(ENDMSG));
}

/* for AIX only */
void getUserSysTime(struct timeval *user_time, struct timeval *sys_time)
{
#ifdef defined(AIX)
    struct rusage rubuff;

    if (getrusage(RUSAGE_SELF, &rubuff) == 0) {
        user_time->tv_sec = rubuff.ru_utime.tv_sec;
        user_time->tv_usec = rubuff.ru_utime.tv_usec;

        sys_time->tv_sec = rubuff.ru_stime.tv_sec;
        sys_time->tv_usec = rubuff.ru_stime.tv_usec;
    } else {
        user_time->tv_sec = user_time->tv_sec = 0;
        sys_time->tv_sec = sys_time->tv_sec = 0;
    }
#endif
#ifdef defined(solaris)
    /* WARNING: not test it yet */
    struct tms t;
    static long ticks = 0;
    register long n;

    if (ticks == 0) ticks = sysconf(_SC_CLK_TCK);

    (void)times(&t);

    user_time->tv_sec = t.tms_utime / ticks;
    user_time->tv_usec = (t.tms_utime % ticks) * 1000000 / ticks;

    sys_time->tv_sec = t.tms_stime / ticks;
    sys_time->tv_usec = (t.tms_stime % ticks) * 1000000 / ticks;
#else
    user_time->tv_sec = 0;
    user_time->tv_usec = 1000;

    sys_time->tv_sec = 0;
    sys_time->tv_usec = 1000;
#endif
}

#ifdef WEB_TPCC_CLIENT
extern void getStatsForm(char* outBufP, char* rawStatsP, int interval);

int web_client_status(char* szFormP, int cmd, int interval)
{
    total_tran_count_t tran_ct;
    tran_info_t *curP;
    struct timeval cur_time;
    struct timezone tz;
    int i, cnt=0;
    int num_threads, num_threadsInit;
    char *statusP;
    char tempP[512];

    if (cmd==1)
        statusP = szFormP;
    else
        statusP = tempP;

    *statusP = '\0';
    memset(&tran_ct, 0, sizeof(tran_ct));

    if (cmd == 2) /* quit */
        return 1;
}

```

```

check_threads(&tran_ct, &num_threads, &num_threadsInit);
gettimeofday(&cur_time, &tz);

prefix_sprintf(statusP, "n");
sprintf(statusP+strlen(statusP), "%d %d %d %d %d\n", cur_time.tv_sec,
        cur_time.tv_usec,
        tran_ct.tran[NEWO_TRANS].num-tran_ct.tran[NEWO_TRANS].errs,
        tran_ct.errors,
        num_threads, num_threadsInit);
sprintf(statusP+strlen(statusP), "0 0 0 0\n");
for (i=0, curP=tran_ct.tran; i<=MAX_TRAN_TYPE; i++, curP++) {
    if (i==0) continue;
    sprintf(statusP+strlen(statusP), "%d %d %d %d %d %d %d %d\n",
            i, curP->RTcount, curP->errs, curP->RTtotal[0],
            curP->RTtotal[1]);
}

if (cmd != 1)
    getStatsForm(szFormP, statusP, interval);
return 0;
}
#endif

```

## client\_listen.c

```

/*
 * client_listen.c
 *
 * $Revision: 1.8 $
 * $Date: 1999/05/06 21:28:26 $
 * $Lo: $
 *
 *
 * $TALog: client_listen.c,v $
 * Revision 1.8 1999/05/06 21:28:26 oz
 * - Removed all the .. from the includes
 * - Added -I.. to the makefiles instead
 * - Moved all the thread related code and connection
 * selection to serverMon.c
 * [from r1.7 by delta oz-24309-TPCC-add-oracle8.1-code, r1.5]
 *
 * Revision 1.7 1999/01/29 20:16:32 wenjian
 * Change logprintf to err_printf
 * [from r1.6 by delta wenjian-23787-TPCC-integrate-code-for-AIX-and-NT, r1.7]
 *
 * Revision 1.6 1998/11/09 16:59:36 wenjian
 * In this revision, most of the changes are related to the directory of header
 * files after directory reorganization. Other changes include adding or removing
 * files to put them in the right directories. Makefiles are written for NT
 * platform so that nmake is working on NT now. Need a top level Makefile for all
 * the directories.
 * [from r1.5 by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.2]
 *
 * Revision 1.5 1998/11/09 14:48:14 wenjian
 * In an effort to make a new directory structure for TPCC, this delta
 * creates two directories: tpcc/client and tpcc/server. All the files
 * for this revision are copied from tpcc/sp-tpcc without any change.
 * Further change may be needed for some files due to the change of
 * the directory structure.
 * [added by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.1]
 *
 * Revision 1.30 1998/09/26 10:56:25 oz
 * - renamed thread_init and thread_done to clnt_thread_init and
 * clnt_thread_done respectively because of name conflicts on AIX.4.3
 * [from r1.22 by delta oz-23339-TPCC-update-for-NT, r1.2]
 *
 * Revision 1.22 1998/08/18 14:38:38 wenjian
 * Minor change
 * [from r1.18 by delta wenjian-21750-TPCC-changes-for-porting-on-NT, r1.4]
 *
 * Revision 1.18 1998/04/29 19:47:41 wenjian
 * - Use fd instead of stream on NT
 * - Add code to consider tpcc_monitor as a special client login
 * - Use TRY and CATCH_ALL to deal with exceptions
 * [from r1.17 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients, r1.1]
 *
 * Revision 1.17 1998/02/17 22:13:28 wenjian
 * [merge of changes from 1.14 to 1.15 into 1.16]
 *
 * Revision 1.15 1998/02/17 16:04:41 oz
 * - Split the login into two parts to allow for special logins
 * - If the warehouse ID is 0, this is a special login to
 * query the client for status
 *
 * - First, login
 * If the w_id is bigger than 0: normal thread.
 * Otherwise, call client_report.
 * [from r1.14 by delta oz-21864-TPCC-split-client-login-screen, r1.1]
 *
 * Revision 1.16 1998/02/17 22:06:59 wenjian
 * Add head files and define macros for win32
 * [from r1.14 by delta wenjian-21750-TPCC-changes-for-porting-on-NT, r1.1]
 *
 * Revision 1.14 1998/01/28 22:24:48 oz
 * [from r1.13 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.4]
 *
 * Revision 1.13 1998/01/26 16:19:22 oz

```

```

* - moved all the code pertaining to the background
* thread to its own file and all the data structures
* to client_utils.h
* [from r1.12 by delta oz-21689-TPCC-move-client-bg-thread-to-separate-file, r1.1]
*
* Revision 1.12 1998/01/24 14:17:04 oz
* - User server name to identify server and name delivery file
* - Use env variable HOME instead of /home/encina if HOME is set
*
* - Print the thread ID on thread exit as well
* [from r1.11 by delta oz-21687-TPCC-use-server-name-to-identify-process, r1.1]
*
* Revision 1.11 1998/01/23 15:07:44 oz
* - Updated the SP TPCC directory to the latest files used
* during the SP tpcc audit.
* [from r1.10 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
*
*
* Exported functions:
*     make_connections
*
* Private functions:
*     process_terminal
*
*/

client_listen.c
Code in the client that listens for requests from the
terminal processes and submits them for processing.

There is one listening function: make_connection.
That function calls cnm_ManageConnection which never returns
and so it is best to call it in its own independent thread.

As soon as cnm_ManageConnections receives a connection it
starts a new thread and calls process_terminal in that
thread passing in the file descriptor for the new connection.

Note that the client does not need to know in advance how many
terminals it will talk to.

The function process_terminal reads initializes the thread
and then calls client_init to process all the requests from
that terminal.
*/

#include <stdlib.h>
#ifdef WIN32
#include <io.h>
#else
#include <stdio.h>
#include <sys/types.h>
#include <tc.h>
#include "common/do_tpcc.h"
#include "common/tpcc_type.h"

#ifdef solaris
#include <dce/pthread.h>
#else /* solaris */
#include <pthread.h>
#endif /* solaris */

#include "client_utils.h"
#ifdef WIN32
#include "cnm.h"
#else
#include <cnm/cnm.h>
#endif

#ifdef WIN32
#define close _close
#define fileno _fileno
#endif
*/
* State about the terminal stored by the terminal thread
* work_entry: The work entry to be used by this terminal thread.
*/
typedef struct {
    int profiling;
    int terminal_id;
    void *handle_contextP;
} terminal_context_t;

/**
** Function Prototypes
**/
static void process_terminal(cnm_arg_t *argP);

extern void client_init(int, int, int, void *);
extern void client_login(int, int, int *, int *);

*
* process_terminal
*
* The argument we get is a file descriptor for a terminal
* process. We read from that file to receive input and send
* output back to that file.

```

```

*/
static void process_terminal(cnm_arg_t *argP)
{
    int w_id, d_id;
    terminal_context_t terminal_context;
    tpcc_data_t tran_data;
    int fdIn;
    pthread_t thread = pthread_self();
    int thread_id = pthread_getunique_np(&thread);
    struct timespec rand_sleep;
#ifdef _AIX
    tid_t tid = thread_self();
#else
    int tid = thread_id;
#endif

#ifdef WIN32
    fdIn = argP->fd;
#else
    fdIn = fileno(argP->stream);
#endif /* WIN32 */

/*
* Default terminal context
* This may be updated later by the terminal
*/
terminal_context.terminal_id = -1;
terminal_context.profiling = 0;

TRY {
    client_login(fdIn, fdIn, &w_id, &d_id);
    if (w_id > 0) {
        /* Initialize the server handle and other thread structures */
        terminal_context.handle_contextP = (void *)clnt_thread_init();

        err_printf("Tid: %d (0x%x) w_id %d, d_id %d\n", tid, tid, w_id, d_id);
        client_init(fdIn, fdIn, w_id, d_id, terminal_context.handle_contextP);
        err_printf("Thread done - Tid %d (0x%x)\n", tid, tid);
        clnt_thread_done(terminal_context.handle_contextP);
    } else {
        err_printf("Starting Auxiliary Thread, Tid %d (0x%x)\n", tid, tid);
        client_status_report(fdIn);
        err_printf("End of Auxiliary Thread, Tid %d (0x%x)\n", tid, tid);
    }
} CATCH_ALL {
    err_printf("An exception happened\n");
    logprintf("End of Auxiliary Thread, Tid %d (0x%x)\n", tid, tid);
}
ENDTRY

close(fdIn);
}

/*
* make_connections
*
* Listen for connections on a socket.
* Whenever a connection is made, start a thread to talk
* to the terminal.
* This functions is spawned on its own thread.
*/
void make_connections(argP)
void *argP;
{
    int port = (int)argP;
    char port_descr[28];
    int rc;

    DPRINT(("Using socket %d\n", port));
    err_printf("Using thread stack size default\n");
    sprintf(port_descr, "ncacn_ip_tcp[%d]", port);
    rc = cnm_ManageConnections(port_descr,

                                (cnm_userRoutine_t)process_terminal,
                                NULL,
                                0, /* Max Connections */
                                1); /* Spawn threads */

    err_printf("cnm_ManageConnections returned %d\n", rc);
}

client_listen.h

/*
* client_listen.h
*
* $Revision: 1.5 $
* $Date: 1998/11/09 14:48:14 $
* $Log: $
*
* $TALog: client_listen.h,v $
* Revision 1.5 1998/11/09 14:48:14 wenjian
* In an effort to make a new directory structure for TPCC, this delta
* creates two directories: tpcc/client and tpcc/server. All the files
* for this revision are copied from tpcc/sp-tpcc without any change.
* Further change may be needed for some files due to the change of
* the directory structure.
* [added by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.1]
*
* Revision 1.1 1997/04/20 11:57:55 oz

```



```

* - This is the code base modified at IBM Poughkeepsie
* by Ofer Zajicek and Radha Sivaramakrishnan for the
* SP scaling test for TPCC.
* [added by delta oz-19782-TPCC-add-ibm-sp-code, r1.1]
*
* Revision 1.1 1995/07/09 18:12:10 oz
* - Modified the client side of the TPCC benchmark to have multithreaded
* clients. There is a terminal process for each terminal -- when
* not using the terminal emulator each terminal process emulates one
* terminal. The terminal processes communication with the client
* process using a unix socket.
*
* On the client side there is a thread for each terminal process.
* That thread receives the request from the terminal and puts it on
* a queue. There is one processing thread that dequeues the requests
* and sends them to the server for processing.
* [added by delta oz-15875-TPCC-reduce-the-number-of-clients, r1.1]
*
*
*
*/
* client_listen.h
*/

#ifndef TPCC_CLIENT_LISTEN_H
#define TPCC_CLIENT_LISTEN_H

void make_connections(void *argP);

#endif /* TPCC_CLIENT_LISTEN_H */

                client_main.c

#include "string.h"
#include "tpcc.h"

extern void client_init(int infd, int outfd, int w_id, int d_id, void *conP);
extern void client_login(int infd, int outfd, int *w_idP, int *d_idP);

main()
{
    int w_id, d_id;
    client_login(0, 1, &w_id, &d_id);
    client_init(0, 1, w_id, d_id, (void *)0);
}

int send_new_order(void *contextP, NewOrder_data *data) {
    int i;

    data->s_W_ID = 11;
    data->s_D_ID = 22;
    data->s_C_ID = 3333;
    strcpy((char *)data->s_C_LAST, "1234567890123456");
    strcpy((char *)data->s_C_CREDIT, "BC");
    data->s_C_DISCOUNT = 0.1556;
    data->s_O_ID = 4444;
    strcpy((char *)data->s_O_ENTRY_D, "1992-10-2 12:33:11");
    strcpy((char *)data->s_status_line, "123456789012345678901234");
    data->s_total_amount = 12.98;
    data->s_transtatus = 0;
    data->s_W_TAX = 0.1234;
    data->s_D_TAX = 0.5678;

    for (i=0; i < data->s_OL_CNT; i++) {
        strcpy((char *)data->item[i].s_I_NAME, "123456789012345678901234");
        data->item[i].s_S_QUANTITY = i + 1;
        data->item[i].s_brand_generic[0] = 'B';
        data->item[i].s_I_PRICE = i + 1;
        data->item[i].s_OL_AMOUNT = i + 1;
    }
    return 0;
}

int send_payment(void *contextP, Payment_data *data) {
    data->s_W_ID = 11;
    data->s_D_ID = 22;
    data->s_C_ID = 3333;
    data->s_C_W_ID = 44;
    data->s_C_D_ID = 55;
    data->s_H_AMOUNT = 9.55;
    strcpy((char *)data->s_W_STREET_1, "12345678901234567890");
    strcpy((char *)data->s_W_STREET_2, "12345678901234567890");
    strcpy((char *)data->s_W_CITY, "12345678901234567890");
    strcpy((char *)data->s_W_STATE, "PR");
    strcpy((char *)data->s_W_ZIP, "123456789");
    strcpy((char *)data->s_D_STREET_1, "12345678901234567890");
    strcpy((char *)data->s_D_STREET_2, "12345678901234567890");
    strcpy((char *)data->s_D_CITY, "12345678901234567890");
    strcpy((char *)data->s_D_STATE, "PR");
    strcpy((char *)data->s_D_ZIP, "123456789");
    strcpy((char *)data->s_C_FIRST, "1234567890123456");
    strcpy((char *)data->s_C_MIDDLE, "12");
    strcpy((char *)data->s_C_LAST, "1234567890123456");
    strcpy((char *)data->s_C_STREET_1, "12345678901234567890");
    strcpy((char *)data->s_C_STREET_2, "12345678901234567890");
}

```

```

strcpy((char *)data->s_C_CITY, "12345678901234567890");
strcpy((char *)data->s_C_STATE, "PR");
strcpy((char *)data->s_C_ZIP, "123456789");
strcpy((char *)data->s_C_PHONE, "1234567890123456");
strcpy((char *)data->s_C_SINCE, "1992-23-22 21:11:11");
strcpy((char *)data->s_H_DATE, "1992-10-2 12:33:11");
strcpy((char *)data->s_C_CREDIT, "BC");
data->s_C_CREDIT_LIM = 5000;
data->s_C_DISCOUNT = 0.10;
data->s_C_BALANCE = 122.10;
strcpy((char *)data->s_C_DATA,
"1234567890123456789012345678901234567890123456789012345678901234567890");
return 0;
}

int send_order_status(void *contextP, OrderStatus_data *data) {
    int i;

    data->s_W_ID = 11;
    data->s_D_ID = 22;
    data->s_C_ID = 3333;
    strcpy((char *)data->s_C_FIRST, "1234567890123456");
    strcpy((char *)data->s_C_MIDDLE, "12");
    strcpy((char *)data->s_C_LAST, "1234567890123456");
    data->s_C_BALANCE = 122.10;
    data->s_O_ID = 44;
    strcpy((char *)data->s_O_ENTRY_D, "1992-10-2 12:33:11");
    data->s_O_CARRIER_ID = 55;
    data->s_ol_cnt = 10;

    for (i=0; i < data->s_ol_cnt; i++) {
        data->item[i].s_OL_SUPPLY_W_ID = i + 1;
        data->item[i].s_OL_I_ID = i + 1;
        data->item[i].s_OL_QUANTITY = i + 1;
        data->item[i].s_OL_AMOUNT = i + 1;
        strcpy((char *)data->item[i].s_OL_DELIVERY_D, "1992-10-2 12:33:11");
    }
    return 0;
}

int send_delivery(void *contextP, Delivery_data *data) {
    strcpy((char *)data->s_exec_status, "Delivery has been queued");
    return 0;
}

int send_stock_level(void *contextP, StockLevel_data *data) {
    data->s_low_stock = 22;
    return 0;
}

```

## client\_utils.c

```

/*
 *
 * client_utils.c
 *
 * $Revision: 1.9 $
 * $Date: 1999/05/06 21:28:26 $
 * $Log: $
 *
 *
 *
 * $TALog: client_utils.c,v $
 * Revision 1.9 1999/05/06 21:28:26 oz
 * - Removed all the .. from the includes
 * - Added -I. to the makefiles instead
 * - Moved all the thread related code and connection
 * selection to serverMon.c
 * [from r1.7 by delta oz-24309-TPCC-add-oracle8.1-code, r1.5]
 *
 *
 * Revision 1.7 1998/12/11 16:37:57 wenjian
 * Move some common functions from client/client_utils.c to common/tpcc_utils.c.
 * In this version, we only move time_diff_ms(). Need some work in order to
 * move other functions like ERROROUT.
 *
 *
 * - Move time_diff_ms() to common/tpcc_utils.c
 * [from r1.6 by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.2]
 *
 *
 * Revision 1.6 1998/11/09 16:59:36 wenjian
 * In this revision, most of the changes are related to the directory of header
 * files after directory reorganization. Other changes include adding or removing
 * files to put them in the right directories. Makefiles are written for NT
 * platform so that nmake is working on NT now. Need a top level Makefile for all
 * the directories.
 * [from r1.5 by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.2]
 *
 *
 * Revision 1.5 1998/11/09 14:48:14 wenjian
 * In an effort to make a new directory structure for TPCC, this delta
 * creates two directories: tpcc/client and tpcc/server. All the files
 * for this revision are copied from tpcc/sp-tpcc without any change.
 * Further change may be needed for some files due to the change of
 * the directory structure.
 * [added by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.1]
 *
 *
 * Revision 1.9 1998/10/08 14:18:00 dongfeng
 * Add codes for doing web-based tpcc.

```

```

* [from r1.7 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.1]
*
* Revision 1.7 1998/04/29 19:47:42 wenjian
* - Add prefix_sprintf
* - Remove ENCINA_C_CALLING_CONVENTION from err_printf
* [from r1.6 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients, r1.1]
*
* Revision 1.6 1998/02/17 22:07:00 wenjian
* Minor changes for NT
* [from r1.5 by delta wenjian-21750-TPCC-changes-for-porting-on-NT, r1.1]
*
* Revision 1.5 1998/01/24 14:17:04 oz
* - User server name to identify server and name delivery file
* - Use env variable HOME instead of /home/encina if HOME is set
*
* - Flush the logfile after each write
* [from r1.4 by delta oz-21687-TPCC-use-server-name-to-identify-process, r1.1]
*
* Revision 1.4 1998/01/23 15:07:46 oz
* - Updated the SP TPCC directory to the latest files used
* - during the SP tpcc audit.
* [from r1.3 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
*
*
* client_utils.c
* Generic utilities used by the client processes
*/

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

#if defined (solaris)
#include <dce/pthread.h>
#else /* solaris */
#include <pthread.h>
#endif

#include "common/databuf.h"
#include "client_utils.h"
#include "common/do_tpcc.h"
#include "common/tpcc_type.h"

#define CASE(a) case a: retVal = #a; break

int print_thread_id = 1;
extern int user_id;
extern char *user_code;
/*
 * Translate the tpcc return code to a string value
 */
static char *TpccRcToStr(rc)
tpcc_rc_t rc;
{
    char *retVal;
    switch (rc) {
        CASE(INVALID_NEWO);
        CASE(INVALID_HANDLE);
        CASE(SQL_ERROR);
        CASE(TRPC_ERROR);
        CASE(DCE_ERROR);
        CASE(NO_SUCH_LAST_NAME);
        CASE(INVALID_TRAN_TYPE);
        CASE(TPCC_ERROR_BEGIN_NEWO);
        CASE(TPCC_ERROR_DECL_NEWO_SEL_ITEM);
        CASE(TPCC_ERROR_OPEN_NEWO_SEL_ITEM);
        CASE(TPCC_ERROR_OPEN_DIST_NEWO_SEL_ITEM);
        CASE(TPCC_ERROR_FETCH_NEWO_SEL_ITEM);
        CASE(TPCC_ERROR_FETCH_DIST_NEWO_SEL_ITEM);
        CASE(TPCC_ERROR_PREP_NEWO_SEL_STCK);
        CASE(TPCC_ERROR_DECL_NEWO_SEL_STCK);
        CASE(TPCC_ERROR_OPEN_NEWO_SEL_STCK);
        CASE(TPCC_ERROR_OPEN_DIST_NEWO_SEL_STCK);
        CASE(TPCC_ERROR_FETCH_NEWO_SEL_STCK);
        CASE(TPCC_ERROR_FETCH_DIST_NEWO_SEL_STCK);
        CASE(TPCC_ERROR_NEWO_SELECT);
        CASE(TPCC_ERROR_NEWO_UPD_STCK);
        CASE(TPCC_ERROR_DIST_NEWO_UPD_STCK);
        CASE(TPCC_ERROR_NEWO_SELECT_2);
        CASE(TPCC_ERROR_DECL_NEWO_SEL_CUST);
        CASE(TPCC_ERROR_OPEN_NEWO_SEL_CUST);
        CASE(TPCC_ERROR_OPEN_DIST_NEWO_SEL_CUST);
        CASE(TPCC_ERROR_FETCH_NEWO_SEL_CUST);
        CASE(TPCC_ERROR_FETCH_DIST_NEWO_SEL_CUST);
        CASE(TPCC_ERROR_DECL_NEWO_SEL_DIST);
        CASE(TPCC_ERROR_OPEN_NEWO_SEL_DIST);
        CASE(TPCC_ERROR_OPEN_DIST_NEWO_SEL_DIST);
        CASE(TPCC_ERROR_FETCH_NEWO_SEL_DIST);
        CASE(TPCC_ERROR_FETCH_DIST_NEWO_SEL_DIST);
        CASE(TPCC_ERROR_PREP_NEWO_INS_OL);
        CASE(TPCC_ERROR_DECL_NEWO_INS_OL);
        CASE(TPCC_ERROR_OPEN_NEWO_INS_OL);
        CASE(TPCC_ERROR_OPEN_DIST_NEWO_INS_OL);
        CASE(TPCC_ERROR_PUT_NEWO_INS_OL);
        CASE(TPCC_ERROR_PUT_DIST_NEWO_INS_OL);
        CASE(TPCC_ERROR_DECL_NEWO_SEL_WARE);
        CASE(TPCC_ERROR_OPEN_NEWO_SEL_WARE);
    }
}

```

```

CASE(TPCC_ERROR_OPEN_DIST_NEWO_SEL_WARE);
CASE(TPCC_ERROR_FETCH_NEWO_SEL_WARE);
CASE(TPCC_ERROR_FETCH_DIST_NEWO_SEL_WARE);
CASE(TPCC_ERROR_EXECUTE_NEWO_UPD_INS);
CASE(TPCC_ERROR_UPDATE_NEWO_NEXT_OID);
CASE(TPCC_ERROR_PREP_NEWO_INS);
CASE(TPCC_ERROR_EXECUTE_DIST_NEWO_INS);
CASE(TPCC_ERROR_EXECUTE_NEWO_COMMIT);
CASE(TPCC_ERROR_ROLLBACK_NEWO);
CASE(TPCC_ERROR_REMOTE_OL_SELECT);
CASE(TPCC_ERROR_REMOTE_OL_UPDATE);
CASE(TPCC_ERROR_OPEN_ORDS_CNT_CID);
CASE(TPCC_ERROR_FETCH_ORDS_CNT_CID);
CASE(TPCC_ERROR_OPEN_ORDS_SEL_CLAST);
CASE(TPCC_ERROR_FETCH_ORDS_SEL_CLAST);
CASE(TPCC_ERROR_OPEN_ORDS_SEL_CID);
CASE(TPCC_ERROR_FETCH_ORDS_SEL_CID);
CASE(TPCC_ERROR_OPEN_ORDS_SEL_OLDORD);
CASE(TPCC_ERROR_FETCH_ORDS_OLDORD);
CASE(TPCC_ERROR_OPEN_ORDS_SEL_OL);
CASE(TPCC_ERROR_FETCH_ORDS_SEL_OL);
CASE(TPCC_ERROR_EXECUTE_ORDS_COMMIT);
CASE(TPCC_ERROR_OPEN_DELIVERY_OLDEST_OID);
CASE(TPCC_ERROR_FETCH_DELIVERY_OLDEST_OID);
CASE(TPCC_ERROR_EXECUTE_DELIVERY_COMMIT);
CASE(TPCC_ERROR_OPEN_DELIVERY_SEL_ORD);
CASE(TPCC_ERROR_FETCH_DELIVERY_SEL_ORD);
CASE(TPCC_ERROR_OPEN_DELIVERY_SEL_SUM_OL);
CASE(TPCC_ERROR_FETCH_DELIVERY_SEL_SUM_OL);
CASE(TPCC_ERROR_EXECUTE_DELIVERY_EXEC_DVRY);
CASE(TPCC_ERROR_SELECT_DELIVERY_ORDER_ID);
CASE(TPCC_ERROR_SELECT_DELIVERY_CARRIER_ID);
CASE(TPCC_ERROR_SELECT_DELIVERY_BALANCE);
CASE(TPCC_ERROR_OPEN_STOCKLEVEL_SEL_OID);
CASE(TPCC_ERROR_FETCH_STOCKLEVEL_SEL_OID);
CASE(TPCC_ERROR_OPEN_STOCKLEVEL_CNT_SID);
CASE(TPCC_ERROR_FETCH_STOCKLEVEL_CNT_SID);
CASE(TPCC_ERROR_OPEN_STOCKLEVEL_FIND);
CASE(TPCC_ERROR_FETCH_STOCKLEVEL_FIND);
CASE(TPCC_ERROR_EXECUTE_STOCKLEVEL_COMMIT);
CASE(TPCC_ERROR_OPEN_PAYMENT_CNT_CID);
CASE(TPCC_ERROR_FETCH_PAYMENT_CNT_CID);
CASE(TPCC_ERROR_OPEN_PAYMENT_SEL_CLAST);
CASE(TPCC_ERROR_FETCH_PAYMENT_SEL_CLAST);
CASE(TPCC_ERROR_OPEN_PAYMENT_SEL_CID);
CASE(TPCC_ERROR_FETCH_PAYMENT_SEL_CID);
CASE(TPCC_ERROR_DECL_PAYMENT_SEL_DIST);
CASE(TPCC_ERROR_OPEN_PAYMENT_SEL_DIST);
CASE(TPCC_ERROR_ERROR_DIST_PAYMENT_SEL_DIST);
CASE(TPCC_ERROR_FETCH_PAYMENT_SEL_DIST);
CASE(TPCC_ERROR_FETCH_DIST_PAYMENT_SEL_DIST);
CASE(TPCC_ERROR_DECL_PAYMENT_SEL_WARE);
CASE(TPCC_ERROR_OPEN_PAYMENT_SEL_WARE);
CASE(TPCC_ERROR_OPEN_DIST_PAYMENT_SEL_WARE);
CASE(TPCC_ERROR_FETCH_PAYMENT_SEL_WARE);
CASE(TPCC_ERROR_FETCH_DIST_PAYMENT_SEL_WARE);
CASE(TPCC_ERROR_EXECUTE_PAYMENT_UPD_CUST_LAST);
CASE(TPCC_ERROR_EXECUTE_PAYMENT_UPD_CUST_ID);
CASE(TPCC_ERROR_COMMIT_PAYMENT_UPD_CUST);
CASE(TPCC_ERROR_SELECT_PAYMENT_W_YTD);
CASE(TPCC_ERROR_SELECT_PAYMENT_D_YTD);
CASE(TPCC_ERROR_BEGIN_PAYMENT);
CASE(TPCC_ERROR_EXECUTE_PAYMENT_COMMIT);

    default: retVal = "-Unknown-"; break;
}
return(retVal);
}

/*
 * get_thread_id
 * A function that returns the thread ID of the current thread
 */
int get_thread_id()
{
    #ifdef WEB_TPCC_CLIENT
        return(GetCurrentThreadId());
    #else
        pthread_t thread = pthread_self();
        int thread_id = pthread_getunique_np(&thread);
        return(thread_id);
    #endif
}

#define A_CASE(a,b) case a: retVal = b; break
/*
 * Translate the transaction code to its name - for formatting
 */
char *clientUtils_TransCodeToName(type)
int type;
{
    char *retVal = "-Unknown-";
    switch (type) {
        A_CASE(NEWO_TRANS, "NEWOR");
        A_CASE(PAYMENT_TRANS, "PAYMN");
        A_CASE(ORDER_STAT_TRANS, "ORDER");
        A_CASE(DELIVERY_TRANS, "DELIV");
        A_CASE(STOCK_TRANS, "STOCK");
    }
}
return(retVal);

```

```

*)
*)
*) Print the return status of a TPCC transaction
*) and the corresponding SQL codes and ISAM codes
*/
void clientUtils_ReportReturn(msg, statusP)
char *msg;
data_header *statusP;
{
switch (statusP->returncode) {
case SUCCESS_CODE:
err_printf("After %s, rc = %d\n", msg, statusP->returncode);
break;
case SQL_ERROR:
err_printf("ERROR: After %s, rc = SQL_ERROR, SQL=%d, ISAM=%d\n", msg,
statusP->sql_code,
statusP->isam_code);
break;
case INVALID_NEWO:
err_printf("After %s, rc = INVALID_NEWO\n", msg);
break;
case DCE_ERROR:
err_printf("ERROR: After %s, rc = DCE_ERROR\n", msg);
break;
case TRPC_ERROR:
err_printf("ERROR: After %s, rc = TRPC_ERROR\n", msg);
break;
case NO_SUCH_LAST_NAME:
err_printf("After %s, rc = NO_SUCH_LAST_NAME.\n", msg);
break;
case DISTRIBUTED_TRAN_FAILED:
err_printf("After %s, rc = DISTRIBUTED_TRAN_FAILED.\n", msg);
break;
default:
err_printf("ERROR: After %s, rc = %s (%d), SQL=%d, ISAM=%d\n", msg,
TpccRcToStr(statusP->returncode),
statusP->returncode,
statusP->sql_code, statusP->isam_code);
break;
}
}

*)
*) clientUtils_SetReturnCode
*)
*) Set the return code in the dataP union.
*) dataP is a pointer to a union of all the transaction types.
*) Each member of the union has a header field that contains
*) a return code. Set the returncode value of the header field
*) for dataP to be code.
*/
void clientUtils_SetReturnCode(dataP, code)
tpcc_data_t *dataP;
tpcc_rc_t code;
{
switch (dataP->tran_type) {
case NEWO_TRANS: {
newOrder_data_t *ptr = &dataP->data.new_order;
ptr->header.returncode = code;
break;
}
case PAYMENT_TRANS: {
payment_data_t *ptr = &dataP->data.payment;
ptr->header.returncode = code;
break;
}
case ORDER_STAT_TRANS: {
orderStatus_data_t *ptr = &dataP->data.order_status;
ptr->header.returncode = code;
break;
}
case DELIVERY_TRANS: {
delivery_data_t *ptr = &dataP->data.delivery;
ptr->header.returncode = code;
break;
}
case STOCK_TRANS: {
stockLevel_data_t *ptr = &dataP->data.stock_level;
ptr->header.returncode = code;
break;
}
}
}

*)
*) get_prefix
*)
*) Format the output prefix for printing:
*) It contains the user_id, 'C' or 'T' depending on whether it
*) is a terminal or a client and optional a thread identifier
*) The prefix is written in the buffer passed in by the caller.
*/
void get_prefix(buffer)
char *buffer;
{
if (print_thread_id) {
int thread_id = get_thread_id();
sprintf(buffer, "%s(%d-%s-%d)%s",
user_id < 10 ? " " : user_id < 100 ? " " : "",
user_id,
user_code,
thread_id,
thread_id < 10 ? " " : "");
} else {
sprintf(buffer, "%s(%d-%s)",
user_id < 10 ? " " : "", user_id, user_code);
}
}

/*
*) err_printf
*) A var-arg function that appends the current time and
*) other data to the print request and sends it to stderr
*) if it is not a web client, to a file if it is
*/
void err_printf(char *format, ...)
{
time_t cur_time;
char time_str[30];
char line_prefix[50];
va_list ap;

va_start(ap, format);

cur_time = time(&cur_time);
strftime(time_str, 29, "%X", localtime(&cur_time));

get_prefix(line_prefix);

fprintf(ERROUT, "%s %s - ", line_prefix, time_str);
vfprintf(ERROUT, format, ap);
#ifdef WEB_TPCC_CLIENT
flush(ERROUT);
#endif
va_end(ap);
}

/*
*) logprintf
*) A var-arg function that prints both to standard error to
*) the log file. It prepends every line with the current time
*) and the user id.
*/
void logprintf(char *format, ...)
{
time_t cur_time;
char time_str[30];
char line_prefix[50];
va_list ap;

va_start(ap, format);

cur_time = time(&cur_time);
strftime(time_str, 29, "%X", localtime(&cur_time));

get_prefix(line_prefix);

fprintf(logtpcc ? logtpcc : ERROUT, "%s %s - ", line_prefix, time_str);
vfprintf(logtpcc ? logtpcc : ERROUT, format, ap);
if (logtpcc)
flush(logtpcc);

if (debug && logtpcc) {
fprintf(ERROUT, "%s %s - ", line_prefix, time_str);
vfprintf(ERROUT, format, ap);
}

va_end(ap);
}

void prefix_sprintf(char *buf, char *format, ...)
{
time_t cur_time;
char time_str[30];
char line_prefix[50];
char info[256];
va_list ap;

va_start(ap, format);

cur_time = time(&cur_time);
strftime(time_str, 29, "%X", localtime(&cur_time));

get_prefix(line_prefix);

sprintf(buf, "%s %s - ", line_prefix, time_str);
vsprintf(info, format, ap);
streat(buf, info);

va_end(ap);
}
}

```

**client\_utils.h**

<pre> * * *      client_utils.h * * \$Revision: 1.11 \$ * \$Date: 1999/05/06 21:28:26 \$ * \$Log:   \$ * * * * STALog: client_utils.h,v \$ * Revision 1.11 1999/05/06 21:28:26 oz * - Removed all the .. from the includes * - Added -I.. to the makefiles instead * - Moved all the thread related code and connection   selection to serverMon.c * [from r1.8 by delta oz-24309-TPCC-add-oracle8.1-code, r1.5] * * Revision 1.8 1998/12/14 20:27:54 wenjian * Made corresponding changes due to data structure change of tran_info_t. * * - Change data structure tran_info_t * [from r1.7 by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.3] * * Revision 1.7 1998/12/11 16:14:19 wenjian * Add code for checking statistic data in a single variable and collecting * statistic data based on iStatsFrequency. * * - Add total_num_trans to tran_info_t. * [from r1.6 by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.1] * * Revision 1.6 1998/11/09 16:59:37 wenjian * In this revision, most of the changes are related to the directory of header * files after directory reorganization. Other changes include adding or removing * files to put them in the right directories. Makefiles are written for NT * platform so that nmake is working on NT now. Need a top level Makefile for all * the directories. * [from r1.5 by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.2] * * Revision 1.5 1998/11/09 14:48:15 wenjian * In an effort to make a new directory structure for TPCC, this delta * creates two directories: tpcc/client and tpcc/server. All the files * for this revision are copied from tpcc/sp-tpcc without any change. * Further change may be needed for some files due to the change of * the directory structure. * [added by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.1] * * Revision 1.4 1998/11/09 14:26:51 wenjian * Change enc_status to a data structure that has fields: * - Status code * - Line Number * - File Name * - Encina Error Code * - Error Msg * Remove statusMsgs in web_tpcc.c * * Add definition of enc_status_t * [from r1.19 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.6] * * Revision 1.19 1998/10/22 21:13:07 wenjian * [merge of changes from 1.11 to 1.18 into 1.14] * * Revision 1.18 1998/10/22 19:18:32 dongfeng * [from r1.17 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.2] * * Revision 1.17 1998/10/08 14:18:00 dongfeng * Add codes for doing web-based tpcc. * [from r1.11 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.1] * * Revision 1.14 1998/08/18 14:38:39 wenjian * Minor change * [from r1.13 by delta wenjian-21750-TPCC-changes-for-porting-on-NT, r1.4] * * Revision 1.13 1998/08/18 13:35:42 wenjian * Remove NUM_NEXT_REPORTS since it is no use. * [from r1.11 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients, r1.7] * * Revision 1.11 1998/06/17 15:28:51 wenjian * Add 'double time' in struct total_tran_count_t. * [from r1.10 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients, r1.2] * * Revision 1.10 1998/04/29 19:47:43 wenjian * - Define ENDMMSG marking the end of socket message between tpcc_client   * and tpcc_monitor * - Remove ENCINA_C_CALLING_CONVENTION from err_printf * [from r1.9 by delta wenjian-22495-TPCC-add-new-feature-to-monitor-tpcc-clients, r1.1] * * Revision 1.9 1998/02/17 22:13:41 wenjian * [merge of changes from 1.6 to 1.7 into 1.8] * * Revision 1.7 1998/02/17 16:04:41 oz * - Split the login into two parts to allow for special logins * - If the warehouse ID is 0, this is a special login to   * query the client for status * [from r1.6 by delta oz-21864-TPCC-split-client-login-screen, r1.1] * * Revision 1.8 1998/02/17 22:07:00 wenjian * Minor changes for NT </pre>	<pre> * [from r1.6 by delta wenjian-21750-TPCC-changes-for-porting-on-NT, r1.1] * * Revision 1.6 1998/01/26 16:43:32 oz * - Removed the code for collecting stats in the client   * and dumping them before exit. * * - Removed timeP and time_allocated from thread_info_t * [from r1.5 by delta oz-21691-TPCC-remove-client-stats-code, r1.1] * * Revision 1.5 1998/01/26 16:19:23 oz * - moved all the code pertaining to the background   * thread to its own file and all the data structures   * to client_utils.h * [from r1.4 by delta oz-21689-TPCC-move-client-bg-thread-to-separate-file, r1.1] * * Revision 1.4 1998/01/23 15:07:47 oz * - Updated the SP TPCC directory to the latest files used   * during the SP tpcc audit. * [from r1.3 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1] * * *      client_utils.h * *      Generic utilities used by the client processes */  #ifndef TPCC_CLIENT_UTILS_H #define TPCC_CLIENT_UTILS_H  #include "common/tpcc_type.h" #include &lt;stdio.h&gt; #include &lt;time.h&gt; #include &lt;encina/encina.h&gt; #include "client.h" #ifdef WIN32 #include &lt;winsock.h&gt; #endif  /*  * err_printf  *      Print a string to stderr after prefixing it with the client   * info and the current time.  * log_printf  *      Prints as above to the log file.  */ #ifdef WEB_TPCC_CLIENT extern FILE *errtpcc; #endif extern FILE *logtpcc; extern char log_file_name[]; extern void logprintf( char *format, ...); extern void err_printf( char *format, ...); extern void prefix_sprintf( char *buf, char *format, ...);  /* tran_timing_t: for debug:  * Keep track of the timestamps of all the transactions  * and dump it out upon exit. There is an array of timestamps  * per thread and each thread dumps it when it exits.  */ typedef struct {     int server;     int terminal;     int tran;     int sub_tran; /* Subclass: for NewOrder and payment: 1=&gt;hasRemote */     struct timeval start; /* Time received from terminal */     struct timeval send; /* Time the RPC was made (explicit only) */     struct timeval svr_start; /* Time received by server */     struct timeval svr_done; /* Time sent by server */     struct timeval end; /* Time sent to terminal */     int num_rms; /* Number of RMs the tran involved */     int tran_failed; } tran_timing_t;  typedef enum {     thread_state_init = 0,     thread_state_called,     thread_state_sent,     thread_state_received,     thread_state_returned,     thread_state_done } thread_state_t;  #define NUM_STATES thread_state_done #define ENDMMSG "... " /* a special string to mark the end of a message */  typedef struct {     thread_state_t state;     int tran;     struct timeval init, called, sent, received, returned, done;     int printed, done_printed; } thread_descr_t;  typedef struct {     int num;     int errs;     double RTtotal[2];     int RTcount; } tran_info_t; */ </pre>
--	---

```

* total_tran_count_t
*
* structure that holds the total count of transaction of each type
* as well as the reposne times.
*
*/
typedef struct {
    tran_info_t tran[MAX_TRAN_TYPE + 1];
    int errors;
    double time; /* used for tools/tpcc_monitor.c */
} total_tran_count_t;

* enc_status_t
* structure that holds error information
*/
typedef struct {
    int status;
    int line;
    char file[268];
    unsigned long encinaError;
    char errorMsg[ENCINA_MAX_STATUS_STRING_SIZE];
} enc_status_t;

*
* thread_info_t
*
* per thread information kept by this module
*/
typedef struct {
    int thread_index;
    int thread_id;
    int initialized;
    tran_timing_t last_tran;
    int num_trans;
    int consecutive_errors;
    thread_descr_t descr;
    tran_info_t tran[MAX_TRAN_TYPE + 1];
    int done;
} thread_info_t;

int time_diff_ms(struct timeval *t2, struct timeval *t1);

extern int debug;
#define DPRINT(args) if (debug) err_printf args

extern MUTEX_T init_lock;
extern int info_list_len;
extern thread_info_t **info_list; /* List of all the thread info */

*
* A global variable by which the process would like to
* identify itself in the prefix to output
*/
extern int user_id;

*
* clientUtils_ReportReturn
* Called when a transaction is returned in order to error codes
*/
extern void clientUtils_ReportReturn(char *msg, data_header *statusP);

#define CHECK_ENVIRON(str,var) if (str == NULL) { fprintf(ERROROUT, \
    "%s environment variable is not defined.\n",var); }

char *clientUtils_TransCodeToName(int type);

#endif /* TPCC_CLIENT_UTILS_H */

    databuf.h

*
* databuf.h
*
* $Revision: 1.1 $
* $Date: 1998/11/06 21:10:11 $
* $Log: databuf.h,v $
* Revision 4.2 95/05/16 10:55:31 10:55:31 tpcc (TPCC Benchmark)
* Added necessary RCS ident strings
*
* Revision 4.1 95/05/09 15:21:02 15:21:02 strue (Scott Truesdale)
* New code from Transarc - initial version
*
* Revision 3.2 95/04/03 17:43:09 17:43:09 strue (Scott Truesdale)
* Changes from Transarc - added sql error handling in client; cleaned up debug handling with macros; added
check on db paramters via call to server.
*
* Revision 3.1 95/04/03 15:10:30 15:10:30 strue (Scott Truesdale)
* Base of rev 3 - shipped to transarc
*
*
*
*
* $TALog: databuf.h,v $
* Revision 1.1 1998/11/06 21:10:11 dongfeng
* - Move all files common to client and server to tpcc/common
* directory
* [added by delta dongfeng-23677-TPCC-new-directory-structures, r1.1]
*
* Revision 1.3 1998/10/22 15:33:04 wenjian
* Make changes to Encina server code to connect with SQL server and add
* callsql.c and sql directory.
*
* Add ERR_BAD_ITEM_ID, which is returned by SLQnew and same as INVALID_NEWO
* [from r1.2 by delta wenjian-23529-TPCC-integrate-with-SQL-server, r1.1]
*
* Revision 1.2 1998/01/23 15:07:47 oz
* - Updated the SP TPCC directory to the latest files used
* during the SP tpcc audit.
* [from r1.1 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
*
* Revision 1.1 1997/04/20 11:57:57 oz
* - This is the code base modified at IBM Poughkeepsie
* by Ofer Zajicek and Radha Sivaramakrishnan for the
* SP scaling test for TPCC.
* [added by delta oz-19782-TPCC-add-ibm-sp-code, r1.1]
*
* Revision 1.31 1995/10/30 19:10:54 oz
* [merge of changes from 1.29 to 1.30 into 1.27]
*
* Revision 1.30 1995/10/27 15:41:30 oz
* - Modified the tpc-c code to work with the new informix
* sql code that is in ex_trans.ec
* [from r1.29 by delta oz-16761-TPCC-modify-code-to-work-with-oracle, r1.1]
*
* Revision 1.27 1995/10/20 18:44:30 ctipper
* [merge of changes from 1.17 to 1.25 into 1.22]
*
* Revision 1.25 1995/10/20 18:15:34 ctipper
* Incorporate changes per code review.
*
* - add DISTRIBUTED_TRAN_FAILED, TPCC_DB_INFO_PARTIAL, and
* TPCC_DB_INFO_FAILED error codes to tpcc_rc_t
* - got rid of MAX_NUM_SERVERS variables
* [from r1.23 by delta ctipper-16547-TPCC-more-distributed-trans, r1.2]
*
* Revision 1.23 1995/10/13 17:00:26 ctipper
* This delta encompasses all changes necessary to do distributed, XA
* transactions with the TPCC benchmark. This includes the changes
* necessary to build with Informix version 6.
*
* Each client still talks to only one server, however, if a distributed
* transaction is necessary, the client sends the request to a different
* interface of that server which then forwards all or part of the
* request on to the appropriate remote server.
*
* - added new error codes to the tpcc_rc_t enumeration.
* - defined MAX_NUM_SERVERS to be 10
* [from r1.19 by delta ctipper-16547-TPCC-more-distributed-trans, r1.1]
*
* Revision 1.19 1995/09/20 21:02:39 oz
* -Corrected code for the payment transaction
* - The distributed case now no longer uses
* stored procedures
* [from r1.18 by delta oz-16547-TPCC-add-distributed-transactions, r1.2]
*
* Revision 1.18 1995/09/20 17:51:10 oz
* - Added distributed transactions for the new order and
* payment transaction
*
* - Added new error codes
* [from r1.17 by delta oz-16547-TPCC-add-distributed-transactions, r1.1]
*
* Revision 1.22 1995/10/02 20:31:07 oz
* - Corrected definition of ERROR()
* [from r1.21 by delta oz-16638-tpcc-modify-terminal-for-RTE, r1.3]
*
* Revision 1.21 1995/10/02 18:51:45 oz
* - Added definitions needed for utils.c and liberty.c
* [from r1.20 by delta oz-16638-tpcc-modify-terminal-for-RTE, r1.2]
*
* Revision 1.20 1995/10/02 15:52:35 oz
* - Modified the TPC-C benchmark to be compatible with the RTE.
* - There are now 3 terminal processes:
* emulator: the old terminal process with a built in
* simple emulator
* curses: An interactive terminal process using curses
* liberty: An interactive terminal process to be used with
* the RTE compatible with the liberty freedom terminal.
*
* - Define TRUE and FALSE only if they are not already defined.
* (curses.h defines TRUE)
* - Removed READ_TO_DATE and YEAR_TO_SECOND
* - Added term_type_t
* - Added
* GOOD_INPUT (0)
* WRONG_INPUT (10)
* [from r1.17 by delta oz-16638-tpcc-modify-terminal-for-RTE, r1.1]
*
* Revision 1.17 1995/07/28 15:28:23 oz
* - Added a -null and -no_marshall option to TPCC
*
* - Added INVALID_TRAN_TYPE return code
* [from r1.16 by delta oz-16070-TPCC-add-null-and-marshalling-test, r1.1]
*
* Revision 1.16 1995/07/18 17:02:38 oz
* - Added a DCE_ERROR error code
* [from r1.15 by delta oz-15938-TPCC-add-dce-only-client, r1.1]

```

```

*
* Revision 1.15 1995/05/22 19:50:48 shl
* [merge of changes from 1.12 to 1.13 into 1.14]
*
* Revision 1.13 1995/05/18 15:11:27 oz
* [from r1.12 by delta oz-15290-TPCC-incorporate-hp-drop-of-05-16-95, r1.1]
*
* Revision 1.14 1995/05/22 17:26:35 ctipper
* [merge of changes from 1.5 to 1.9 into 1.11]
*
* [*** log entries omitted ***]
*
*/

#ifndef TPCC_DATABUF_H
#define TPCC_DATABUF_H

#define I_NAME_LEN 24
#define I_DATA_LEN 50
#define W_NAME_LEN 10
#define ADDR_LEN 20
#define STATE_LEN 2
#define ZIP_LEN 9
#define DIST_INFO_LEN 24
#define S_DATA_LEN 50
#define D_NAME_LEN 10
#define H_DATA_LEN 24
#define CARRIER_LEN 2
#define C_LAST_LEN 17
#define C_MID_LEN 2
#define PHONE_LEN 16
#define CREDIT_LEN 2
#define C_DATA_LEN 500
#define BC_DTA_LEN 23

#define YEAR_TO_DATE 1
#define YEAR_TO_SECOND 2

#define ERROR(x) fprintf(stderr, "Error: %s\n", #x), exit(11)

#define MAX_STR_LEN 255
#define MAX_OL 15

#ifndef TRUE
#define TRUE 1
#endif
#ifndef FALSE
#define FALSE 0
#endif

#define CANCEL -1

#define DATETIME_LEN 19

#define D_PER_W 10

#define COLLECTOR 1 /* ctipper 5/3/95 */

#define ERR_BAD_ITEM_ID 1 /* copied from sql/tpcc.h */
#define RPC_ERROR -2
#define SUCCESS_CODE 0

#define CHAR_NULL '0' /* strue 1/23/95 */

typedef enum {
    liberty_term,
    curses_term,
    emulator_term
} term_type_t;

typedef enum {
    TPCC_SUCCESS = 0,
    GOOD_INPUT = 0,

    INVALID_NEWO = 100,
    SQL_ERROR = 2,
    TRPC_ERROR = 3,
    DCE_ERROR = 4,
    NO_SUCH_LAST_NAME = 5,
    INVALID_TRAN_TYPE = 6,
    INVALID_HANDLE = 7,

    WRONG_INPUT = 10,

    DISTRIBUTED_TRAN_FAILED = 15,

    TPCC_DB_INFO_PARTIAL = 20,
    TPCC_DB_INFO_FAILED,

    TPCC_ERROR_BEGIN_NEWO = 110,

    TPCC_ERROR_DECL_NEWO_SEL_ITEM,
    TPCC_ERROR_OPEN_NEWO_SEL_ITEM,
    TPCC_ERROR_OPEN_DIST_NEWO_SEL_ITEM,
    TPCC_ERROR_FETCH_NEWO_SEL_ITEM,
    TPCC_ERROR_FETCH_DIST_NEWO_SEL_ITEM,
    TPCC_ERROR_PREP_NEWO_SEL_STCK,
    TPCC_ERROR_DECL_NEWO_SEL_STCK,
    TPCC_ERROR_OPEN_NEWO_SEL_STCK,

    TPCC_ERROR_OPEN_DIST_NEWO_SEL_STCK,
    TPCC_ERROR_FETCH_NEWO_SEL_STCK,
    TPCC_ERROR_FETCH_DIST_NEWO_SEL_STCK,
    TPCC_ERROR_NEWO_SELECT,
    TPCC_ERROR_NEWO_UPD_STCK,
    TPCC_ERROR_DIST_NEWO_UPD_STCK,
    TPCC_ERROR_NEWO_SELECT_2,
    TPCC_ERROR_DECL_NEWO_SEL_CUST,
    TPCC_ERROR_OPEN_NEWO_SEL_CUST,
    TPCC_ERROR_OPEN_DIST_NEWO_SEL_CUST,
    TPCC_ERROR_FETCH_NEWO_SEL_CUST,
    TPCC_ERROR_FETCH_DIST_NEWO_SEL_CUST,
    TPCC_ERROR_DECL_NEWO_SEL_DIST,
    TPCC_ERROR_OPEN_NEWO_SEL_DIST,
    TPCC_ERROR_OPEN_DIST_NEWO_SEL_DIST,
    TPCC_ERROR_FETCH_NEWO_SEL_DIST,
    TPCC_ERROR_FETCH_DIST_NEWO_SEL_DIST,
    TPCC_ERROR_PREP_NEWO_INS_OL,
    TPCC_ERROR_DECL_NEWO_INS_OL,
    TPCC_ERROR_OPEN_NEWO_INS_OL,
    TPCC_ERROR_OPEN_DIST_NEWO_INS_OL,
    TPCC_ERROR_PUT_NEWO_INS_OL,
    TPCC_ERROR_PUT_DIST_NEWO_INS_OL,
    TPCC_ERROR_DECL_NEWO_SEL_WARE,
    TPCC_ERROR_OPEN_NEWO_SEL_WARE,
    TPCC_ERROR_OPEN_DIST_NEWO_SEL_WARE,
    TPCC_ERROR_FETCH_NEWO_SEL_WARE,
    TPCC_ERROR_FETCH_DIST_NEWO_SEL_WARE,
    TPCC_ERROR_EXECUTE_NEWO_UPD_INS,
    TPCC_ERROR_UPDATE_NEWO_NEXT_OID,
    TPCC_ERROR_PREP_NEWO_INS,
    TPCC_ERROR_EXECUTE_DIST_NEWO_INS,
    TPCC_ERROR_EXECUTE_NEWO_COMMIT,
    TPCC_ERROR_ROLLBACK_NEWO,
    TPCC_ERROR_REMOTE_OL_SELECT,
    TPCC_ERROR_REMOTE_OL_UPDATE,

    TPCC_ERROR_OPEN_ORDS_CNT_CID = 200,
    TPCC_ERROR_FETCH_ORDS_CNT_CID,
    TPCC_ERROR_OPEN_ORDS_SEL_CLAST,
    TPCC_ERROR_FETCH_ORDS_SEL_CLAST,
    TPCC_ERROR_OPEN_ORDS_SEL_CID,
    TPCC_ERROR_FETCH_ORDS_SEL_CID,
    TPCC_ERROR_OPEN_ORDS_SEL_OLDORD,
    TPCC_ERROR_FETCH_ORDS_OLDORD,
    TPCC_ERROR_OPEN_ORDS_SEL_OL,
    TPCC_ERROR_FETCH_ORDS_SEL_OL,
    TPCC_ERROR_EXECUTE_ORDS_COMMIT,

    TPCC_ERROR_OPEN_DELIVERY_OLDEST_OID = 300,
    TPCC_ERROR_FETCH_DELIVERY_OLDEST_OID,
    TPCC_ERROR_EXECUTE_DELIVERY_COMMIT,
    TPCC_ERROR_OPEN_DELIVERY_SEL_ORD,
    TPCC_ERROR_FETCH_DELIVERY_SEL_ORD,
    TPCC_ERROR_OPEN_DELIVERY_SEL_SUM_OL,
    TPCC_ERROR_FETCH_DELIVERY_SEL_SUM_OL,
    TPCC_ERROR_EXECUTE_DELIVERY_EXEC_DVRY,
    TPCC_ERROR_SELECT_DELIVERY_ORDER_ID,
    TPCC_ERROR_SELECT_DELIVERY_CARRIER_ID,
    TPCC_ERROR_SELECT_DELIVERY_BALANCE,

    TPCC_ERROR_OPEN_STOCKLEVEL_SEL_OID = 400,
    TPCC_ERROR_FETCH_STOCKLEVEL_SEL_OID,
    TPCC_ERROR_OPEN_STOCKLEVEL_CNT_SID,
    TPCC_ERROR_FETCH_STOCKLEVEL_CNT_SID,
    TPCC_ERROR_OPEN_STOCKLEVEL_FIND,
    TPCC_ERROR_FETCH_STOCKLEVEL_FIND,
    TPCC_ERROR_EXECUTE_STOCKLEVEL_COMMIT,

    TPCC_ERROR_OPEN_PAYMENT_CNT_CID = 500,
    TPCC_ERROR_FETCH_PAYMENT_CNT_CID,
    TPCC_ERROR_OPEN_PAYMENT_SEL_CLAST,
    TPCC_ERROR_FETCH_PAYMENT_SEL_CLAST,
    TPCC_ERROR_OPEN_PAYMENT_SEL_CID,
    TPCC_ERROR_FETCH_PAYMENT_SEL_CID,
    TPCC_ERROR_DECL_PAYMENT_SEL_DIST,
    TPCC_ERROR_OPEN_PAYMENT_SEL_DIST,
    TPCC_ERROR_OPEN_DIST_PAYMENT_SEL_DIST,
    TPCC_ERROR_FETCH_PAYMENT_SEL_DIST,
    TPCC_ERROR_FETCH_DIST_PAYMENT_SEL_DIST,
    TPCC_ERROR_DECL_PAYMENT_SEL_WARE,
    TPCC_ERROR_OPEN_PAYMENT_SEL_WARE,
    TPCC_ERROR_OPEN_DIST_PAYMENT_SEL_WARE,
    TPCC_ERROR_FETCH_PAYMENT_SEL_WARE,
    TPCC_ERROR_FETCH_DIST_PAYMENT_SEL_WARE,
    TPCC_ERROR_EXECUTE_PAYMENT_UPD_CUST_LAST,
    TPCC_ERROR_EXECUTE_PAYMENT_UPD_CUST_ID,
    TPCC_ERROR_COMMIT_PAYMENT_UPD_CUST,
    TPCC_ERROR_SELECT_PAYMENT_W_YTD,
    TPCC_ERROR_SELECT_PAYMENT_D_YTD,
    TPCC_ERROR_BEGIN_PAYMENT,
    TPCC_ERROR_EXECUTE_PAYMENT_COMMIT,
    TPCC_ERROR_PAYMENT_UPD_CUST_BY_NAME,
    TPCC_ERROR_PAYMENT_UPD_CUST_BY_ID,
    TPCC_ERROR_PAYMENT_UPDATE_DIST,
    TPCC_ERROR_PAYMENT_UPDATE_WH,
    TPCC_ERROR_PAYMENT_INSERT_HISTORY,
    TPCC_ERROR_EXECUTE_PAYMENT_WH_DIST
}

```

```

} tpcc_rc_t;

typedef enum {
    TPCC_DEADLOCK_MSG = 10,
    TPCC_RETRY_MSG
} tpcc_msg_t;

#endif /* __TPCC_DATABUF_H__ */

                                debug.c

#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include <sys/stat.h>
#include <errno.h>
#include <math.h>
#include <time.h>
#include <fcntl.h>
#include <unistd.h>
#ifdef WIN32
#include <process.h>
#else
#include <termio.h>
#endif

int print_thread_id = 0;
int user_id = 1;
char *user_code = "C";

int get_thread_id()
{
    return(0);
}

* get_prefix
* Format the output prefix for printing:
* It contains the user_id, 'C' or 'T' depending on whether it
* is a terminal or a client and optional a thread identifier
* The prefix is written in the buffer passed in by the caller.
*/

void get_prefix(buffer)
char *buffer;
{
    if (print_thread_id) {
        int thread_id = get_thread_id();
        sprintf(buffer, "%s(%d-%s-%d)%s",
            user_id < 10 ? " " : user_id < 100 ? " " : "",
            user_id,
            user_code,
            thread_id,
            thread_id < 10 ? " " : "");
    } else {
        sprintf(buffer, "%s(%2d-%s)",
            user_id < 10 ? " " : "", user_id, user_code);
    }
}

* err_printf
* A var-arg function that appends the current time and
* other data to the print request and sends it to stderr
*/

void err_printf(char *format, ...)
{
    static int initialized = 0;
    static FILE *debug_f = NULL;
    time_t cur_timet;
    char time_str[30];
    char line_prefix[50];
    va_list ap;

    va_start(ap, format);

    if (!initialized) {
        char fileName[45];
        initialized = 1;
        sprintf(fileName, "DebugFile.%d", getpid());
        debug_f = fopen(fileName, "w");
    }

    cur_timet = time(&cur_timet);
    strftime(time_str, 29, "%X", localtime(&cur_timet));

    get_prefix(line_prefix);

    if (debug_f) {
        fprintf(debug_f, "%s %s - ", line_prefix, time_str);
        vfprintf(debug_f, format, ap);
        fflush(debug_f);
    }

    va_end(ap);
}

void set_client_debug_state(void *contextP, int state, int tran)

```

```

{
}

                                delivery.tacf

/*
 * Copyright (C) 1991, 1990 Transarc Corporation
 * All Rights Reserved
 */
/*
 * neworder.tacf -- attribute configuration file for tpcc server.
 * used for transparent binding
 *
 * $Revision: 1.1 $
 * $Date: 1998/11/06 21:10:11 $
 * $Log: tpcc.tacf,v $
 *
 * $TALog: delivery.tacf,v $
 * Revision 1.1 1998/11/06 21:10:11 dongfeng
 * - Move all files common to client and server to tpcc/common
 * directory
 * [added by delta dongfeng-23677-TPCC-new-directory-structures, r1.1]
 *
 * Revision 1.1 1997/04/20 11:57:57 oz
 * - This is the code base modified at IBM Poughkeepsie
 * by Ofer Zajicek and Radha Sivaramakrishnan for the
 * SP scaling test for TPCC.
 * [added by delta oz-19782-TPCC-add-ibm-sp-code, r1.1]
 *
 * Revision 1.3 1996/01/12 16:06:44 oz
 * - Added transaction specific servers: there are 5 different interfaces
 * one for each transaction type.
 * [added by delta oz-16955-TPCC-add-transaction-specific-servers, r1.1]
 */

[implicit_handle (mon_handle_t handle)]
interface delivery
{
}

                                delivery.tidl

/*
 * id: Sid: $
 *
 * component_name: encina benchmarks
 *
 * the following functions list may not be complete.
 * functions defined by/via macros may not be included.
 *
 * functions:
 * <fill_me_in>
 *
 * origins: transarc corp.
 *
 * (c) copyright transarc corp. 1995, 1993
 * all rights reserved
 * licensed materials - property of transarc
 *
 * us government users restricted rights - use, duplication or
 * disclosure restricted by gsa adp schedule contract with transarc corp
 */
/*
 * history
 * $Talog: $
 */

/*
 * delivery.tidl -- interface definition file for tpccserver.
 *
 * $Revision: 1.11 $
 * $date: 1995/10/20 21:55:05 $
 * $Log: tpcc.tidl,v $
 */

[uuid(d714d8f8-2105-11cf-830f-0800093b9834), version(1.0)]

interface delivery
{
    import "tpm/mon/mon_handle.idl";
    import "tpcc_type.idl";

    [nontransactional] void
        impTPCCDelivery([in,out] delivery_data_t *dataP,
            [out] trpc_status_t * trpcStatus);
}

                                do_tpcc.c

/*
 * do_tpcc.c
 *
 * $Revision: 1.12 $
 * $Date: 1999/05/06 21:28:26 $
 */

```

```

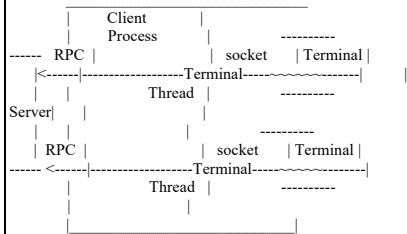
* $Log: do_tpcc.c,v $
*
* $TALog: do_tpcc.c,v $
* Revision 1.12 1999/05/06 21:28:26 oz
* - Removed all the .. from the includes
* - Added -l. to the makefiles instead
* - Moved all the thread related code and connection
* selection to serverMon.c
* [from r1.8 by delta oz-24309-TPCC-add-oracle8.1-code, r1.5]
*
* Revision 1.8 1999/01/29 20:16:32 wenjian
* Call init_encina_client since client_init has been renamed to
* init_encina_client
* [from r1.7 by delta wenjian-23787-TPCC-integrate-code-for-AIX-and-NT, r1.7]
*
* Revision 1.7 1998/12/09 14:44:53 wenjian
* Add a call to client_init() in main
* [from r1.6 by delta wenjian-23787-TPCC-integrate-code-for-AIX-and-NT, r1.3]
*
* Revision 1.6 1998/11/09 16:59:37 wenjian
* In this revision, most of the changes are related to the directory of header
* files after directory reorganization. Other changes include adding or removing
* files to put them in the right directories. Makefiles are written for NT
* platform so that nmake is working on NT now. Need a top level Makefile for all
* the directories.
* [from r1.5 by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.2]
*
* Revision 1.5 1998/11/09 14:48:15 wenjian
* In an effort to make a new directory structure for TPCC, this delta
* creates two directories: tpcc/client and tpcc/server. All the files
* for this revision are copied from tpcc/sp-tpcc without any change.
* Further change may be needed for some files due to the change of
* the directory structure.
* [added by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.1]
*
* Revision 1.15 1998/02/17 22:07:01 wenjian
* Define macros to deal with the different function names on NT
* [from r1.14 by delta wenjian-21750-TPCC-changes-for-porting-on-NT, r1.1]
*
* Revision 1.14 1998/01/26 20:37:34 oz
* - Remove all the code associated with explicit binding
*
* - Removed bindingType
* - Removed client_first_wh and client_last_wh
* - Removed command line args: binding, offset, ware
* [from r1.13 by delta oz-21697-TPCC-remove-explicit-binding-code, r1.1]
*
* Revision 1.13 1998/01/26 15:33:31 oz
* - Changed default binding to transparent
* [from r1.12 by delta oz-21671-TPCC-merge-online-transaction-interfaces, r1.2]
*
* Revision 1.12 1998/01/24 14:17:05 oz
* - User server name to identify server and name delivery file
* - Use env variable HOME instead of /home/encina if HOME is set
* [from r1.11 by delta oz-21687-TPCC-use-server-name-to-identify-process, r1.1]
*
* Revision 1.11 1998/01/23 15:07:48 oz
* - Updated the SP TPCC directory to the latest files used
* during the SP tpcc audit.
* [from r1.10 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
*
* Revision 1.8 1997/08/04 19:50:41 oz
* [from r1.7 by delta oz-20506-TPCC-convert-to-new-format-of-connection-manager, r1.2]

```

```

* do_tpcc.c
*
* This is the main client program for the TPCC benchmark using Encina.
*
* The client program is multi-threaded: there is one thread for each
* terminal and one thread to process incoming connections.
*
* When the client starts up it starts a listening thread. That thread
* calls the encina function cnm_ManageConnections and provides it a
* port number. The client spawns a thread (through cnm_ManageConnections)
* for each terminal that connects to it. That thread receives the input
* from the terminal and translates it to a transaction data structure
* (such as payment_data_t or newOrder_data_t). The terminal thread
* (in the client) then sends an RPC over to the server to process the request.

```



```

*
*
*=====
*/

#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include <sys/stat.h>
#include <errno.h>
#include <math.h>
#include <time.h>
#include <fcntl.h>
#ifdef WIN32
#include <termio.h>
#endif
#include <unistd.h>
#include <sys/ipc.h>
#include <tpm/mon/mon_client.h>
#include <tc/tc.h>
#include <dce/rpc.h>

#ifdef solaris
#include <dce/pthread.h>
#else /* solaris */
#include <pthread.h>
#endif

#include "common/tpcc_type.h"
#include "common/utilities.h"
#include "client_utils.h"
#include "common/do_tpcc.h"
#include "client_listen.h"
#include "client.h"

extern char *sys_errlist[]; /* Translations of errno file errors */

#ifdef WIN32
#define STRCMP _stricmp
#else
#define STRCMP strcasecmp
#endif

/*
 * ENTERING
 * A macro that is called before processing a TPCC transaction.
 * If debug mode is enabled it prints a message containing the name
 * of the transaction being executed.
 */
#define ENTERING(msg) if (debug) err_printf("Entering %s\n", msg)

#define MAX_CONSECUTIVE_ERRS 3000
/*
 * RETURNED
 * A macro that is called after a transaction has been processed.
 * If the transaction failed it reports an error.
 * In debug mode it also emits a message indicating the processing
 * has been completed.
 */
#define RETURNED(msg, hdrP) \
{ \
    if (((hdrP)->returncode == TPCC_SUCCESS) || \
        ((hdrP)->returncode == INVALID_NEWO)) { \
        consecutiveErrors = 0; \
    } else { \
        consecutiveErrors++; \
    } \
    if (debug || \
        (((hdrP)->returncode != SUCCESS_CODE) && \
         ((hdrP)->returncode != INVALID_NEWO))) { \
        clientUtils_ReportReturn(msg, hdrP); \
        if (consecutiveErrors > MAX_CONSECUTIVE_ERRS) { \
            err_printf("Too many consecutive errors (%d)\n", \
                consecutiveErrors); \
            exit_program(1); \
        } \
    } \
}

int useSecurity = FALSE;
int null_test = 0;
int client_lock_handles = 0;

/* The following are global to the client */
char *LOG_FILE_DIR = "runs/threads";
int user_id;
int user_port = 4011;
char *user_code = "C"; /* Prefix for output to identify this
                        * process as a client or a terminal
                        */

int consecutiveErrors = 0;
char *result_dir;

int debug = 0;
char log_file_name[100];
int logtrans = 0;

```



```

FILE *logtpcc = NULL;

static void check_parms(int argc, char *argv[]);
static void print_header(int argc, char *argv[]);

/*=====*/

main(argc, argv)
int argc;
char *argv[];
{
    check_parms(argc,argv); /* Read and parse the command line parameters */

    err_printf("Client %d starting.\n", user_id);

    init_encina_client(user_id);
    enroll_client(user_id); /* enroll as a client */

    /*
     * Open log file
     */
    logtpcc = fopen(log_file_name, "w");
    print_header(argc, argv); /* Print a test header to the logfile */

    /*
     * Start the listening thread:
     * This call will not return
     */
    make_connections((void *)user_port);

    exit_program(0);
    return (0); /* to satisfy lint */
} /*=====*/

/*
 * User must supply user_id as a parm and all other parameters
 * as environment variables.
 */

/*-----*/
/*
 * Check Parameters
 * Check the parameters passed in.
 *
 * Not all the parameters are relevant for this executable.
 * This code is shared between the regular Encina Monitor
 * based TPC-C client and other test clients that do not
 * use the Encina Monitor. The type of this executable is
 * in client_type and is set to mon_client for the TPCC
 * Monitor based client (the audited client).
 */
/*-----*/
static void check_parms (argc,argv)
int argc;
char *argv[];
{
    char *host_name = getenv("HOST");
    char *home_dir = getenv("HOME");
    int next_arg = 1;
    int errors = 0;
    char *progName;
    int print_help = 0;

    user_id = -1;
    result_dir = ".";

    while (next_arg < argc) {
        if (!STRCMP("-debug", argv[next_arg])) {
            /* Enable debug mode (for testing) */
            debug = 1;
        } else if (!STRCMP("-dir", argv[next_arg])) {
            /* The directory for the client output */
            result_dir = argv[next_arg];
        } else if (!STRCMP("-log", argv[next_arg])) {
            /* A less intrusive form of debug mode */
            logtrans = 1;
        } else if (!STRCMP("-id", argv[next_arg])) {
            /* The id of this client */
            user_id = atoi(argv[next_arg]);
        } else if (!STRCMP("-port", argv[next_arg])) {
            /* The id of this client */
            user_port = atoi(argv[next_arg]);
            if (user_id < 0) user_id = user_port;
        } else if (!STRCMP("-security", argv[next_arg])) {
            /* Enable security between the client and the server.
             * This is enabled by default
             */
            useSecurity = TRUE;
        } else if (!STRCMP("-noSecurity", argv[next_arg])) {
            /* Disable security between the client and the server.
             * This is enabled by default
             */
            useSecurity = FALSE;
        } else if (!STRCMP("-null", argv[next_arg])) {

```

```

/* For testing: do not access the data in the DB */
logprintf("Performing NULL test\n");
null_test = 1;
} else if (!STRCMP("-lock", argv[next_arg])) {
    logprintf("Locking longterm handles\n");
    client_lock_handles = atoi(argv[next_arg]);
} else {
    printf("invalid parameter: %s\n", argv[next_arg]);
    print_help = 1;
    break;
}
next_arg++;
}

if (user_id < 0) {
    printf("Missing User Id\n");
    print_help = 1;
}

if (print_help) {
    progName = strchr(argv[0], '/');
    progName = (progName ? progName + 1 : argv[0]);

    printf("\nusage:\n You can specify the following in any order\n");
    printf(" You must specify the Id\n");

    printf(" -id <num>      The user ID for this client\n");
    printf(" -dir <dir>      Directory for output (default \".\")\n");
    printf(" -debug          enable debugging\n");
    printf(" -log           log all activity to a file\n");
    printf(" -security      enable secure communications between the client and PA\n");
    printf(" -null         NULL test: the server immediately returns\n");

    exit(-1);
}

sprintf(log_file_name, "%s/%s/C.%s.%d",
        home_dir ? home_dir : "home/encina",
        LOG_FILE_DIR,
        host_name ? host_name : "host", user_id);
}

/*
 * print_header:
 * Print some feedback to the user on the client configuration
 */
static void print_header(int argc, char *argv[])
{
    int i;
    if (!logtpcc)
        return;

    logprintf("Client %d starting a %s test.\n",
            user_id,
            null_test ? "NULL" : "DB");

    logprintf("Params: ");

    for (i=0; i<argc; i++) {
        fprintf(logtpcc, "%s ", argv[i]);
    }
    fprintf(logtpcc, "\n");
    fflush(logtpcc);
}

do tpcc.h

/*
 * do_tpcc.h
 *
 * $Revision: 1.1 $
 * $Date: 1998/11/09 16:00:05 $
 * $Log: do_tpcc.h,v $
 *
 * $TALog: do_tpcc.h,v $
 * Revision 1.1 1998/11/09 16:00:05 dongfeng
 * Move do_tpcc.h to common directory
 * [added by delta dongfeng-23677-TPCC-new-directory-structures, r1.4]
 *
 * Revision 1.7 1998/01/23 15:07:49 oz
 * - Updated the SP TPCC directory to the latest files used
 * during the SP tpcc audit.
 * [from r1.6 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
 *
 */

#ifdef DO_TPCC_H_INCLUDED_
#define DO_TPCC_H_INCLUDED_

#include <dce/tpc.h>
#include <trpc/tpc.h>
#include "databuf.h"

#define WRONG_INPUT 0
#define NEW_ORDER 1
#define PAYMENT 2
#define ORDER_STATUS 3
#define DELIVERY 4

```

```

#define STOCK_LEVEL 5
#define QUIT 9
#define MIN_OL 5
#define MAX_FLDS 200 /* Maximum fields in a TPC-C form */
#define THRESHOLD_LEN 2
#define ON 1
#define OFF 0
#define YES 1
#define NO 0
#define INSIZE 1024
#define DO_ROLLBACK 1
#define DONT_ROLLBACK 0
/** The response time requirements for the transactions in seconds.
** 90% of the transactions are required to have a response time less
** than or equal to the value below.
**/
#define NEWORD_90RT 5
#define PAYMENT_90RT 5
#define ORDSTAT_90RT 5
#define DELIVERY_90RT 5 /* 5 for interactive or 80 for background */
#define STOCKLEV_90RT 20
/*
* What type of client is this?
*/
typedef enum {
tk_client,
dce_client,
mon_client,
db_client
} client_type_t;
extern client_type_t client_type;
typedef enum {
transparent, explicit, longTerm, noReservation
} binding_t;
/* Handle from client to PA is now described using both the paHandle
and the mondHandle. */
#define NUM_TRANS 5
#define NEWO_ERR 6
#define PAYMENT_ERR 7
#define ORD_STAT_ERR 8
#define DELIVERY_ERR 9
#define STOCK_ERR 10
#define NEWO_ROLLBACK 11
#define END_OF_WINDOW 0xff
#define BEGIN_WINDOW 0xaa
#ifdef SHORT_WAITS
#define NEWO_MEAN_THINK_TIME 122
#define PAYMENT_MEAN_THINK_TIME 122
#define ORDER_STAT_MEAN_THINK_TIME 102
#define DELIVERY_MEAN_THINK_TIME 51
#define STOCK_MEAN_THINK_TIME 51
#define NEWO_MIN_KEY_TIME 185
#define PAYMENT_MIN_KEY_TIME 31
#define ORDER_STAT_MIN_KEY_TIME 21
#define DELIVERY_MIN_KEY_TIME 21
#define STOCK_MIN_KEY_TIME 21
#else
#define NEWO_MEAN_THINK_TIME 61
#define PAYMENT_MEAN_THINK_TIME 61
#define ORDER_STAT_MEAN_THINK_TIME 51
#define DELIVERY_MEAN_THINK_TIME 26
#define STOCK_MEAN_THINK_TIME 26
#define NEWO_MIN_KEY_TIME 93
#define PAYMENT_MIN_KEY_TIME 16
#define ORDER_STAT_MIN_KEY_TIME 11
#define DELIVERY_MIN_KEY_TIME 11
#define STOCK_MIN_KEY_TIME 11
#endif
#endif /* _DO_TPCC_H_INCLUDED_ */

```

**encina.C**

```

/* (C)1997 IBM Corporation */
*****
/*
*/
/* File: tuxclient.h */
*/
*****
#include <stdlib.h>
#include "inout.h"
#include "encina.h"

```

```

extern "C" {
}
extern "C" send_new_order(void *contextP, NewOrder_data *dataP);
extern "C" send_payment(void *contextP, Payment_data *dataP);
extern "C" send_stock_level(void *contextP, StockLevel_data *dataP);
extern "C" send_order_status(void *contextP, OrderStatus_data *dataP);
extern "C" send_delivery(void *contextP, Delivery_data *dataP);
void Encina::cleanup() {
}
Encina::Encina() {
return;
}
Encina::~Encina() {
return;
}
int Encina::tran(NewOrder_data *dataP, void *contextP, char *servname) {
send_new_order(contextP, dataP);
return 0;
}
int Encina::tran(Payment_data *dataP, void *contextP, char *servname) {
send_payment(contextP, dataP);
return 0;
}
int Encina::tran(OrderStatus_data *dataP, void *contextP, char *servname) {
send_order_status(contextP, dataP);
return 0;
}
int Encina::tran(StockLevel_data *dataP, void *contextP, char *servname) {
send_stock_level(contextP, dataP);
return 0;
}
int Encina::tran(Delivery_data *dataP, void *contextP, char *servname) {
send_delivery(contextP, dataP);
return 0;
}
int Encina::tran(char *servname) {
return -1;
}
int Encina::atran(char *servname) {
return 0;
}

```

**encina.h**

```

/* (C)1997 IBM Corporation */
*****
/*
*/
/* File: tuxclient.h */
/*
*/
*****
#ifdef ENCINA_H
#define ENCINA_H
const int TMINBUFSIZE = 1536;
class Encina {
public:
static void cleanup();
int tran(char *servname);
int tran(NewOrder_data *dataP, void *contextP, char *servname);
int tran(Payment_data *dataP, void *contextP, char *servname);
int tran(StockLevel_data *dataP, void *contextP, char *servname);
int tran(OrderStatus_data *dataP, void *contextP, char *servname);
int tran(Delivery_data *dataP, void *contextP, char *servname);
int atran(char *servname);
Encina();
~Encina();
};
extern Encina encina;
#endif

```

**encina\_client.c**

```

/*
*
* encina_client.c
*

```

```

* $Revision: 1.7 $
* $Date: 1999/05/06 21:28:26 $
* $Log: $
*
* $TALog: encina_client.c,v $
* Revision 1.7 1999/05/06 21:28:26 oz
* - Removed all the .. from the includes
* - Added -L. to the makefiles instead
* - Moved all the thread related code and connection
* selection to serverMon.c
* [from r1.6 by delta oz-24309-TPCC-add-oracle8.1-code, r1.5]
*
* Revision 1.6 1998/11/09 16:59:37 wenjian
* In this revision, most of the changes are related to the directory of header
* files after directory reorganization. Other changes include adding or removing
* files to put them in the right directories. Makefiles are written for NT
* platform so that nmake is working on NT now. Need a top level Makefile for all
* the directories.
* [from r1.5 by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.2]
*
* Revision 1.5 1998/11/09 14:48:16 wenjian
* In an effort to make a new directory structure for TPCC, this delta
* creates two directories: tpcc/client and tpcc/server. All the files
* for this revision are copied from tpcc/sp-tpcc without any change.
* Further change may be needed for some files due to the change of
* the directory structure.
* [added by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.1]
*
* Revision 1.5 1998/01/23 15:07:51 oz
* - Updated the SP TPCC directory to the latest files used
* during the SP tpcc audit.
* [from r1.4 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
*
*
*
*/
*
* encina_client.c
*
* The Encina related code in the client that is common to both
* the monitor client and the toolkit client.
*
*/
#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include <trpc/trpc.h>
#include <encina/encina.h>
#include "common/utilities.h"
#include "client_utils.h"
#include "encina_client.h"

static trpc_handle_t bind_to_server(char *name);

*
* encina_error_message
*
* Report an encina error message by interpreting it and writing
* it to both the logfile (if any) and to standard error
*/
void encina_error_message(msg, n)
char *msg;
unsigned long n;
{
char errorMsg[ENCINA_MAX_STATUS_STRING_SIZE];
encina_StatusToString(n, ENCINA_MAX_STATUS_STRING_SIZE, errorMsg);
err_printf("ERROR: %s. Error code = %s (%d 0x%x) \n", msg, errorMsg, n, n);
}

*
* encina_error
*
* This is called for FATAL errors. It reports the error and exits.
*/
void encina_error(funcName, n)
char *funcName;
unsigned long n;
{
char msg[128];
sprintf("%s failed", funcName);
encina_error_message(msg, n);
exit_program(1);
}

*
* secure_handle
*
* Secure a handle to an encina server.
* This can be called with either a PA handle or with
* a trpc handle to a toolkit server.
*/
void secure_handle(trpc_handle_t handle, int use_security)
{
rpc_binding_handle_t rpcHandle;
unsigned long status = 0;

```

```

unsigned char *serverPrincipal;

ENCINA_CALL("trpc_GetRpcHandleFromBinding",
trpc_GetRpcHandleFromBinding(handle, &rpcHandle));

rpc_mgmt_inq_server_princ_name(rpcHandle, rpc_c_authn_default,
&serverPrincipal, &status);

if (use_security) {
DPRINT(("rpc_binding_set_auth_info -> principal %s, protect %d, authn %d authz %d\n",
serverPrincipal, rpc_c_protect_level_connect,
rpc_c_authn_default, rpc_c_authz_dce));

rpc_binding_set_auth_info(rpcHandle, serverPrincipal,
rpc_c_protect_level_connect,
rpc_c_authn_default,
NULL,
rpc_c_authz_dce,
&status);
} else {
DPRINT(("rpc_binding_set_auth_info -> principal %s, protect %d, authn %d authz %d\n",
serverPrincipal, rpc_c_protect_level_none,
rpc_c_authn_default, rpc_c_authz_dce));

rpc_binding_set_auth_info(rpcHandle, serverPrincipal,
rpc_c_protect_level_none,
rpc_c_authn_default,
NULL,
rpc_c_authz_dce,
&status);
}
}

if (status != rpc_s_ok) {
switch (status) {
case rpc_s_invalid_binding :
printf("rpc binding invalid ***** \n");
break;
case rpc_s_wrong_kind_of_binding :
printf("rpc binding is the wrong kind \n");
break;
case rpc_s_unknown_authn_service :
printf("rpc authn service unknown \n");
break;
} /* switch */
bde_Exit(1);
}
}

}

*
* encina_client.h
*
* $Revision: 1.5 $
* $Date: 1998/11/09 14:48:16 $
* $Log: $
*
* $TALog: encina_client.h,v $
* Revision 1.5 1998/11/09 14:48:16 wenjian
* In an effort to make a new directory structure for TPCC, this delta
* creates two directories: tpcc/client and tpcc/server. All the files
* for this revision are copied from tpcc/sp-tpcc without any change.
* Further change may be needed for some files due to the change of
* the directory structure.
* [added by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.1]
*
* Revision 1.5 1998/01/23 15:07:52 oz
* - Updated the SP TPCC directory to the latest files used
* during the SP tpcc audit.
* [from r1.4 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
*
*
* Declarations common to monitor version and toolkit version
*
*/

#ifndef ENCINA_CLIENT_H
#define ENCINA_CLIENT_H

#include <trpc/trpc.h>

void encina_error_message(char *msg, unsigned long n);
void encina_error(char *funcName, unsigned long n);
void secure_handle(trpc_handle_t handle, int use_security);

#endif /* ENCINA_CLIENT_H */

*
* (C)1997 IBM Corporation */
#include <stdio.h>
#include "field.h"

```

```

#include "inout.h"
#include "format.h"
#if 0
#if USE_ALLOCA
#include <alloca.h>
#endif
#endif

extern char const * const blanks;
extern char const * const underscores;
extern char const * const backspaces;
extern int position(InOut *ioP, int x, int y);

Field *genfield(InOut *ioP, int x, int y, int len, int *ptr) {
    return new IntField(ioP, x, y, len, ptr);
}

Field *genfield(InOut *ioP, int x, int y, int len, short *ptr) {
    return new ShortField(ioP, x, y, len, ptr);
}

Field *genfield(InOut *ioP, int x, int y, int len, long *ptr) {
    return new LongField(ioP, x, y, len, ptr);
}

Field *genfield(InOut *ioP, int x, int y, int len, char *ptr) {
    return new TextField(ioP, x, y, len, ptr);
}

Field *genfield(InOut *ioP, int x, int y, int len, double *ptr) {
    return new MoneyField(ioP, x, y, len, ptr);
}

Field *genfield(InOut *ioP, int x, int y, int len, unsigned char *ptr) {
    return new Int8Field(ioP, x, y, len, ptr);
}

}

*****
Field
*****
Field::Field(InOut *inoutP, int size, char *str)
: ioP(inoutP), len(size), pos(0), changed(0), need_redisplay(0)
{
    need_free_string = need_free = 0;
    if (str == NULL) {
        string = new char[len+1];
        need_free_string = 1;
    } else {
        string = str;
    }
    ok_func = NULL;
    ok_data = NULL;
    string[0] = 0;
}

Field::Field(InOut *ioP, int inx, int iny, int size, char *str)
: ioP(ioP), x(inx), y(iny), len(size), pos(0), changed(0), need_redisplay(0)
{
    need_free_string = need_free = 0;
    if (str == NULL) {
        string = new char[len+1];
        need_free_string = 1;
    } else {
        string = str;
    }
    ok_func = NULL;
    ok_data = NULL;
    string[0] = 0;
}

int Field::reset() {
    pos=0;
    changed=0;
    return 0;
}

Field::~Field() {
    if (need_free_string)
        delete [] string;
}

int Field::finalize_field() {
    changed = 0;
    string[pos] = 0;
    return 0;
}

int Field::display_field(int use_underscores) {
    position(ioP, x,y);
    ioP->write(string);
    if (use_underscores) {
        ioP->write(underscores, len-pos);
    } else {
        ioP->write(blanks, len-pos);
    }
    return 0;
}

int Field::get_key() {
    char key;
    int cc;
    cc = ioP->read(&key, 1);

    return (cc == 0) ? EOF : key ;
}

int Field::add_char(int key) {
    if (pos >= len || (isprint(key) && key != '\n')) {
        ioP->write("a", 1);

```

```

        return 1;
    }
    changed = 1;
    string[pos] = key;
    ioP->write(&string[pos++], 1);
    return 0;
}

int Field::backspace() {
    ioP->write("\b\b", 3);
    changed = 1;
    pos--;
    return 0;
}

int Field::start_position () {
    position(ioP, x, y);
    return 0;
}

int Field::get_field (int need_pos) {
    int key;

    if (need_pos)
        position(ioP, x, y);
    if (pos != 0) {
        need_redisplay = 1;
        ioP->write(string, pos);
        ioP->write(underscores, len-pos);
        if (len-pos < 6)
            ioP->write(backspaces, len-pos);
        else
            position(ioP, x+pos, y);
    }

    ioP->mark();
    while (1) {
        key = get_key();
        switch(key) {
            case EOF:
                return EOF;

            case '\r': /* Carriage Return */
            case '\n': /* Newline */
                ioP->hold();
                if (changed) {
                    finalize_field();
                }
                ioP->pop();
                display_field(1);
                return ENTER;
                break;

            case '\t': /* Tab */
            case '\006': /* Ctrl-F */
            case '\016': /* Ctrl-N */
                if (changed) {
                    finalize_field();
                }
                ioP->pop();
                display_field(1);
                return NEXT_FIELD;
                break;

            case '\002': /* Ctrl-B */
            case '\020': /* Ctrl-P */
                if (changed) {
                    finalize_field();
                }
                ioP->pop();
                display_field(1);
                return PREV_FIELD;
                break;

            case '\b': /* Backspace */
            case '\177': /* Del */
                if (pos > 0) {
                    backspace();
                } else
                    ioP->write("a", 1);
                break;

            case '\014': /* Ctrl-L */
                ioP->pop();
                return REDISPLAY;

            case '\030': /* Ctrl-X */
            case '\003': /* Ctrl-C */
                ioP->unmark();
                return ABORT;

            default:
                add_char(key);
        }
    }
}

*****
IntField
*****
IntField::IntField(InOut *ioP, int inx, int iny, int size, int *val) : Field(ioP, inx, iny, size), value(val) {
    if (value==NULL) {
        value = new int;

```



```

}
int LongField::display_field(int use_underscores) {
    int firstchar;
    #if USE_ALLOCA
        char *buf = (char *)alloca(len+1);
    #else
        char *buf = new char[len+1];
    #endif
    if (pos)
        firstchar = format_long(buf, len+1, *value);
    else
        firstchar = len;
    position(ioP, x, y);
    if (use_underscores) {
        ioP->write(underscores, firstchar);
        ioP->write(buf+firstchar, len-firstchar);
    } else {
        ioP->write(buf, len);
    }
    return 0;
}
int LongField::finalize_field() {
    changed = 0;
    string[pos] = 0;
    if (value != NULL)
        *value = atoi(string);
    return 0;
}

/*****
MoneyField
*****/
MoneyField::MoneyField(InOut *ioP, int inx, int iny, int size, double *val) : Field(ioP, inx, iny, size),
value(val) {
    seen_dollar = seen_sign = seen_dot = seen_digit = 0;
    if (value == NULL) {
        value = new double;
        need_free = 1;
    }
}
MoneyField::MoneyField(InOut *ioP, int size, double *val) : Field(ioP, size), value(val) {
    seen_dollar = seen_sign = seen_dot = seen_digit = 0;
    if (value == NULL) {
        value = new double;
        need_free = 1;
    }
}
MoneyField::~MoneyField() {
    if (need_free)
        delete value;
}
int MoneyField::add_char(int key) {
    do {
        if (pos >= len)
            break;
        if (key == '$') {
            if (!(pos == 0 || (pos == 1 && seen_sign))) break;
            seen_dollar = 1;
        } else if (key == '.') {
            if (!(pos == 0 || (pos == 1 && seen_dollar))) break;
            seen_sign = 1;
        } else if (key == ':') {
            if (seen_dot) break;
            seen_dot = 1;
        } else if (isdigit(key))
            break;
        if (seen_dot) {
            if (seen_dot >= 4)
                break;
            seen_dot++;
        }
        changed = 1;
        string[pos] = key;
        ioP->write(&string[pos++], 1);
        return 0;
    } while (0);
    ioP->write("a", 1);
    return 1;
}
int MoneyField::backspace() {
    ioP->write("\b\b", 3);
    changed = 1;
    pos--;
    if (seen_dot)
        seen_dot--;
    if (string[pos] == '.')
        seen_sign = 0;
    if (string[pos] == '$')
        seen_dollar = 0;
    if (string[pos] == ':')
        seen_dot = 0;
    return 0;
}
int MoneyField::display_field(int use_underscores) {
    int firstchar;
    #if USE_ALLOCA
        char *buf = (char *)alloca(len+1);
    #else
        char *buf = new char[len+1];
}
#endif
endif
if (pos)
    firstchar = format_money(buf, len+1, *value);
else
    firstchar = len;
position(ioP, x, y);
if (use_underscores) {
    ioP->write(underscores, firstchar);
    ioP->write(buf+firstchar, len-firstchar);
} else {
    ioP->write(buf, len);
}
return 0;
}
int MoneyField::finalize_field() {
    changed = 0;
    string[pos] = 0;
    if (value != NULL) {
        *value = atof(string + seen_dollar + seen_sign);
        if (seen_sign)
            *value = -*value;
    }
    return 0;
}
int MoneyField::reset() {
    Field::reset();
    seen_dollar = seen_sign = seen_dot = seen_digit = 0;
    return 0;
}

/*****
TextField
*****/
TextField::TextField(InOut *ioP, int inx, int iny, int size, char *str) : Field(ioP, inx, iny, size, str) {
    value = TextField::string;
}
TextField::TextField(InOut *ioP, int size, char *str) : Field(ioP, size, str) {
    value = TextField::string;
}
int TextField::add_char(int key) {
    if (pos >= len || (isalnum(key) && key != ' ' && key != '!')) {
        ioP->write("a", 1);
        return 1;
    }
    changed = 1;
    string[pos] = key;
    ioP->write(&string[pos++], 1);
    return 0;
}

                                         field.h

/* (C)1997 IBM Corporation */
#if !defined(INCLUDE_FIELD_H)
#define INCLUDE_FIELD_H

#include "inout.h"

class Field {
public:
    enum return_codes { INVALID, ENTER, NEXT_FIELD, PREV_FIELD, ABORT, REDISPLAY };
    InOut *ioP;
    int x, y;
    const int len;
    int pos;
    int changed;
    int need_redisplay;
    char *string;
    int (*ok_func)(void *data);
    int need_free;
    int need_free_string;
    void *ok_data;
    Field(InOut *ioP, int size, char *string=NULL);
    Field(InOut *ioP, int x, int y, int size, char *string=NULL);
    virtual ~Field();
    virtual int get_field (int need_pos=1);
    int get_key ();
    virtual int backspace();
    virtual int reset();
    virtual int start_position();
    virtual int add_char(int key);
    virtual int display_field(int use_underscores=0);
    virtual int finalize_field();

    class Error {
        enum { USER_ABORT };
    };
};

class Int8Field : public Field {
public:
    unsigned char *value;
    int add_char(int key);
    int display_field(int use_underscores=0);
    int finalize_field();
}

```

```

Int8Field(InOut *ioP, int x, int y, int size, unsigned char *value=NULL);
Int8Field(InOut *ioP, int size, unsigned char *value=NULL);
virtual ~Int8Field();
};

class ShortField : public Field {
public:
short *value;
int add_char(int key);
int display_field(int use_underscores=0);
int finalize_field();

ShortField(InOut *ioP, int x, int y, int size, short *value=NULL);
ShortField(InOut *ioP, int size, short *value=NULL);
virtual ~ShortField();
};

class IntField : public Field {
public:
int *value;
int add_char(int key);
int display_field(int use_underscores=0);
int finalize_field();

IntField(InOut *ioP, int x, int y, int size, int *value=NULL);
IntField(InOut *ioP, int size, int *value=NULL);
virtual ~IntField();
};

class LongField : public Field {
public:
long *value;
int add_char(int key);
int display_field(int use_underscores=0);
int finalize_field();

LongField(InOut *ioP, int x, int y, int size, long *value=NULL);
LongField(InOut *ioP, int size, long *value=NULL);
virtual ~LongField();
};

class MoneyField : public Field {
public:
int seen_dollar, seen_sign, seen_dot, seen_digit;
double *value;
int add_char(int key);
int reset();
int backspace();
int display_field(int use_underscores=0);
int finalize_field();
MoneyField(InOut *ioP, int x, int y, int size, double *value=NULL);
MoneyField(InOut *ioP, int size, double *value=NULL);
virtual ~MoneyField();
};

class TextField : public Field {
public:
char *value;
int add_char(int key);
TextField(InOut *ioP, int x, int y, int size, char *value=NULL);
TextField(InOut *ioP, int size, char *value=NULL);
};

Field *genfield(InOut *ioP, int x, int y, int len, int *ptr);
Field *genfield(InOut *ioP, int x, int y, int len, short *ptr);
Field *genfield(InOut *ioP, int x, int y, int len, long *ptr);
Field *genfield(InOut *ioP, int x, int y, int len, char *ptr);
Field *genfield(InOut *ioP, int x, int y, int len, unsigned char *ptr);
Field *genfield(InOut *ioP, int x, int y, int len, double *ptr);

#endif /* INCLUDE_FIELD_H */

```

## format.C

```

/* (C)1997 IBM Corporation */
#include <string.h>
#include <math.h>

int format_char(char *buf, int size, char val) {
int neg, pos;
pos = size;
buf[--pos] = 0;
if (val == 0 && pos > 0) {
buf[--pos] = '0';
neg = 0;
} else {
neg = (val < 0) ? 1 : 0;
if (neg) val = -val;
while (val && pos > 0) {
buf[--pos] = (val % 10) + '0';
val /= 10;
}
}
/* Too long */
if (!pos && (val || neg)) {
memset(buf, '*', size);
}
}

```

```

return -1;
}
if (neg)
buf[--pos] = '-';
if (pos)
memset(buf, ' ', pos);
return pos;
}

int format_short(char *buf, int size, short val) {
int neg, pos;
pos = size;
buf[--pos] = 0;
if (val == 0 && pos > 0) {
buf[--pos] = '0';
neg = 0;
} else {
neg = (val < 0) ? 1 : 0;
if (neg) val = -val;
while (val && pos > 0) {
buf[--pos] = (val % 10) + '0';
val /= 10;
}
}
/* Too long */
if (!pos && (val || neg)) {
memset(buf, '*', size);
return -1;
}
if (neg)
buf[--pos] = '-';
if (pos)
memset(buf, ' ', pos);
return pos;
}

int format_int(char *buf, int size, int val) {
int neg, pos;
pos = size;
buf[--pos] = 0;
if (val == 0 && pos > 0) {
buf[--pos] = '0';
neg = 0;
} else {
neg = (val < 0) ? 1 : 0;
if (neg) val = -val;
while (val && pos > 0) {
buf[--pos] = (val % 10) + '0';
val /= 10;
}
}
/* Too long */
if (!pos && (val || neg)) {
memset(buf, '*', size);
return -1;
}
if (neg)
buf[--pos] = '-';
if (pos)
memset(buf, ' ', pos);
return pos;
}

int format_long(char *buf, int size, long val) {
int neg, pos;
pos = size;
buf[--pos] = 0;
if (val == 0 && pos > 0) {
buf[--pos] = '0';
neg = 0;
} else {
neg = (val < 0) ? 1 : 0;
if (neg) val = -val;
while (val && pos > 0) {
buf[--pos] = (val % 10) + '0';
val /= 10;
}
}
/* Too long */
if (!pos && (val || neg)) {
memset(buf, '*', size);
return -1;
}
if (neg)
buf[--pos] = '-';
if (pos)
memset(buf, ' ', pos);
return pos;
}

int format_float(char *buf, int size, int dec, double val) {
static double pow10[] = { 1, 10, 100, 1000, 10000, 100000, 1000000 };
int neg, pos;
pos = size;
buf[--pos] = 0;
#ifdef WIN32
val = rint(val * pow10[dec]);
#else /* there is no rint on NT. Use floor instead */
val = floor(val * pow10[dec] + 0.5);
#endif
neg = (val < 0) ? 1 : 0;
if (neg) val = -val;
}

```

```

while (val >= 1 && pos > 0) {
    if (!dec--) {
        buf[--pos] = '-';
        continue;
    }
    buf[--pos] = (int)fmod(val, 10) + '0';
    val /= 10;
}
if (dec >= 0) {
    while (dec >= 0 && pos > 0) {
        if (!dec--) {
            buf[--pos] = '-';
        } else {
            buf[--pos] = '0';
        }
    }
    if (pos > 0)
        buf[--pos] = '0';
}
/* Too long */
if (!pos && (val >= 1 || neg)) {
    memset(buf, '*', size);
    return -1;
}
if (neg)
    buf[--pos] = '-';
if (pos)
    memset(buf, ' ', pos);
return pos;
}

int format_money(char *buf, int size, double val) {
    int pos;
    pos = format_float(buf, size, 2, val);
    if (pos > 0)
        buf[--pos] = 'S';
    return pos;
}

int format_date(char *buf, int size, unsigned char* val) {
    memcpy(buf, val, size);
    buf[size]=0;
    return 0;
}

int format_phone(char *buf, int size, unsigned char *phone) {
    buf[0] = phone[0];
    buf[1] = phone[1];
    buf[2] = phone[2];
    buf[3] = phone[3];
    buf[4] = phone[4];
    buf[5] = phone[5];
    buf[6] = '-';
    buf[7] = phone[6];
    buf[8] = phone[7];
    buf[9] = phone[8];
    buf[10] = '-';
    buf[11] = phone[9];
    buf[12] = phone[10];
    buf[13] = phone[11];
    buf[14] = '-';
    buf[15] = phone[12];
    buf[16] = phone[13];
    buf[17] = phone[14];
    buf[18] = phone[15];
    buf[19] = '\0';
    return size;
}

int format_zip(char *buf, int size, unsigned char *zip) {
    buf[0] = zip[0];
    buf[1] = zip[1];
    buf[2] = zip[2];
    buf[3] = zip[3];
    buf[4] = zip[4];
    buf[5] = '-';
    buf[6] = zip[5];
    buf[7] = zip[6];
    buf[8] = zip[7];
    buf[9] = zip[8];
    buf[10] = '\0';
    return size;
}

/* (C)1997 IBM Corporation */
#ifndef INCLUDE_FORMAT_H
#define INCLUDE_FORMAT_H

int format_char(char *buf, int size, char val);
int format_int(char *buf, int size, int val);
int format_long(char *buf, int size, long val);
int format_short(char *buf, int size, short val);
int format_float(char *buf, int size, int dec, double val);
int format_money(char *buf, int size, double val);
int format_date(char *buf, int size, unsigned char *val);

```

## format.h

```

int format_phone(char *buf, int size, unsigned char *phone);
int format_zip(char *buf, int size, unsigned char *zip);

```

```

#endif /* INCLUDE_FORMAT_H */

```

## get\_local\_time.c

```

/*
 *      get_local_time.c
 *
 * $Revision: 1.2 $
 * $Date: 1998/11/06 21:42:02 $
 * $Log: $
 *
 * $TALog: get_local_time.c,v $
 * Revision 1.2 1998/11/06 21:42:02 dongfeng
 * - Add makefile-nt
 *
 * - cast cur_t from double to long to get rid of some warnings.
 * [from r1.1 by delta dongfeng-23677-TPCC-new-directory-structures, r1.2]
 *
 * Revision 1.1 1998/11/06 21:10:12 dongfeng
 * - Move all files common to client and server to tpcc/common
 * directory
 * [added by delta dongfeng-23677-TPCC-new-directory-structures, r1.1]
 *
 * Revision 1.2 1998/10/22 19:18:33 dongfeng
 * [added by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.2]
 *
 */

#ifdef WIN32
#include "get_local_time.h"
#else
#include <sys/time.h>
#endif
#include <stdio.h>

#define Li2Double(x) ((double)((x).HighPart) * 4.294967296E9 + (double)((x).LowPart))

#ifdef WIN32
LARGE_INTEGER pFreq;
double sFreq;

get_time_init()
{
    QueryPerformanceFrequency(&pFreq);
    sFreq=Li2Double(pFreq);
}

get_local_time(struct timeval *timeP)
{
    double cur_t;
    LARGE_INTEGER counter;

    QueryPerformanceCounter(&counter);
    cur_t = Li2Double(counter) / sFreq;
    timeP->tv_sec = (long)cur_t;
    timeP->tv_usec = ((long)cur_t - timeP->tv_sec) * 1000000;
}

int gettimeofday(struct timeval *curTimeP, struct timezone *timezoneP)
{
    get_local_time(curTimeP);
    return 1;
}

#else
get_time_init()
{
}

get_local_time(struct timeval *timeP)
{
    struct timezone tz;

    gettimeofday(timeP, &tz);
}

#endif

#ifdef _GET_LOCAL_TIME_H_
#define _GET_LOCAL_TIME_H_

#ifdef WIN32
#include <windows.h>
#include <winsock.h>

/*
 * gettimeofday is not available in the Microsoft C/C++ Run Time
 * and the Win32 API.
 */

```

## get\_local\_time.h



```

*
* It is not used and just for unix compatibility.
*/
struct timezone {
    char a;
};

get_time_init();

int gettimeofday(struct timeval *curTimeP, struct timezone *timezoneP);

#endif

#endif

                inout.C

/* (C)1997 IBM Corporation */
#include <string.h>
#ifdef WIN32
#include <strings.h>
#endif
#include <unistd.h>
#include <stdlib.h>
#ifdef WIN32
#include <io.h>
#include <winsock.h>
#endif
#include <stdio.h>
#include <ctype.h>
#include <errno.h>

#include "screen.h"

extern char *sys_errlist[];

#if 1
void InOut::write(const void *buf, size_t size) {
    if (IOError) return;
    debug("write(%s, %d)\n", size, size, buf, size);
    output.queue(buf, size);
    if (!Hold && input.len() == 0) { /* Don't write anything until there is no input */
        flush();
    }
}

ssize_t InOut::read(void *buf, size_t size) {
    int rc;
    if (IOError) return(0);
    while (input.len() < size) {
#ifdef WIN32
        rc = recv(in_fd, (char *)input.ptr(), input.free(), 0);
#else
        rc = ::read(in_fd, input.ptr(), input.free());
#endif
        debug("::read(%s, %d) = %d\n", rc, rc, input.ptr(), input.free(), rc);
        if (inlog) {
            fwrite(input.ptr(), rc, 1, inlog);
            fflush(inlog);
        }
        if (rc > 0) {
            input.queue(rc);
        } else if (rc <= 0) {
            IOError = 1;
            return(0);
        }
    }
    memcpy(buf, input.ptr(), size);
    input.dequeue(size);
    debug("read(%s, %d) = %d\n", size, size, buf, size, size);
    return size;
}

#else
void InOut::write(const void *buf, size_t size) {
    debug("write(%s, %d)\n", buf, size);
#ifdef WIN32
    send(out_fd, (char *)buf, size, 0);
#else
    ::write(out_fd, buf, size);
#endif
}

#endif

ssize_t InOut::read(void *buf, size_t size) {
    int rc;
    rc = ::read(in_fd, buf, size);
    debug("read(%s, %d) = %d\n", buf, size, rc);
    return rc;
}

#endif

void InOut::flush() {
    debug("flush()\n");
    Hold = 0;
    if (IOError) return;
    while (output.len()) {
        debug("::write(%s, %d)\n", output.len(), output.len(), output.ptr(), output.len());
#ifdef WIN32
        int rc = send(out_fd, (char *)output.ptr(), output.len(), 0);

```

```

#else
        int rc = ::write(out_fd, output.ptr(), output.len());
#endif
    }
}

void InOut::write(const void *buf) {
    write(buf, strlen((const char *)buf));
}

InOut::InOut(int in, int out) : input(256), output(2048) {
#ifdef WIN32
    struct termios buf;
#endif

#ifdef DEBUG
    {
        char buf[256];
        sprintf(buf, "logs/debug.%d", getpid());
        debugfile = fopen(buf, "w");
        sprintf(buf, "logs/in.%d", getpid());
        inlog = fopen(buf, "w");
        sprintf(buf, "logs/out.%d", getpid());
        outlog = fopen(buf, "w");
    }
#endif

    int rc;
    Hold = 0;
    debugfile = inlog = outlog = (FILE *)0;
    IOError = 0;

    in_fd = in;
    if (out < 0)
        out_fd = in;
    else
        out_fd = out;
#ifdef WIN32
    if ((rc = tcgetattr(in_fd, &save_term)) < 0) {
        return;
    }

    buf = save_term;

    buf.c_lflag &= ~(ECHO | ICANON); /* echo off, canonical mode off */

    buf.c_cc[VMIN] = 1; /* Case B: 1 byte at a time, no timer */
    buf.c_cc[VTIME] = 0;

    err_printf("echo off - tcsetattr on %d\n", in_fd);
    if (tcsetattr(in_fd, TCSAFLUSH, &buf) < 0)
        return;
#endif

    InOut::~InOut() {
#ifdef WIN32
        return;
#else
        if (tcsetattr(in_fd, TCSAFLUSH, &save_term) < 0)
            return;
#endif
    }

    /* (C)1997 IBM Corporation */

#ifdef INOUT_H
#define INOUT_H
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#ifdef WIN32
#include <termios.h>
#endif
#include <stdarg.h>
#include <string.h>

#include "tpcc.h"

/* This is for a VT100 */
#if 1
#define ESC "\033"
#define ESCc "\033"

```

## inout.h

```

#else
#define ESCc '^'
#define ESC "'^"
#endif

#define TRIGGER "021"
#define TRIGGERc "021"

extern "C" err_printf(...);

#define POS(x,y) ESC "[#y ";#x "H"
#define CLEAR_EOS ESC "[J"

#ifdef WIN32
typedef int ssize_t;
#endif

class InOut {
private:
    class Buffer {
    private:
        int BufSize;
        enum { NUMMARKS=8 };
        char *buffer;
        int marks[NUMMARKS];

    public:
        int Pos;
        int Start;

        int num_marks;
        Buffer(int size) {
            BufSize = size;
            buffer = new char [BufSize];
            Pos = Start = 0;
            num_marks = 0;
        }
        int pos() { return Pos; };
        void pos(int P) { Pos = P; };
        int start() { return Start; };
        void start(int S) { Start = S; };
        int len() { return Pos-Start; };
        int free() { return BufSize-Pos-1; };
        void *ptr() { return &buffer[Start]; };
        int lastmark() { if (num_marks) return marks[num_marks-1]; return 999; };

        void mark() {
            if (num_marks < NUMMARKS)
                marks[num_marks++] = Pos;
            else {
                fprintf(stderr, "Buffer mark overflow\n");
                exit (1);
            }
        }
        void unmark() {
            if (num_marks <= 0)
                return;
            num_marks--;
        }
        void pop() {
            if (num_marks <= 0)
                return;
            if (marks[num_marks-1] >= Start) {
                Pos=marks[--num_marks];
            } else {
                num_marks=0;
            }
        }
        void queue(int size) {
            Pos += size;
        }
        void queue(const void *buf, int size) {
            /* If this is too big see if we can move what we have over */
            if (size+Pos >= BufSize) {
                if (size + len() >= BufSize) {
                    fprintf(stderr, "Buffer overflow\n");
                    exit (1);
                }
                /* This requires memcopy to be "safe" */
                if (Start + len() >= BufSize) {
                    fprintf(stderr, "Strange Error: Start %d + len %d >= size %d\n",
                        Start, len(), BufSize);
                    exit(1);
                }
                memcopy(buffer, &buffer[Start], len());
                Pos -= Start;

                /* Fix up our marks*/
                int count = 0;
                for (int i = 0; i < num_marks; i++) {
                    if (marks[i] - Start >= 0)
                        marks[count++] = marks[i] - Start;
                }
                num_marks = count;
                Start = 0;
            }
            memcopy(&buffer[Pos], buf, size);
            Pos += size;
        }
        void dequeue(int size) {
            Start += size;

```

```

            if (Start >= Pos) {
                /* Fix up our marks*/
                int count = 0;
                for (int i = 0; i < num_marks; i++) {
                    if (marks[i] - Start >= 0)
                        marks[count++] = marks[i] - Start;
                }
                num_marks = count;

                Start = Pos = 0;
            }
        };
        int in_fd, out_fd;
        int Hold;
#ifdef WIN32
        struct termios save_term;
#endif
    public:
        Buffer input;
        Buffer output;
        FILE *debugfile;
        FILE *inlog, *outlog;
        int IOError;
        ssize_t read(void *buf, size_t size);
        void write(const void *buf, size_t size);
        void write(const void *buf);
        void flush();
        void mark() { debug("mark()\n"); output.mark(); };
        void unmark() { debug("unmark()\n"); output.unmark(); };
        void pop() { debug("pop()\n"); output.pop(); };
        void hold() { debug("hold()\n"); Hold = 1; };
#ifdef DEBUG
        void debug(char *fmt, ...) {
            va_list args;

            fprintf(debugfile, "Start=%2d, Pos=%2d, Marks=%2d(%03d): ", output.Start, output.Pos,
                output.num_marks, output.lastmark());
            va_start(args,fmt);
            vfprintf (debugfile, fmt, args);
            va_end (args);
            ::flush(debugfile);
        }
#endif
        void debug(char *fmt, ...) {};
    };
    InOut(int in=0, int out=1);
    ~InOut();
};

extern char const * const blanks;
extern char const * const underscores;
extern char const * const backspaces;

int format_int(char *buf, int size, int val);
int format_float(char *buf, int size, int dec, double val);
int format_money(char *buf, int size, double val);

#endif /* INOUT_H */

/*
 *      mon_client.c
 *
 * $Revision: 1.27 $
 * $Date: 1999/05/26 16:29:52 $
 * $Log: $
 *
 * * STALog: mon_client.c.v $
 * * Revision 1.27 1999/05/26 16:29:52 wenjian
 * * Sync with Austin code, and sync code for Oracle DB and SQL server.
 * * [from r1.26 by delta wenjian-24433-TPCC-clean-up-and-update, r1.2]
 *
 * * Revision 1.26 1999/05/06 21:28:26 oz
 * * - Removed all the .. from the includes
 * * - Added -I. to the makefiles instead
 * * - Moved all the thread related code and connection
 * *   selection to serverMon.c
 * * [from r1.16 by delta oz-24309-TPCC-add-oracle8.1-code, r1.5]
 *
 * * Revision 1.16 1999/01/29 20:16:33 wenjian
 * * - Rename client_init to init_encina_client because we have another
 * *   client_init in screen/client.C
 * * - Add code to read StatsFrequency from .tpccrc (UNIX only)
 * * [from r1.15 by delta wenjian-23787-TPCC-integrate-code-for-AIX-and-NT, r1.7]
 *
 * * Revision 1.15 1999/01/12 20:52:55 wenjian
 * * Call initialization function to create the shared file mapping between
 * * it and the corresponding dll.
 * * [from r1.14 by delta wenjian-23856-TPCC-integrate-with-NT-performance-monitor, r1.1]
 *
 * * Revision 1.14 1998/12/28 20:07:12 wenjian
 * * - Change client_info to a pointer pClientInfo for flexibility.
 * * [from r1.13 by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.5]
 *

```

## mon\_client.c

\* Revision 1.13 1998/12/16 17:17:41 wenjian  
 \* - Change (iStatsFrequency <= 1) to (iStatsFrequency < 1) in pre\_rpc.  
 \* [from r1.12 by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.4]  
 \*

\* Revision 1.12 1998/12/14 20:27:54 wenjian  
 \* Made corresponding changes due to data structure change of tran\_info\_t.  
 \* [from r1.11 by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.3]  
 \*

\* Revision 1.11 1998/12/11 16:14:19 wenjian  
 \* Add code for checking statistic data in a single variable and collecting  
 \* statistic data based on iStatsFrequency.  
 \*

\* - Add code to store statistic data in a single var  
 \* - Collect statistic data once every iStatsFrequency transactions  
 \* [from r1.10 by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.1]  
 \*

\* Revision 1.10 1998/12/08 23:03:49 wenjian  
 \* Add (or rename) Makefile for each platform (AIX and NT). Reorganize the  
 \* files a little bit.  
 \*

\* - Define variable iStatsFrequency for AIX  
 \* [from r1.9 by delta wenjian-23787-TPCC-integrate-code-for-AIX-and-NT, r1.1]  
 \*

\* Revision 1.9 1998/12/08 18:55:19 wenjian  
 \* In pre\_rpc, set headerP->stats to iStatsFrequency  
 \* [from r1.8 by delta wenjian-23785-TPCC-pass-statsFrequency-from-client-to-server, r1.1]  
 \*

\* Revision 1.8 1998/12/07 20:04:12 wenjian  
 \* Clean up  
 \* [from r1.7 by delta wenjian-23742-TPCC-update-with-Raleigh-code, r1.2]  
 \*

\* Revision 1.7 1998/11/24 21:45:59 wenjian  
 \* - Add #ifdef MULTIPLE\_INTERFACE  
 \* - Check if we need to collect statistics for response time  
 \* - Do mutex\_lock for terminal\_context\_init  
 \* [from r1.6 by delta wenjian-23742-TPCC-update-with-Raleigh-code, r1.1]  
 \*

\* Revision 1.6 1998/11/09 16:59:38 wenjian  
 \* In this revision, most of the changes are related to the directory of header  
 \* files after directory reorganization. Other changes include adding or removing  
 \* files to put them in the right directories. Makefiles are written for NT  
 \* platform so that nmake is working on NT now. Need a top level Makefile for all  
 \* the directories.  
 \* [from r1.5 by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.2]  
 \*

\* Revision 1.5 1998/11/09 14:48:16 wenjian  
 \* In an effort to make a new directory structure for TPCC, this delta  
 \* creates two directories: tpcc/client and tpcc/server. All the files  
 \* for this revision are copied from tpcc/sp-tpcc without any change.  
 \* Further change may be needed for some files due to the change of  
 \* the directory structure.  
 \* [added by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.1]  
 \*

\* Revision 1.43 1998/11/06 16:10:55 wenjian  
 \* - Minor change to reduce the print statement  
 \* [from r1.42 by delta wenjian-23646-TPCC-clean-up-source-code, r1.1]  
 \*

\* Revision 1.42 1998/10/27 14:57:51 dongfeng  
 \* Change enc\_status to a data structure that has fields:  
 \* - Status code  
 \* - Line Number  
 \* - File Name  
 \* - Encina Error Code  
 \* - Error Msg  
 \* Remove statusMsgs in web\_tpcc.c  
 \* [from r1.41 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.6]  
 \*

\* Revision 1.41 1998/10/26 14:41:34 dongfeng  
 \* Add Init command in web client so when something bad happens during  
 \* initialization web client sends back error information and allows  
 \* reinitialization instead of killing IIS server.  
 \*

\* Define Macro CHK\_STATUS instead of using #ifdef  
 \* [from r1.40 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.3]  
 \*

\* Revision 1.40 1998/10/22 21:24:11 wenjian  
 \* [merge of changes from 1.23 to 1.39 into 1.38]  
 \*

\* Revision 1.39 1998/10/22 19:18:34 dongfeng  
 \* [from r1.37 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.2]  
 \*

\* Revision 1.37 1998/10/08 14:18:01 dongfeng  
 \* Add codes for doing web-based tpcc.  
 \* [from r1.23 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.1]  
 \*

\* Revision 1.38 1998/10/08 18:03:01 gerstl  
 \* Changes to allow configurations where some servers only service  
 \* specific transaction types. Split transaction interfaces by type.  
 \* [from r1.36 by delta gerstl-23515-TPCC-allow-separate-online-transaction-interfaces, r1.1]  
 \*

\* Revision 1.36 1998/10/07 15:14:22 gerstl  
 \* [merge of changes from 1.26 to 1.31 into 1.34]  
 \*

\* Revision 1.31 1998/09/04 19:17:56 wenjian  
 \* Remove log\_file\_handle and related code.  
 \* [from r1.29 by delta wenjian-23183-TPCC-sync-AIX-code-with-Austin, r1.5]  
 \*

\* Revision 1.29 1998/08/28 18:30:00 wenjian  
 \* This delta sync the TPCC code with Austin.

\*  
 \* Remove UNCOND\_EVENT in CALLTPCC, pre\_rpc and post\_rpc.  
 \* [from r1.26 by delta wenjian-23183-TPCC-sync-AIX-code-with-Austin, r1.1]  
 \*

\* Revision 1.34 1998/09/26 10:56:26 oz  
 \* - renamed thread\_init and thread\_done to clnt\_thread\_init and  
 \* clnt\_thread\_done respectively because of name conflicts on AIX4.3  
 \* [from r1.26 by delta oz-23339-TPCC-update-for-NT, r1.2]  
 \*

\* Revision 1.26 1998/08/18 14:38:41 wenjian  
 \* Remove adl.h from this file since it is not ported on NT. Use corresponding  
 \* rpc protect levels and authz levels to replace ADL\_.... macros defined in  
 \* adl.h  
 \* [from r1.23 by delta wenjian-21750-TPCC-changes-for-porting-on-NT, r1.4]  
 \*

\* Revision 1.23 1998/06/17 15:05:45 wenjian  
 \* Somehow, read and write didn't work for socket on NT, although they  
 \* are supposed to work. As a work-around way, use recv and send for  
 \* NT in this revision. We may change them back if the problem is gone.  
 \*

\* Define SKIP\_RPC and add code to do tests without calling DCE.  
 \* This addition is for test purpose only.  
 \* [from r1.22 by delta wenjian-21750-TPCC-changes-for-porting-on-NT, r1.2]  
 \*

\* Revision 1.22 1998/02/17 22:13:49 wenjian  
 \* [merge of changes from 1.19 to 1.20 into 1.21]  
 \*

\* Revision 1.20 1998/02/17 16:04:42 oz  
 \* - Split the login into two parts to allow for special logins  
 \* - If the warehouse ID is 0, this is a special login to  
 \* query the client for status  
 \*

\* - Keep track of threads that have been initialized and also  
 \* threads that are done.  
 \* [from r1.19 by delta oz-21864-TPCC-split-client-login-screen, r1.1]  
 \*

\* Revision 1.21 1998/02/17 22:07:03 wenjian  
 \* Minor changes for NT  
 \* [from r1.19 by delta wenjian-21750-TPCC-changes-for-porting-on-NT, r1.1]  
 \*

\* Revision 1.19 1998/01/26 20:37:35 oz  
 \* - Remove all the code associated with explicit binding  
 \*

\* - Removed GET\_SERVER\_INDEX  
 \* - Removed bindingType  
 \* - Removed explicit binding from CALLTPCC  
 \* - Removed calls to cancel\_all\_reservations and to init\_handles  
 \* [from r1.18 by delta oz-21697-TPCC-remove-explicit-binding-code, r1.1]  
 \*

\* Revision 1.18 1998/01/26 16:43:32 oz  
 \* - Removed the code for collecting stats in the client  
 \* and dumping them before exit.  
 \*

\* - Removed pre\_rpc\_stats and post\_rpc\_stats  
 \* - Removed code to write the stats out  
 \* [from r1.17 by delta oz-21691-TPCC-remove-client-stats-code, r1.1]  
 \*

\* Revision 1.17 1998/01/26 16:19:23 oz  
 \* - moved all the code pertaining to the background  
 \* thread to its own file and all the data structures  
 \* to client\_utils.h  
 \* [from r1.16 by delta oz-21689-TPCC-move-client-bg-thread-to-separate-file, r1.1]  
 \*

\* Revision 1.16 1998/01/26 15:33:32 oz  
 \* - call impTPCCNOInfo to make sure there is a server out there  
 \* [from r1.15 by delta oz-21671-TPCC-merge-online-transaction-interfaces, r1.2]  
 \*

\* Revision 1.15 1998/01/23 21:58:51 oz  
 \* - In order to simplify the Encina TPCC code: Merge the four  
 \* online transactions into 1 interface  
 \* - Moved all the scripts to a scripts subdirectory  
 \* - Removed unused files  
 \* [from r1.14 by delta oz-21671-TPCC-merge-online-transaction-interfaces, r1.1]  
 \*

\* Revision 1.14 1998/01/23 15:07:53 oz  
 \* - Updated the SP TPCC directory to the latest files used  
 \* during the SP tpcc audit.  
 \* [from r1.13 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]  
 \*

\*  
 \*  
 \*  
 \*/  
 #include "common/get\_local\_time.h"  
 #include <stdio.h>  
 #include <stdlib.h>  
 #include <string.h>  
 #include <stdarg.h>  
 #include <time.h>  
 #if defined (solaris)  
 #include <dce/pthread.h>  
 #else /\* solaris \*/  
 #include <pthread.h>  
 #endif /\* solaris \*/  
 #include <tpm/mon/mon.h>  
 #include <utils/trace.h>  
 #include "common/delivery.h"  
 #ifdef MULTIPLE\_INTERFACE  
 #include "common/neworder.h"

```

#include "common/payment.h"
#include "common/stocklevel.h"
#include "common/orderstatus.h"
#else
#include "common/tpcc_trans.h"
#endif
#include "common/utilities.h"
#include "client_utils.h"
#include "common/do_tpcc.h"
#include "client.h"
#include "encina_client.h"

#if 0
#define SKIP_RPC
#endif

extern void start_bg_debug_thread(void);
extern total_tran_count_t *perfCntDataInit();

#define MAX_CONSECUTIVE_ERRORS 20

static void read_mon_environment(void);
static void client_trace(char *comp, int value, int add);
static void dump_pa_ring_buffer(tpcc_handle_t pa_handle);

extern int warehouse_offset;

unsigned32 client_authnLevel;
unsigned32 client_authzSvc;

char *cellName;
int envRetrieval = 0;

static total_tran_count_t total_counts; /* counts of transactions over
                                         * the entire test
                                         */

#ifndef WEB_TPCC_CLIENT
#undef CHK_STATUS
#define CHK_STATUS(status, val, a) if(status) {exit_program(status);}
MUTEX_T init_lock;
static int iStatsFrequency = 1;
#else
extern enc_status_t enc_status;
CRITICAL_SECTION init_lock;
extern int iStatsFrequency;
#endif
int info_list_len = 0;
thread_info_t **info_list = NULL; /* List of all the thread info
                                   * structures. This can be used
                                   * upon exit to cancel all the
                                   * reservations
                                   */

total_tran_count_t *pClientInfo=NULL; /* keep stats for the client process */
static num_active_threads = 0;

#define NewOrder_code NEWO_TRANS
#define Payment_code PAYMENT_TRANS
#define OrderStatus_code ORDER_STAT_TRANS
#define Delivery_code DELIVERY_TRANS
#define StockLevel_code STOCK_TRANS

extern int useSecurity;

#define INT_ENV_VALUE(var, default) \
(var = getenv(#var) ? atoi(getenv(#var)) : default)

#define PRE_RPC_WORK(contextP, dataP, tran, sub_tran) \
if (contextP != NULL) \
pre_rpc(contextP, &(dataP)->header, tran, sub_tran)

#define POST_RPC_WORK(contextP, dataP, tran) \
if (contextP != NULL) \
post_rpc(contextP, &(dataP)->header, tran)

#define TIME_STR_P(infoP) (&(infoP)->last_tran)

/* CALTPCC
 * Macro to sends 1 RPC and then handles any errors.
 */
/* The macro takes the name of the RPC (e.g., NewOrder)
 * and makes the RPC by calling the appropriate function
 * (e.g., impTPCCNewOrder).
 */
#ifdef SKIP_RPC
#define CALLTPCC(name,infoP,data,tpccStatusP) \
{
struct timezone tz;
struct timespec timeP;
char tran_type[30];
strcpy(tran_type,UTIL_STRING(name));
timeP.tv_sec = 0;
timeP.tv_nsec = 190000000;
if ( strcmp(tran_type,"NewOrder")==0) \
timeP.tv_nsec = 450000000; \
if ( strcmp(tran_type,"Payment")==0) \
timeP.tv_nsec = 900000000; \
pthread_delay_np(&timeP);
gettimeofday(&TIME_STR_P(infoP)->send, &tz);
}
#else

```

```

#define CALLTPCC(name,infoP,data,tpccStatusP) \
{
struct timezone tz;
\
if (infoP) gettimeofday(&TIME_STR_P(infoP)->send, &tz);
UTIL_CONCAT(impTPCC,name)(data,tpccStatusP);
if (*(tpccStatusP)) {
char msg[100];
sprintf(msg, "TRPC error during impTPCC%s", UTIL_STRING(name)); \
(data)->header.returncode = TRPC_ERROR;
encina_error_message(msg, *(tpccStatusP)); \
} else if (((data)->header.returncode != TPCC_SUCCESS) && \
((data)->header.returncode != INVALID_NEWO)) { \
char msg[100];
sprintf(msg, "App error during impTPCC%s: ", UTIL_STRING(name)); \
encina_error_message(msg, (data)->header.returncode); \
} \
}
#endif

/*
 * pre_rpc -- For debug purposes
 */
/* Called before an RPC is made.
 * Set the state of the thread and keep track of the time the RPC is sent.
 * This is used by the Background thread to report the state of the client.
 */
static void pre_rpc(thread_info_t *thread_infoP,
data_header *headerP,
int tran_type,
int sub_tran_type)
{
tran_timing_t *curP;
struct timezone tz;

curP = &thread_infoP->last_tran;
curP->terminal = thread_infoP->thread_index;
curP->tran = tran_type;
curP->sub_tran = sub_tran_type;

if (iStatsFrequency < 1) {
headerP->stats = 0;
} else {
int num;
num = ++ (pClientInfo->tran[tran_type].num);
headerP->stats = (num % iStatsFrequency == 0) ? 1 : 0;
}
if (headerP->stats) { /* measure the time for RT */
gettimeofday(&curP->start, &tz);
headerP->start_time.sec = 0;
headerP->start_time.usec = 0;
headerP->end_time.sec = 0;
headerP->end_time.usec = 0;
}

#ifdef KEEP_TERMINAL_INFO
set_client_debug_state((void *)thread_infoP, thread_state_sent, tran_type);
#endif
}

/*
 * post_rpc
 */
/* Called when the RPC returns from the server
 */
/* Keeps track of the client response time and the server response time
 * as well as the state of the thread. This is used by the background
 * debug thread to report the state of the client
 */
static void post_rpc(thread_info_t *thread_infoP,
data_header *headerP,
int tran_type)
{
double time_diff_s, time_diff_c;
tran_timing_t *curP;
struct timezone tz;

if (!thread_infoP) return;

curP = &thread_infoP->last_tran;

curP->server = headerP->dtype; /* The server sets this by convention */
if (headerP->stats) {
curP->svr_start.tv_sec = headerP->start_time.sec;
curP->svr_start.tv_usec = headerP->start_time.usec;
curP->svr_done.tv_sec = headerP->end_time.sec;
curP->svr_done.tv_usec = headerP->end_time.usec;

gettimeofday(&curP->end, &tz);
}

#ifdef KEEP_TERMINAL_INFO
/* Store the info for each terminal */
thread_infoP->num_trans++;
thread_infoP->tran[tran_type].num++;
if ((headerP->returncode == TPCC_SUCCESS) ||
(headerP->returncode == INVALID_NEWO)) {
thread_infoP->consecutive_errors = 0;
curP->tran_failed = 0;
}
}

```

<pre> if (headerP-&gt;returncode == INVALID_NEWO) {     curP-&gt;sub_tran  = 0x100; } } else {     thread_infoP-&gt;tran[tran_type].errs ++;     thread_infoP-&gt;consecutive_errors++;     curP-&gt;tran_failed = 1; } }  if (headerP-&gt;stats &amp;&amp; tran_type &lt;= MAX_TRAN_TYPE &amp;&amp; tran_type &gt; 0     &amp;&amp; !curP-&gt;tran_failed) {     set_client_debug_state((void *)thread_infoP, thread_state_received, 0);      /* update total server round trip response time */     time_diff_s = time_diff_ms(&amp;(curP-&gt;svr_done), &amp;(curP-&gt;svr_start));     thread_infoP-&gt;tran[tran_type].RTtotal[1] += time_diff_s;      /* update total client round trip response time */     time_diff_c = time_diff_ms(&amp;(curP-&gt;end), &amp;(curP-&gt;start));     thread_infoP-&gt;tran[tran_type].RTtotal[0] += time_diff_c;      /* update num for the number of trans which have RT measured */     thread_infoP-&gt;tran[tran_type].RTcount ++; }  #else /* Store the info for each client.  * Note: since we don't use mutex for performance reason, pClientInfo  * may not be accurate if more than one thread work on the same  * data at a same time. But this can reduce the overhead caused  * by scanning info_list for each terminal.  */ if ((headerP-&gt;returncode == TPCC_SUCCESS)        (headerP-&gt;returncode == INVALID_NEWO)) {     curP-&gt;tran_failed = 0;     if (headerP-&gt;returncode == INVALID_NEWO) {         curP-&gt;sub_tran  = 0x100;     } } else {     pClientInfo-&gt;tran[tran_type].errs ++;     pClientInfo-&gt;errors ++;     curP-&gt;tran_failed = 1; }  if (headerP-&gt;stats &amp;&amp; tran_type &lt;= MAX_TRAN_TYPE &amp;&amp; tran_type &gt; 0     &amp;&amp; !curP-&gt;tran_failed) {     /* update total server round trip response time */     time_diff_s = time_diff_ms(&amp;(curP-&gt;svr_done), &amp;(curP-&gt;svr_start));     pClientInfo-&gt;tran[tran_type].RTtotal[1] += time_diff_s;      /* update total client round trip response time */     time_diff_c = time_diff_ms(&amp;(curP-&gt;end), &amp;(curP-&gt;start));     pClientInfo-&gt;tran[tran_type].RTtotal[0] += time_diff_c;      /* update num for the number of trans which have RT measured */     pClientInfo-&gt;tran[tran_type].RTcount ++; } #endif */ * exit_program - restores original terminal attributes before leaving the * program. */  void exit_program( err ) {     short int err;     {         if ( err )             fprintf(ERROUT, "exit_program: Error Code = %d\n", err);          MUTEX_LOCK(&amp;init_lock);         /** Cancel all the longterm reservations (if any)          ** and write out the time-stamps          **/         if (info_list &amp;&amp; (info_list_len &gt; 0)) {             int i;              for (i=0; i&lt;info_list_len; i++) {                  if (info_list[i] &amp;&amp; info_list[i]-&gt;initialized) {                     info_list[i]-&gt;initialized = 0;                 }             }         }          MUTEX_UNLOCK(&amp;init_lock);          if (logtpcc) {             fclose(logtpcc);         } else {             if (logtpcc = fopen(log_file_name, "w")) {                 fprintf(logtpcc, "ERROR: Client exiting before SYNC with error %d\n",                     err);                 fclose(logtpcc);             }         }     } }  mon_ExitClient( err ); </pre>	<pre> #ifdef WEB_TPCC_CLIENT     exit( err ); #endif }  /*  * clnt_thread_init  *  * This function must be called by each work thread  * It returns a pointer to a context that must be passed  * on calls back to this module.  * There is 1 threadInfo entry in an array for each executor thread.  * When an executor thread is started the first thing it does is call  * this clnt_thread_init function. This function creates a context for the  * thread and if longterm reservations are used this function  * initializes the pa handle.  */ void *clnt_thread_init(void) {     int thread_index;     struct timezone tz;     thread_info_t *thread_infoP;      if (iStatsFrequency &lt; 1)         return(NULL);      thread_infoP = (thread_info_t *)calloc(1, sizeof(thread_info_t));      thread_infoP-&gt;descr.state = thread_state_init;     gettimeofday(&amp;thread_infoP-&gt;descr.init, &amp;tz);     thread_infoP-&gt;initialized = 1;      MUTEX_LOCK(&amp;init_lock);     thread_index = info_list_len++;     thread_infoP-&gt;thread_index = thread_index;     thread_infoP-&gt;thread_id = get_thread_id();      num_active_threads++;     info_list =         (thread_info_t **)realloc((void *)info_list,             sizeof(thread_info_t *) * info_list_len);     info_list[thread_index] = thread_infoP;      MUTEX_UNLOCK(&amp;init_lock);      if (num_active_threads % 200 == 0)         err_printf("Thread %d Initialized (currently %d are active).\n",             thread_index, num_active_threads);      return(thread_infoP); }  /*  * clnt_thread_done  *  * Called before a thread exits.  * Perform some cleanup.  */ void clnt_thread_done(contextP) void *contextP; {     int all_done = 0;     int j;     thread_info_t *infoP = (thread_info_t *)contextP;      if (!infoP) return;      MUTEX_LOCK(&amp;init_lock);      num_active_threads--;      #if 0     err_printf("&gt; thread_done, %d active\n", num_active_threads);     #endif      set_client_debug_state((void *)infoP, thread_state_done, 0);     infoP-&gt;done = 1;      if (num_active_threads == 0) {         all_done = 1;     }      if (info_list[infoP-&gt;thread_index] != infoP) {         fprintf(ERROUT, "Strange error: expected to find %d in info_list[%d] and found %d instead\n",             infoP, infoP-&gt;thread_index,             info_list[infoP-&gt;thread_index]);     }      MUTEX_UNLOCK(&amp;init_lock);     if (all_done) {         int i;         thread_info_t **curP;      #if 0         fprintf(ERROUT, "All Done - exiting\n");     #endif } </pre>
--	--

```

MUTEX_LOCK(&init_lock);
for (i=0, curP=info_list; i<info_list_len; i++, curP++) {
    free(*curP);
}
free(info_list);
info_list = NULL;
info_list_len = 0;
MUTEX_UNLOCK(&init_lock);
#endif
exit(0);
#endif
}
}
*
* The following send_*** functions are called from the screen
* module after the transaction data is received in order to
* send the data to the server for processing.
*
*/
*
* send_new_order
* Send a new order request to the server
*/
void send_new_order(contextP, dataP)
void *contextP;
newOrder_data_t *dataP;
{
    thread_info_t *thread_context = (thread_info_t *)contextP;
    trpc_status_t trpcStatus;

    DPRINT(("New Order, w_id %d, %d orders\n", dataP->w_id, dataP->o_ol_cnt));
    PRE_RPC_WORK(thread_context, dataP, NEWO_TRANS, dataP->o_all_local = 0);
    CALLTPCC(NewOrder, thread_context, dataP, &trpcStatus);
    POST_RPC_WORK(thread_context, dataP, NEWO_TRANS);
}
*
* send_payment
* Send a payment request to the server
*/
void send_payment(contextP, dataP)
void *contextP;
payment_data_t *dataP;
{
    trpc_status_t trpcStatus;
    thread_info_t *thread_context = (thread_info_t *)contextP;

    PRE_RPC_WORK(thread_context, dataP, PAYMENT_TRANS,
        dataP->w_id != dataP->c_w_id);
    CALLTPCC(Payment, thread_context, dataP, &trpcStatus);
    POST_RPC_WORK(thread_context, dataP, PAYMENT_TRANS);
}
*
* send_order_status
* Send an order status request to the server
*/
void send_order_status(contextP, dataP)
void *contextP;
orderStatus_data_t *dataP;
{
    trpc_status_t trpcStatus;
    thread_info_t *thread_context = (thread_info_t *)contextP;

    PRE_RPC_WORK(thread_context, dataP, ORDER_STAT_TRANS, 0);
    CALLTPCC(OrderStatus, thread_context, dataP, &trpcStatus);
    POST_RPC_WORK(thread_context, dataP, ORDER_STAT_TRANS);
}
*
* send_delivery
* Send a delivery request to the server
*/
void send_delivery(contextP, dataP)
void *contextP;
delivery_data_t *dataP;
{
    trpc_status_t trpcStatus;
    thread_info_t *thread_context = (thread_info_t *)contextP;

    PRE_RPC_WORK(thread_context, dataP, DELIVERY_TRANS, 0);
    CALLTPCC(Delivery, thread_context, dataP, &trpcStatus);
    POST_RPC_WORK(thread_context, dataP, DELIVERY_TRANS);
}
*
* send_stock_level
* Send a stock level request to the server
*/
void send_stock_level(contextP, dataP)
void *contextP;
stockLevel_data_t *dataP;
{
    trpc_status_t trpcStatus;
    thread_info_t *thread_context = (thread_info_t *)contextP;

    PRE_RPC_WORK(thread_context, dataP, STOCK_TRANS, 0);
}

```

```

CALLTPCC(StockLevel, thread_context, dataP, &trpcStatus);
POST_RPC_WORK(thread_context, dataP, STOCK_TRANS);
}
}
int too_many_errors(contextP)
void *contextP;
{
    thread_info_t *thread_context = (thread_info_t *)contextP;

    return (thread_context->consecutive_errors > MAX_CONSECUTIVE_ERRORS);
}
/*
* Enroll the client:
* Perform the needed initialization
*/
void init_encina_client(user_id)
int user_id;
{
    int i;
    mon_status_t monStatus;
    char *env_str;
    char serverName[48];
    struct timezone tz;
    struct timeval a_time;
    unsigned long status;
    FILE *rcFile;

#ifdef WIN32
    get_time_init();
    pClientInfo = perfCntDataInit();
#endif
    if (pClientInfo == NULL)
        pClientInfo = malloc(sizeof(total_tran_count_t));
    memset(pClientInfo, 0, sizeof(total_tran_count_t));

    read_mon_environment();

    if (!cellName)
        CHK_STATUS(30, CELL_NAME_UNAVAILABLE,
            "ENCINA_TPM_CELL is not set!");

    MUTEX_INIT(&init_lock);

    info_list = NULL;
    info_list_len = 0;

#ifdef WEB_TPCC_CLIENT
    /* initialize iStatsFrequency */
    iStatsFrequency = 1;
    rcFile = fopen("./tpccrc", "r");
    if (rcFile != NULL) {
        char buf[100];
        int num = 1;
        while (1) { /* read the whole rcFile */
            num = fscanf(rcFile, "%s", buf);
            if (num <= 0) break;
            if (strcmp(buf, "StatsFrequency") == 0) {
                fscanf(rcFile, "%d", &iStatsFrequency);
                break;
            }
        }
    }
    err_printf("iStatsFrequency=%d\n", iStatsFrequency);
#endif

    gettimeofday(&a_time, &tz);
#ifdef WIN32
    srand(a_time.tv_sec ^ a_time.tv_usec);
#else
    srand48(a_time.tv_sec ^ a_time.tv_usec);
#endif
}
/*
* Enroll the client:
* get the necessary handles.
*/
void enroll_client(user_id)
int user_id;
{
    int i;
    mon_status_t monStatus;
    char *env_str;
    char serverName[48];
    static char *clientName = "tpcc_client";
    unsigned long status;
    static int client_enrolled = 0;

    MUTEX_LOCK(&init_lock);
    if (client_enrolled) {
        MUTEX_UNLOCK(&init_lock);
        return;
    }
    if (useSecurity) {
        client_authnLevel = rpc_c_protect_level_connect;
        client_authzSvc = rpc_c_authz_dce;
    } else {
        client_authnLevel = rpc_c_protect_level_none;
    }
}

```

```

client_authzSvc = rpc_c_authz_none;
}

if (envRetrieval == 0) {
ENCINA_CALL_RC("mon_RetrieveEnable",mon_RetrieveEnable(FALSE),status);
CHK_STATUS(status, MON_RETRIEVEENABLE_FAILED,
"mon_RetrieveEnable failed");
}

DPRINT(("Cell name: %s\n", cellName));

ENCINA_CALL_RC("mon_InitClient",mon_InitClient(clientName,cellName),
status);
CHK_STATUS(status, MON_INITCLIENT_FAILED,
"mon_InitClient failed");

DPRINT(("mon_SecuritySetDefaults-> authn %d, authz %d\n",
client_authnLevel, client_authzSvc));
ENCINA_CALL_RC("mon_SecuritySetDefaults",
mon_SecuritySetDefaults(client_authnLevel,client_authzSvc),
status);
CHK_STATUS(status, MON_SECURITYSET_FAILED,
"mon_SecuritySetDefaults failed");

ENCINA_CALL_RC("mon_SetHandleCacheRefreshInterval",
mon_SetHandleCacheRefreshInterval(300), status);
CHK_STATUS(status, MON_SETREFRESHINTERVAL_FAILED,
"mon_SetHandleCacheRefreshInterval failed");

{
dbInfo_data_t data;
trpc_status_t trpcStatus;
/* Get DB Info -- currently id does not do anything
but it will tell us if there is a server out there.
Better to know instead of when all the terminals
are up and ready
*/
impTPCCNOInfo(&data, &trpcStatus);
if (trpcStatus) {
char msg[100];
sprintf(msg, "TRPC error during db info at init.");
encina_error_message(msg, trpcStatus);
CHK_STATUS(33,NOINFO_TRPC_ERROR,
"TRPC error during db info at init");
}
}

/* Start bg_thread for debug purpose and performance tuning.
* In the final test, we do not start it in order to get the
* best performance.
* On NT, bg_thread may use lots of CPU. But we need to verify it.
*/
if (1)
start_bg_debug_thread();

client_enrolled = 1;
MUTEX_UNLOCK(&init_lock);
}

-----*/
/* Read environment parameters */
-----*/
static void read_mon_environment()
{
char *env_str;

cellName = getenv("ENCINA_TPM_CELL");
CHECK_ENVIRON(cellName, "ENCINA_TPM_CELL");

if (env_str = getenv("TPCC_ENV_RETRIEVE")) {
envRetrieval = atoi(env_str);
}
}

/*
* dump_pa_ring_buffer() -- For Debugging --
* Dump the ring buffer in the PA we are talking to
* Only works if we are using long term reservation
*/
static void dump_pa_ring_buffer(pa_handle)
trpc_handle_t pa_handle;
{
err_printf("Dumping Ring Buffer of server\n");
admin_trace_DumpRingBuffer((handle_t)pa_handle, "stderr");
}

/*terminal_context_init:
* The same function as thread_init in the thread-pool version.
*
* This function must be called by each terminal when using
* thread pool. After a terminal is logged on, the first thing
* it does is to call this function.
* This function creates a context for the terminal.
* It returns a pointer to a context that must be passed
* on calls back to this module.
*/

```

```

void *terminal_context_init(int fdIn)
{
int thread_index;
struct timezone tz;
thread_info_t *thread_infoP;

if (iStatsFrequency < 1)
return(NULL);

thread_infoP = (thread_info_t *)calloc(1, sizeof(thread_info_t));

thread_infoP->descr.state = thread_state_init;
gettimeofday(&thread_infoP->descr.init, &tz);
thread_infoP->initialized = 1;

MUTEX_LOCK(&init_lock);
thread_index = info_list_len++;
thread_infoP->thread_index = thread_index;
thread_infoP->thread_id = fdIn;

num_active_threads++;
info_list =
(thread_info_t **)realloc((void *)info_list, \
sizeof(thread_info_t *) * \
info_list_len);
info_list[thread_index] = thread_infoP;

MUTEX_UNLOCK(&init_lock);

if (num_active_threads % 200 == 0)
err_printf("Terminal %d Initialized (currently %d are \
active).\n", \
thread_index, num_active_threads);

return(thread_infoP);
}

neworder.tacf

/*
* Copyright (C) 1991, 1990 Transarc Corporation
* All Rights Reserved
*/
/*
* neworder.tacf -- attribute configuration file for tpcc server.
* used for transparent binding
*
* $Revision: 1.1 $
* $Date: 1998/11/06 21:10:13 $
* $Log: neworder.tacf,v $
*
* $ALog: neworder.tacf,v $
* Revision 1.1 1998/11/06 21:10:13 dongfeng
* - Move all files common to client and server to tpcc/common
* directory
* [added by delta dongfeng-23677-TPCC-new-directory-structures, r1.1]
*
* Revision 1.2 1998/10/08 18:03:01 gerstl
* Changes to allow configurations where some servers only service
* specific transaction types. Split transaction interfaces by type.
* [added by delta gerstl-23515-TPCC-allow-separate-online-transaction-interfaces, r1.1]
*
*
*
*/

[implicit_handle (mon_handle_t handle)]
interface neworder
{
}

neworder.tidl

/*
* id: Sid: $
*
* component_name: encina benchmarks
*
* the following functions list may not be complete.
* functions defined by/via macros may not be included.
*
* functions:
* <fill_me_in>
*
* origins: transarc corp.
*
* (c) copyright transarc corp. 1995, 1993
* all rights reserved
* licensed materials - property of transarc
*
* us government users restricted rights - use, duplication or
* disclosure restricted by gsa adp schedule contract with transarc corp

```

```
*/
*
* history
* Stalog: $
*/
*
* neworder.tidl -- interface definition file for tpccserver.
*
* $Revision: 1.0 $
* $Date: 1995/10/20 21:55:05 $
* $Log: tpcc.tidl,v $
*/
[
  uuid(f7065094-5e04-11d2-b351-9e621208aa77),
  version(1.0)
]
interface neworder
{
import "tpm/mon/mon_handle.idl";
import "tpcc_type.idl";

[nontransactional] void
  impTPCCNewOrder([in,out] newOrder_data_t *dataP,
                  [out] trpc_status_t * trpcStatus);

[nontransactional] void
  impTPCCNOInfo([out] dbInfo_data_t *dataP,
                [out] trpc_status_t * trpcStatus);
}

orderstatus.tacf

/*
* Copyright (C) 1991, 1990 Transarc Corporation
* All Rights Reserved
*/
*
* orderstatus.tacf -- attribute configuration file for tpcc server.
* used for transparent binding
*
* $Revision: 1.1 $
* $Date: 1998/11/06 21:10:14 $
* $Log: $
*
* $TALog: orderstatus.tacf,v $
* Revision 1.1 1998/11/06 21:10:14 dongfeng
* - Move all files common to client and server to tpcc/common
* directory
* [added by delta dongfeng-23677-TPCC-new-directory-structures, r1.1]
*
* Revision 1.2 1998/10/08 18:03:02 gerstl
* Changes to allow configurations where some servers only service
* specific transaction types. Split transaction interfaces by type.
* [added by delta gerstl-23515-TPCC-allow-separate-online-transaction-interfaces, r1.1]
*
*
*
*
* [implicit_handle (mon_handle_t handle)]
interface orderstatus
{
}

orderstatus.tidl

*
* id: Sid: $
*
* component_name: encina benchmarks
*
* the following functions list may not be complete.
* functions defined by/via macros may not be included.
*
* functions:
* <fill_me_in>
*
* origins: transarc corp.
*
* (c) copyright transarc corp. 1995, 1993
* all rights reserved
* licensed materials - property of transarc
*
* us government users restricted rights - use, duplication or
* disclosure restricted by gsa adp schedule contract with transarc corp
*/
*
* history
* Stalog: $
*/
*
* orderstatus.tidl -- interface definition file for tpccserver.
*
* $Revision: 1.0 $
```

```
* $date: 1995/10/20 21:55:05 $
* $log: tpcc.tidl,v $
*/
[
  uuid(06287200-5e05-11d2-8984-9e621208aa77),
  version(1.0)
]
interface orderstatus
{
import "tpm/mon/mon_handle.idl";
import "tpcc_type.idl";

[nontransactional] void
  impTPCCOrderStatus([in,out] orderStatus_data_t *dataP,
                     [out] trpc_status_t * trpcStatus);
}

payment.tacf

/*
* Copyright (C) 1991, 1990 Transarc Corporation
* All Rights Reserved
*/
*
* payment.tacf -- attribute configuration file for tpcc server.
* used for transparent binding
*
* $Revision: 1.1 $
*
*
*
* [implicit_handle (mon_handle_t handle)]
interface payment
{
}

payment.tidl

/*
* payment.tidl -- interface definition file for tpccserver.
*
* $Revision: 1.0 $
* $Date: 1995/10/20 21:55:05 $
* $Log: tpcc.tidl,v $
*/
[
  uuid(1341a902-5e05-11d2-bb70-9e621208aa77),
  version(1.0)
]
interface payment
{
import "tpm/mon/mon_handle.idl";
import "tpcc_type.idl";

[nontransactional] void
  impTPCCPayment([in,out] payment_data_t *dataP,
                  [out] trpc_status_t * trpcStatus);
}

screen.C

/* (C)1997 IBM Corporation */
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <ctype.h>
#include <string.h>
#include <math.h>

#include "screen.h"
#include "format.h"
#include "encina.h"

#define USE_INSULTS
#define LOCAL_SESSION_DATA

extern "C" err_printf(...);

extern char const * const blanks;
extern char const * const underscores;
extern char const * const backspaces;

static int clear_eos(InOut *ioP);
static int clear_eos(char *buf);
static int string_empty(char const *text);
static int pos_zero(int const *val);
static int pos_nonzeros(int const **val);

/*****
Screen
*****/
```



```

int Screen::reset() {
    has_data=0;
    pos=0;
    if (dataptr) memset(dataptr, 0, data_len);
    for (int i = 0; fields && fields[i] != NULL; i++) {
        fields[i]->reset();
    }
    return 0;
};

int Screen::present_empty_fields() {
    if (empty_fields)
        threadP->write(empty_fields, empty_fields_len);
    // threadP->write(end_str, end_str_len);
    return 0;
}

int Screen::present() {
    threadP->write(screen, screen_len);
    threadP->write(session_data, session_data_len);
    if (has_data) {
        for (int i = 0; fields[i] != NULL; i++) {
            fields[i]->display_field(1);
        }
        // threadP->write(end_str, end_str_len);
    } else {
        present_empty_fields();
    }
    return 0;
};

int Screen::user_input() {
    int key;
    has_data = 1;
    fields[pos]->start_position();
    threadP->flush();
    // threadP->mark();
    key = fields[pos]->get_field(0);
    do {
        switch (key) {
            case EOF:
                return 0;
                break;
            case Field::NEXT_FIELD:
                if (fields[++pos] == NULL) {
                    pos = 0;
                }
                break;
            case Field::PREV_FIELD:
                if (--pos < 0) {
                    while (fields[++pos] != NULL);
                    pos--;
                }
                break;
            case Field::REDISPLAY:
                present();
                break;
            case Field::ABORT:
                position(1, 2);
                threadP->write(end_str, end_str_len);
                return 0;
            case Field::ENTER:
                if (validate()) {
                    threadP->pop();
                    return 1;
                }
                break;
        }
        key = fields[pos]->get_field(0);
    } while (1);
    return 0;
}

Screen::~Screen() {
    if (fields != NULL) {
        for (int lpos = 0; fields[lpos] != NULL; lpos++) {
            delete fields[lpos];
        }
        delete [] fields;
    }
    fields=NULL;
}

int Screen::display_status(int status) {
    position(threadP, status_x, status_y);
    threadP->write("Execution Status: ");
    if (status == TRAN_OK) {
        threadP->write("Transaction Committed");
    } else if (status == INVALID_ITEM) {
        threadP->write("Item number is not valid");
    } else {
        threadP->write("ERROR: Rollback -- ");
        // threadP->write("Rollback -- ");
        char buf[6];
        format_int(buf, 5, status);
        threadP->write(buf, 5);
    }
    return 0;
}

int Screen::handle() {
    threadP->debug("%s - reset\n", tran_type);
    reset();

    threadP->debug("%s - present\n", tran_type);
    threadP->hold();
    present();
    threadP->write(TRIGGER, 1);
    threadP->debug("%s - user_input\n", tran_type);
    if (user_input()) {
        threadP->write(end_str, end_str_len);
        threadP->write(TRIGGER, 1);
        return -1;
    }
    threadP->flush();
    threadP->hold();
    threadP->debug("%s - process\n", tran_type);
    if (process()) {
        threadP->write(end_str, end_str_len);
        threadP->write(TRIGGER, 1);
        return -1;
    }
    threadP->debug("%s - respond\n", tran_type);
    respond();
    // position(threadP, 1, 2);
    threadP->write(end_str, end_str_len);
    threadP->write(TRIGGER, 1);
    threadP->flush();
    return 0;
}

//*****
NewOrder
//*****

int NewOrder::reset() {
    Screen::reset();
    pos=start_field;
    memset(dataptr, 0, sizeof(*data));
    return 0;
};

NewOrder::NewOrder(User_data *udP, Thread_data *threadP) : Screen(udP, threadP) {
    tran_type = NEWORDER_SERVICE;
    dataptr = data = new NewOrder_data;
    data_len = sizeof(NewOrder_data);

    status_x = 1;
    status_y = 24;

    screen = static_screen;
    empty_fields = static_empty_fields;
    #ifdef LOCAL_SESSION_DATA
    session_data = new char[static_session_data_len+1];
    sprintf(session_data, "%s%5d", POS(12,4), user_dataP->warehouse);
    #else
    session_data = static_session_data;
    sprintf(session_data, "%s%5d", POS(12,4), user_dataP->warehouse);
    #endif
    screen_len = static_screen_len;
    empty_fields_len = static_empty_fields_len;
    session_data_len = static_session_data_len;

    int lpos = 0;
    fields = new Field *[2+MAX_ITEMS*3+1];
    for (int i = 0; i < MAX_ITEMS; i++) {
        fields[lpos++] = genfield(threadP, 3, 9+i, 5, &data->item[i].s_OL_SUPPLY_W_ID);
        fields[lpos++] = genfield(threadP, 10, 9+i, 6, &data->item[i].s_OL_I_ID);
        fields[lpos++] = genfield(threadP, 45, 9+i, 2, &data->item[i].s_OL_QUANTITY);
    }
    #if defined(USE_SMART_FIELDS)
    if (i > 0) {
        int **tmp = new int *[4];
        tmp[0] = &fields[lpos-6]->pos;
        tmp[1] = &fields[lpos-5]->pos;
        tmp[2] = &fields[lpos-4]->pos;
        tmp[3] = NULL;
        fields[lpos-3]->ok_func = (int (*)(void*))pos_nonzeros;
        fields[lpos-3]->ok_data = tmp;
        fields[lpos-2]->ok_func = (int (*)(void*))pos_nonzeros;
        fields[lpos-2]->ok_data = tmp;
        fields[lpos-1]->ok_func = (int (*)(void*))pos_nonzeros;
        fields[lpos-1]->ok_data = tmp;
    }
    #endif
    start_field = lpos;
    fields[lpos++] = genfield(threadP, 29, 4, 4, &data->s_D_ID); /* District */
    fields[lpos++] = genfield(threadP, 12, 5, 4, &data->s_C_ID); /* Customer */
    fields[lpos++] = NULL;
    reset();
};

int NewOrder::validate() {
    if (!fields[start_field]->pos) {
        pos=start_field;
        message(threadP, "District ID is a required field");
        return 0;
    }
    if (!fields[start_field+1]->pos) {
        pos=start_field+1;
        message(threadP, "Customer ID is a required field");
        return 0;
    }
}

```

```

int last=-1;
data->s_O_OL_CNT = 0;
data->s_all_local = 1;
data->s_W_ID = user_dataP->warehouse;
for (int i = 0; i < MAX_ITEMS*3; i+=3) {
    if (fields[i]->pos || fields[i+1]->pos || fields[i+2]->pos) {
        if (!fields[i]->pos) {
            pos=i;
        }
        #if defined(USE_INSULTS)
            message(threadP, "Yeah, I think this is a bogus field too.");
        #else
            message(threadP, "Warehouse ID is a required field");
        #endif
        return 0;
    }
    if (!fields[i+1]->pos) {
        pos=i+1;
    }
    #if defined(USE_INSULTS)
        message(threadP, "Umm, WHAT did you want?");
    #else
        message(threadP, "Item ID is a required field");
    #endif
    return 0;
}
if (data->item[i/3].s_OL_QUANTITY <= 0) {
    pos=i+2;
}
#if defined(USE_INSULTS)
    message(threadP, "So something plus nothing is...");
#else
    message(threadP, "Please enter a quantity greater than 0");
#endif
return 0;
}
if (data->item[i/3].s_OL_SUPPLY_W_ID != data->s_W_ID) {
    data->s_all_local=0;
}
data->s_O_OL_CNT++;
} else if (last < 0) {
    last = i;
}
}
if (data->s_O_OL_CNT <= 0) {
    pos=0;
}
#if defined(USE_INSULTS)
    message(threadP, "It's kind of pointless without ordering something isn't it?");
#else
    message(threadP, "Please enter an item to order");
#endif
return 0;
}
// Compress the order lines: some of them may be empty
int ind;
for (i=0, ind=0; ind<data->s_O_OL_CNT; i++) {
    if (fields[i*3]->pos) {
        if (i > ind) {
            data->item[ind] = data->item[i];
        }
        ind ++;
    }
}
if (i > ind) {
    int j;
    for (j=ind; j<i; j++) {
        /* At least one empty line was skipped */
        data->item[j].s_OL_SUPPLY_W_ID = 0;
        data->item[j].s_OL_I_ID = 0;
        data->item[j].s_OL_QUANTITY = 0;
    }
}
return 1;
}
int NewOrder::respond() {
    int i;
    double amount, total_amount, cost;
    char buf[32];
    position(threadP, 1, 9); clear_eos(threadP);
    position(threadP, 25, 5); threadP->write(data->s_C_LAST);
    position(threadP, 52, 5); threadP->write(data->s_C_CREDIT);
    position(threadP, 15, 6); format_int(buf, 9, data->s_O_ID); threadP->write(buf, 8);

    display_status(data->s_transtatus);
    if (data->s_transtatus != TRAN_OK) {
        return -1;
    }

    position(threadP, 25, 5); threadP->write( data->s_C_LAST);
    position(threadP, 52, 5); threadP->write( data->s_C_CREDIT);
    position(threadP, 15, 6); format_int( buf, 9, data->s_O_ID); threadP->write(buf, 8);
    position(threadP, 48, 6); format_int( buf, 3, data->s_O_OL_CNT); threadP->write(buf, 2);
    position(threadP, 61, 4); format_date(buf, 20, data->s_O_ENTRY_D); threadP->write(buf, 19);
    position(threadP, 64, 5); format_float(buf, 6, 2, data->s_C_DISCOUNT/100); threadP->write(buf, 5);
    position(threadP, 59, 6); format_float(buf, 6, 2, data->s_W_TAX/100); threadP->write(buf, 5);
    position(threadP, 74, 6); format_float(buf, 6, 2, data->s_D_TAX/100); threadP->write(buf, 5);
    total_amount = 0;
    for (i=0; i < data->s_O_OL_CNT; i++) {
        position(threadP, 3, 9+i); format_int(buf, 6, data->item[i].s_OL_SUPPLY_W_ID);
        threadP->write( buf, 5);
        position(threadP, 10, 9+i); format_int(buf, 7, data->item[i].s_OL_I_ID); threadP->write(
        buf, 6);
        position(threadP, 19, 9+i); threadP->write( data->item[i].s_I_NAME);
        position(threadP, 45, 9+i); format_int(buf, 3, data->item[i].s_OL_QUANTITY);
        threadP->write(buf, 2);
        position(threadP, 51, 9+i); format_int(buf, 4, data->item[i].s_S_QUANTITY);
        threadP->write(buf, 3);
        position(threadP, 58, 9+i); threadP->write(&data->item[i].s_brand_generic, 1);
        position(threadP, 62, 9+i); format_money(buf, 8, data->item[i].s_I_PRICE);
        threadP->write(buf, 7);
        position(threadP, 71, 9+i); format_money(buf, 10, data->item[i].s_OL_AMOUNT);
        threadP->write(buf, 9);
    }
    /* Clear the screen of any empty input fields */
    position(threadP, 63, 24); threadP->write( "Total:");
    position(threadP, 70, 24); format_money( buf, 10, data->s_total_amount ); threadP->write( buf, 9 );

    return 0;
}

/*****
Payment
*****/
Payment::Payment(User_data *udP, Thread_data *threadP) : Screen(udP, threadP) {
    tran_type = PAYMENT_SERVICE;
    dataptr = data = new Payment_data;
    data_len = sizeof(Payment_data);

    int lpos = 0;
    screen = static_screen;
    empty_fields = static_empty_fields;
    #if def LOCAL_SESSION_DATA
        session_data = new char[static_session_data_len+1];
        sprintf(session_data, "%s%5d", POS(12,6), user_dataP->warehouse);
    #else
        session_data = static_session_data;
        sprintf(session_data, "%s%5d", POS(12,6), user_dataP->warehouse);
    #endif
    screen_len = static_screen_len;
    empty_fields_len = static_empty_fields_len;
    session_data_len = static_session_data_len;

    fields = new Field *[7];
    fields[lpos++] = genfield(threadP, 52, 6, 2, &data->s_D_ID); /* District */
    fields[lpos++] = genfield(threadP, 11, 11, 4, &data->s_C_ID); /* Customer # */
    fields[lpos++] = genfield(threadP, 29, 12, 16, (char *)data->s_C_LAST); /* Name */
    fields[lpos++] = genfield(threadP, 33, 11, 5, &data->s_C_W_ID); /* Cust-Warehouse */
    fields[lpos++] = genfield(threadP, 54, 11, 2, &data->s_C_D_ID); /* Cust-District */
    fields[lpos++] = genfield(threadP, 23, 17, 8, &data->s_H_AMOUNT); /* Amount Paid */
    fields[lpos++] = NULL;
    #if defined(USE_SMART_FIELDS)
        fields[1]->ok_func = (int (*)(void*))pos_zero;
        fields[1]->ok_data = &fields[2]->pos;
        fields[2]->ok_func = (int (*)(void*))pos_zero;
        fields[2]->ok_data = &fields[1]->pos;
    #endif
};

int Payment::validate() {
    if (!fields[0]->pos) {
        pos=0;
        message(threadP, "District ID is a required field");
        return 0;
    }
    if (fields[1]->pos) {
        #if defined(USE_BYNAME)
            data->s_byname = 0;
        #endif
        #if defined(USE_BYNAME)
            #else if (fields[2]->pos) {
                #if defined(USE_BYNAME)
                    data->s_byname = 1;
                #endif
            } else {
                pos=1;
                message(threadP, "Customer ID or Name is required");
                return 0;
            }
        }
        if (!fields[3]->pos) {
            pos=3;
            message(threadP, "Customer Warehouse is a required field");
            return 0;
        }
        if (!fields[4]->pos) {
            pos=4;
            message(threadP, "Customer District is a required field");
            return 0;
        }
        if (data->s_H_AMOUNT <= 0) {
            pos=5;
            message(threadP, "Enter a positive amount");
            return 0;
        }
    }
    data->s_W_ID = user_dataP->warehouse;
}

```

<pre> return 1; }  int Payment::respond() { if (data-&gt;s_transtatus != TRAN_OK) {     display_status(data-&gt;s_transtatus);     return -1; }  char buf[32]; position(threadP, 52, 6); format_int(buf, 3, data-&gt;s_D_ID); threadP-&gt;write(buf, 2); position(threadP, 33, 11); format_int(buf, 5, data-&gt;s_C_W_ID); threadP-&gt;write(buf, 4); position(threadP, 54, 11); format_int(buf, 3, data-&gt;s_C_D_ID); threadP-&gt;write(buf, 2); position(threadP, 7, 4); threadP-&gt;write( data-&gt;s_H_DATE ); position(threadP, 1, 7); threadP-&gt;write( data-&gt;s_W_STREET_1); position(threadP, 42, 7); threadP-&gt;write( data-&gt;s_D_STREET_1); position(threadP, 1, 8); threadP-&gt;write( data-&gt;s_W_STREET_2); position(threadP, 42, 8); threadP-&gt;write( data-&gt;s_D_STREET_2); position(threadP, 1, 9); threadP-&gt;write( data-&gt;s_W_CITY); position(threadP, 22, 9); threadP-&gt;write( data-&gt;s_W_STATE); position(threadP, 25, 9); format_zip(buf, 10, data-&gt;s_W_ZIP); threadP-&gt;write(buf, 10); position(threadP, 42, 9); threadP-&gt;write( data-&gt;s_D_CITY); position(threadP, 63, 9); threadP-&gt;write( data-&gt;s_D_STATE); position(threadP, 66, 9); format_zip(buf, 10, data-&gt;s_D_ZIP); threadP-&gt;write(buf, 10); position(threadP, 11, 11); format_int(buf, 5, data-&gt;s_C_ID); threadP-&gt;write(buf, 4); position(threadP, 9, 12); threadP-&gt;write( data-&gt;s_C_FIRST); position(threadP, 26, 12); threadP-&gt;write( data-&gt;s_C_MIDDLE); position(threadP, 29, 12); threadP-&gt;write( data-&gt;s_C_LAST); position(threadP, 58, 12); format_date(buf, 10, data-&gt;s_C_SINCE); threadP-&gt;write( buf, 10); position(threadP, 9, 13); threadP-&gt;write( data-&gt;s_C_STREET_1); position(threadP, 58, 13); threadP-&gt;write( data-&gt;s_C_CREDIT); position(threadP, 9, 14); threadP-&gt;write( data-&gt;s_C_STREET_2); position(threadP, 58, 14); format_float(buf, 6, 2, data-&gt;s_C_DISCOUNT/100); threadP-&gt;write(buf, 6); position(threadP, 9, 15); threadP-&gt;write( data-&gt;s_C_CITY); position(threadP, 30, 15); threadP-&gt;write( data-&gt;s_C_STATE); position(threadP, 33, 15); format_zip(buf, 10, data-&gt;s_C_ZIP); threadP-&gt;write(buf, 10); position(threadP, 58, 15); format_phone(buf, 18, data-&gt;s_C_PHONE ); threadP-&gt;write(buf, 18); position(threadP, 17, 17); format_money( buf, 15, data-&gt;s_H_AMOUNT); threadP-&gt;write(buf, 14); position(threadP, 55, 17); format_money( buf, 16, data-&gt;s_C_BALANCE); threadP-&gt;write(buf, 15); position(threadP, 17, 18); format_money( buf, 15, data-&gt;s_C_CREDIT_LIM); threadP-&gt;write(buf, 14);  if (data-&gt;s_C_CREDIT[0] == 'B' &amp;&amp; data-&gt;s_C_CREDIT[1] == 'C') {     int i, size = strlen((char *)data-&gt;s_C_DATA);     for (i = 0; i &lt; 4; i++) {         position(threadP, 12, 20+i);         threadP-&gt;write(data-&gt;s_C_DATA, (size &gt; 50)?50:size);         size -= 50;         if (size &lt;= 0) break;     } }  return 0; }  ***** OrderStatus ***** OrderStatus::OrderStatus(User_data *udP, Thread_data *threadP) : Screen(udP, threadP) {     tran_type = ORDERSTATUS_SERVICE;     dataptr = data = new OrderStatus_data;     data_len = sizeof(OrderStatus_data);      status_x=1;     status_y=25;      int pos = 0;     screen = static_screen;     empty_fields = static_empty_fields; #define LOCAL_SESSION_DATA     session_data = new char[static_session_data_len+1];     sprintf(session_data, "%s%5d", POS(12,4), user_dataP-&gt;warehouse); #else     session_data = static_session_data;     sprintf(session_data, "%s%5d", POS(12,4), user_dataP-&gt;warehouse); #endif     screen_len = static_screen_len;     empty_fields_len = static_empty_fields_len;     session_data_len = static_session_data_len;      fields = new Field *[4];     fields[pos++] = genfield(threadP, 29, 4, 2, &amp;data-&gt;s_D_ID); /* District */     fields[pos++] = genfield(threadP, 11, 5, 4, &amp;data-&gt;s_C_ID); /* Customer ID */     fields[pos++] = genfield(threadP, 44, 5, 16, (char *)data-&gt;s_C_LAST); /* Customer Name */     fields[pos++] = NULL; #define USE_SMART_FIELDS     fields[1]-&gt;ok_func = (int (*)(void*))pos_zero;     fields[1]-&gt;ok_data = &amp;fields[2]-&gt;pos;     fields[2]-&gt;ok_func = (int (*)(void*))pos_zero;     fields[2]-&gt;ok_data = &amp;fields[1]-&gt;pos; #endif };  int OrderStatus::validate() { if (!fields[0]-&gt;pos) {     pos=0;     message(threadP, "District ID is a required field");     return 0; } </pre>	<pre> } if (fields[1]-&gt;pos) { #define USE_BYNAME     data-&gt;s_byname = 0; #endif } else if (fields[2]-&gt;pos) { #define USE_BYNAME     data-&gt;s_byname = 1; #endif } else {     pos=1;     message(threadP, "Customer ID or Name is required");     return 0; }  data-&gt;s_W_ID = user_dataP-&gt;warehouse;  return 1; }  int OrderStatus::respond() { display_status(data-&gt;s_transtatus); if (data-&gt;s_transtatus != TRAN_OK)     return -1;  char buf[32];  position(threadP, 11, 5); format_int(buf, 5, data-&gt;s_C_ID); threadP-&gt;write(buf, 4); position(threadP, 24, 5); threadP-&gt;write(data-&gt;s_C_FIRST); position(threadP, 41, 5); threadP-&gt;write(data-&gt;s_C_MIDDLE); position(threadP, 44, 5); threadP-&gt;write(data-&gt;s_C_LAST); position(threadP, 15, 6); format_money(buf, 11, data-&gt;s_C_BALANCE); threadP-&gt;write(buf, 10); position(threadP, 15, 8); format_int(buf, 9, data-&gt;s_O_ID); threadP-&gt;write(buf, 8); position(threadP, 38, 8); format_date(buf, 19, data-&gt;s_O_ENTRY_D); threadP-&gt;write(buf); if (data-&gt;s_O_CARRIER_ID &gt; 0) {     position(threadP, 76, 8);     format_int(buf, 3, data-&gt;s_O_CARRIER_ID);     threadP-&gt;write(buf, 2); }  for (int i=0; i &lt; data-&gt;s_ol_cnt; i++) {     position(threadP, 3, i+10);     format_int(buf, 6, data-&gt;item[i].s_OL_SUPPLY_W_ID);     threadP-&gt;write(buf, 5);      position(threadP, 14, i+10);     format_int(buf, 7, data-&gt;item[i].s_OL_I_ID);     threadP-&gt;write(buf, 6);      position(threadP, 25, i+10);     format_int(buf, 3, data-&gt;item[i].s_OL_QUANTITY);     threadP-&gt;write(buf, 2);      position (threadP, 32, i+10);     format_money(buf, 10, data-&gt;item[i].s_OL_AMOUNT);     threadP-&gt;write(buf, 9);      position (threadP, 47, i+10);     format_date(buf, 20, data-&gt;item[i].s_OL_DELIVERY_D);     threadP-&gt;write(buf, 19); }  return 0; }  ***** Delivery ***** Delivery::Delivery(User_data *udP, Thread_data *threadP) : Screen(udP, threadP) {     tran_type = DELIVERY_SERVICE;     dataptr = data = new Delivery_data;     data_len = sizeof(Delivery_data);      status_x = 1;     status_y = 8;      int pos = 0;     screen = static_screen;     empty_fields = static_empty_fields; #define LOCAL_SESSION_DATA     session_data = new char[static_session_data_len+1];     sprintf(session_data, "%s%5d", POS(12,4), user_dataP-&gt;warehouse); #else     session_data = static_session_data;     sprintf(session_data, "%s%5d", POS(12,4), user_dataP-&gt;warehouse); #endif     screen_len = static_screen_len;     empty_fields_len = static_empty_fields_len;     session_data_len = static_session_data_len;      fields = new Field *[2];     fields[pos++] = genfield(threadP, 17, 6, 2, &amp;data-&gt;s_O_CARRIER_ID); /* Carrier Number */     fields[pos++] = NULL; };  int Delivery::validate() { if (!fields[0]-&gt;pos) {     pos=0; } </pre>
--	--

```

        message(threadP, "Carrier ID is a required field");
        return 0;
    }
    time((time_t *)&(data->s_queued_time));

    data->s_W_ID = user_dataP->warehouse;

    return 1;
}

int Delivery::respond() {
    if (data->s_transtatus == TRAN_OK) {
        position(threadP, status_x, status_y);
        threadP->write("Execution Status: Delivery has been queued");
    } else {
        display_status(data->s_transtatus);
        return -1;
    }
    return 0;
}

*****
StockLevel
*****/
StockLevel::StockLevel(User_data *udP, Thread_data *threadP) : Screen(udP, threadP) {
    tran_type = STOCKLEVEL_SERVICE;
    dataptr = data = new StockLevel_data;
    data_len = sizeof(StockLevel_data);

    status_x = 1;
    status_y = 10;

    int pos = 0;
    screen = static_screen;
    empty_fields = static_empty_fields;
    session_data = static_session_data;
#ifdef LOCAL_SESSION_DATA
    session_data = new char[static_session_data_len+1];
    sprintf(session_data, "%s%5d%s%2d", POS(12,4), user_dataP->warehouse,
                                                    POS(29,4),
user_dataP->district);
#else
    session_data = static_session_data;
    sprintf(session_data, "%s%5d%s%2d", POS(12,4), user_dataP->warehouse,
                                                    POS(29,4),
user_dataP->district);
#endif
    screen_len = static_screen_len;
    empty_fields_len = static_empty_fields_len;
    session_data_len = static_session_data_len;

    fields = new Field * [2];
    fields[pos++] = genfield(threadP, 24, 6, 2, &data->s_threshold); /* Threshold */
    fields[pos++] = NULL;
};

int StockLevel::validate() {
    if (data->s_threshold <= 0) {
        pos=0;
        message(threadP, "A positive non-zero threshold is required");
        return 0;
    }
    data->s_W_ID = user_dataP->warehouse;
    data->s_D_ID = user_dataP->district;

    return 1;
}

int StockLevel::respond() {
    display_status(data->s_transtatus);
    if (data->s_transtatus != TRAN_OK)
        return -1;

    position(threadP, 12, 8);
    char buf[5];
    format_int(buf, 4, data->s_low_stock);
    threadP->write(buf, 4);

    return 0;
}

*****
perform
*****/
int NewOrder::process() {
    if (tran_type == NULL)
        return 0;

    if (encina.tran(data, threadP->contextP, tran_type) < 0) {
        return -1;
    }
    return 0;
}

int Payment::process() {
    if (tran_type == NULL)
        return 0;

    if (encina.tran(data, threadP->contextP, tran_type) < 0) {
        return -1;
    }
    return 0;
}

}
return 0;
}

int StockLevel::process() {
    if (tran_type == NULL)
        return 0;

    if (encina.tran(data, threadP->contextP, tran_type) < 0) {
        return -1;
    }
    return 0;
}

int OrderStatus::process() {
    if (tran_type == NULL)
        return 0;

    if (encina.tran(data, threadP->contextP, tran_type) < 0) {
        return -1;
    }
    return 0;
}

int Delivery::process() {
    if (tran_type == NULL)
        return 0;

    if (encina.tran(data, threadP->contextP, tran_type) < 0) {
        return -1;
    }
    return 0;
}

int Screen::process() {
    if (tran_type == NULL)
        return 0;
}

/*****
Login
*****/
Login::Login(User_data *udP, Thread_data *threadP) : Screen(udP, threadP) {
    tran_type = NULL;
    status_x=1;
    status_y=24;

    dataptr = NULL;
    data_len = 0;

    int pos = 0;
    screen = static_screen;
    screen_len = static_screen_len;
    empty_fields = static_empty_fields;
    empty_fields_len = static_empty_fields_len;

    fields = new Field * [3];
    fields[pos++] = genfield(threadP, 16, 5, 5, &(udP->warehouse)); //Warehouse
    fields[pos++] = genfield(threadP, 34, 5, 2, &(udP->district)); //District
    fields[pos++] = NULL;
};

int Login::validate() {
    if (!fields[0]->pos) {
        pos=0;
        message(threadP, "Warehouse ID is a required field");
        return 0;
    }
    if (!fields[1]->pos) {
        pos=1;
        message(threadP, "District ID is a required field");
        return 0;
    }
}

return 1;
}

/*****
Menu
*****/
Menu::Menu(User_data *udP, Thread_data *threadP) : Screen(udP, threadP) {
    tran_type = NULL;
    status_x=1;
    status_y=24;

    int pos = 0;
    screen = static_screen;
    screen_len = static_screen_len;
    empty_fields = NULL;
    empty_fields_len = 0;

    fields = NULL;
};

```

```

*****
Static data
*****
char const * const blanks = " ";
char const * const underscores = "_____";
char const * const backspaces = "\b\b\b\b\b\b\b\b\b\b";
*****
Utility Functions
*****
static int string_empty(char const *data) {
    return data[0] == 0;
}

static int pos_zero(int const *val) {
    return *val == 0;
}

static int pos_nonzeros(int const **val) {
    int const **ptr;
    for (ptr = val; ptr++; ) {
        if (**ptr == 0)
            return 0;
    }
    return 1;
}

int position(int x, int y, char *buf) {
    int pos = 0;
    buf[pos++] = ESCc;
    buf[pos++] = '[';
    if (y >= 10) buf[pos++] = (y / 10) + '0';
    buf[pos++] = (y % 10) + '0';
    buf[pos++] = ':';
    if (x >= 10) buf[pos++] = (x / 10) + '0';
    buf[pos++] = (x % 10) + '0';
    buf[pos++] = 'H';
    buf[pos++] = 0;
    return 0;
}

int position(InOut *threadP, int x, int y) {
    char buf[16];
    position(x, y, buf);
    threadP->write(buf);
    return 0;
}

static int clear_eos(InOut *threadP) {
    threadP->write(ESC "[J");
    return 0;
}

int message(InOut *threadP, char const *text, int need_flush) {
    position(threadP, 1,25);
    threadP->write(text);
    clear_eos(threadP);
    if (need_flush)
        threadP->flush();
    return 0;
}

static int clear_eos(char *buf) {
    buf[0] = ESCc;
    buf[1] = '[';
    buf[2] = 'J';
    return 0;
}

* (C)1997 IBM Corporation */
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#ifdef WIN32
#include <termios.h>
#endif
#include <time.h>

#include "field.h"
#include "inout.h"
#include "tpcc.h"

extern int position(int x, int y, char *buf);
extern int position(InOut *ioP, int x, int y);
extern int message(InOut *ioP, char const *text, int need_flush=1);

class User_data {
public:
    int warehouse;
    int district;
};

class Thread_data : public InOut {
public:
    void *contextP;

```

## screen.h

```

Thread_data(int infd, int outfd, void *conP) : InOut(infd, outfd), contextP(conP) {};
};

class Screen {
protected:
    static char const end_str[];
    static int end_str_len;
    int has_data;
    void *dataptr;
    char *tran_type;
    char const *screen;
    char const *empty_fields;
    char *session_data;
    int screen_len;
    int session_data_len;
    int empty_fields_len;
    int pos;
    int status_x, status_y;
    int data_len;
    Thread_data *threadP;

public:
    User_data *user_dataP;
    Field **fields;
    virtual char const *isa() { return "Screen"; };
    virtual int reset();
    virtual int present();
    virtual int present_empty_fields();
    virtual int process();
    virtual int user_input();
    virtual int validate() { return 1; };
    virtual int respond() { return 0; };
    int handle();
    int display_status(int status);
    Screen(User_data *udP, Thread_data *thrP) {
        user_dataP = udP;
        threadP = thrP;
        has_data = 0;
        pos = 0;
        fields = NULL;
        screen = empty_fields = session_data = NULL;
        screen_len = session_data_len = empty_fields_len = 0;
    };
    virtual ~Screen();
};

class Login : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int static_screen_len;
    static int static_empty_fields_len;
    static int static_session_data_len;
public:
    int validate();
    Login::Login(User_data *udP, Thread_data *thrP);
};

class NewOrder : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int static_screen_len;
    static int static_empty_fields_len;
    static int static_session_data_len;
    int start_field;
    void swap_fields(int i, int j);
public:
    NewOrder_data *data;

    int reset();
    NewOrder::NewOrder(User_data *udP, Thread_data *thrP);
    int validate();
    int process();
    int respond();
};

class Payment : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int static_screen_len;
    static int static_empty_fields_len;
    static int static_session_data_len;
public:
    Payment_data *data;
    int validate();
    int process();
    int respond();
    Payment(User_data *udP, Thread_data *thrP);
};

class OrderStatus : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];

```

```

static char static_session_data[];
static int static_screen_len;
static int static_empty_fields_len;
static int static_session_data_len;
public:
    OrderStatus_data *data;
    int validate();
    int process();
    int respond();

    OrderStatus(User_data *udP, Thread_data *thrP);
};

class Delivery : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int static_screen_len;
    static int static_empty_fields_len;
    static int static_session_data_len;
public:
    Delivery_data *data;
    int validate();
    int process();
    int respond();

    Delivery(User_data *udP, Thread_data *thrP);
};

class StockLevel : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int static_screen_len;
    static int static_empty_fields_len;
    static int static_session_data_len;
public:
    StockLevel_data *data;
    int validate();
    int process();
    int respond();

    StockLevel(User_data *udP, Thread_data *thrP);
};

class Menu : public Screen {
protected:
    static char const static_screen[];
    static char const static_empty_fields[];
    static char static_session_data[];
    static int static_screen_len;
    static int static_empty_fields len;
    static int static_session_data len;
public:
    Menu(User_data *udP, Thread_data *thrP);
};

/* (C)1997 IBM Corporation */
#include "screen.h"

char const NewOrder::static_screen[] =
POS( 1, 3) CLEAR EOS
POS(36, 3) "New Order"
POS( 1, 4) "Warehouse"
POS(19, 4) "District:"
POS(55, 4) "Date:"
POS( 1, 5) "Customer:"

POS(19, 5) "Name:"
POS(44, 5) "Credit:"
POS(57, 5) "Disc.:"
POS( 1, 6) "Order Number:"
POS(25, 6) "Number of Lines:"
POS(52, 6) "W_Tax:"
POS(67, 6) "D_Tax:"
POS( 2, 8) "Supp_W Item_Num Item_Name"
POS(44, 8) "Qty Stock B/G Price Amount"

char const NewOrder::static_empty_fields[] =
POS(29, 4) " " /* District */
POS(12, 5) " " /* Customer */

POS( 3, 9) " "
POS(10, 9) " "
POS(45, 9) " "
POS( 3,10) " "
POS(10,10) " "
POS(45,10) " "
POS( 3,11) " "
POS(10,11) " "
POS(45,11) " "

POS( 3,12) " "
POS(10,12) " "
POS(45,12) " "
POS( 3,13) " "
POS(10,13) " "
POS(45,13) " "
POS( 3,14) " "
POS(10,14) " "
POS(45,14) " "
POS( 3,15) " "
POS(10,15) " "
POS(45,15) " "
POS( 3,16) " "
POS(10,16) " "
POS(45,16) " "
POS( 3,17) " "
POS(10,17) " "
POS(45,17) " "
POS( 3,18) " "
POS(10,18) " "
POS(45,18) " "
POS( 3,19) " "
POS(10,19) " "
POS(45,19) " "
POS( 3,20) " "
POS(10,20) " "
POS(45,20) " "
POS( 3,21) " "
POS(10,21) " "
POS(45,21) " "
POS( 3,22) " "
POS(10,22) " "
POS(45,22) " "
POS( 3,23) " "
POS(10,23) " "
POS(45,23) " "

;

char NewOrder::static_session_data[] =
    POS(12,4) "#####" /* Warehouse Id */

;

int NewOrder::static_screen_len = sizeof(NewOrder::static_screen) - 1;
int NewOrder::static_empty_fields_len = sizeof(NewOrder::static_empty_fields) - 1;
int NewOrder::static_session_data_len = sizeof(NewOrder::static_session_data) - 1;

/* Payment */
char const Payment::static_screen[] =
POS( 1, 3) CLEAR EOS
POS( 38,3) "Payment"
POS( 1,4) "Date:"
POS( 1,6) "Warehouse:"
POS( 42,6) "District:"
POS( 1,11) "Customer:"
POS( 17,11) "Cust-Warehouse:"
POS( 39,11) "Cust-District:"
POS( 1,12) "Name:"
POS( 50,12) "Since:"
POS( 50,13) "Credit:"
POS( 50,14) "%Disc:"
POS( 50,15) "Phone:"
POS( 1,17) "Amount Paid:"
POS( 37,17) "New Cust-Balance:"
POS( 1,18) "Credit Limit:"
POS( 1,20) "Cust-Data:"

;

char const Payment::static_empty_fields[] =
POS( 52, 6) " " /* District */
POS( 11,11) " " /* Customer # */
POS( 33,11) " " /* Cust-Warehouse */
POS( 54,11) " " /* Cust-District */
POS( 29,12) " " /* Name */
POS( 23,17) " " /* Amount Paid */

;

char Payment::static_session_data[] =
    POS( 12,6) "#####" /* Warehouse */

;

int Payment::static_screen_len = sizeof(Payment::static_screen) - 1;
int Payment::static_empty_fields_len = sizeof(Payment::static_empty_fields) - 1;
int Payment::static_session_data_len = sizeof(Payment::static_session_data) - 1;

/* Order Status */
char const OrderStatus::static_screen[] =
POS( 1, 3) CLEAR EOS
POS(35, 3) "Order-Status"
POS( 1, 4) "Warehouse:"
POS(19, 4) "District:"
POS( 1, 5) "Customer:"
POS(18, 5) "Name:"
POS( 1, 6) "Cust-Balance:"
POS( 1, 8) "Order-Number"
POS(26, 8) "Entry-Date:"
POS(60, 8) "Carrier-Number:"
POS( 1, 9) "Supply-W"
POS(14, 9) "Item-Num"
POS(25, 9) "Qty"
POS(33, 9) "Amount"

```

**screen\_data.C**

```

POS(45, 9) "Delivery-Date"

char const OrderStatus::static_empty_fields[] =
  POS(29, 4) "___" /* District */
  POS(11, 5) "____" /* Customer ID */
  POS(44, 5) "_____ " /* Customer Name */

char OrderStatus::static_session_data[] =
  POS(12, 4) "#####" /* Warehouse */

int OrderStatus::static_screen_len = sizeof(OrderStatus::static_screen) - 1;
int OrderStatus::static_empty_fields_len = sizeof(OrderStatus::static_empty_fields) - 1;
int OrderStatus::static_session_data_len = sizeof(OrderStatus::static_session_data) - 1;

/* Delivery */
char const Delivery::static_screen[] =
  POS(1,3) CLEAR_EOS
  POS( 38,3) "Delivery"
  POS( 1,4) "Warehouse:"
  POS( 1,6) "Carrier Number:"

char const Delivery::static_empty_fields[] =
  POS( 17,6) "___" /* Carrier Number */

char Delivery::static_session_data[] =
  POS( 12, 4) "#####" /* Warehouse */

int Delivery::static_screen_len = sizeof(Delivery::static_screen) - 1;
int Delivery::static_empty_fields_len = sizeof(Delivery::static_empty_fields) - 1;
int Delivery::static_session_data_len = sizeof(Delivery::static_session_data) - 1;

/* Stock level */
char const StockLevel::static_screen[] =
  POS( 1, 3) CLEAR_EOS
  POS(35, 3) "Stock-Level"
  POS( 1, 4) "Warehouse:"
  POS(19, 4) "District:"
  POS( 1, 6) "Stock Level Threshold:"
  POS( 1, 8) "Low Stock:"

char const StockLevel::static_empty_fields[] =
  POS( 24,6) "___" /* Threshold */

char StockLevel::static_session_data[] =
  POS( 12,4) "#####" /* Warehouse */
  POS( 29,4) "####" /* District */

int StockLevel::static_screen_len = sizeof(StockLevel::static_screen) - 1;
int StockLevel::static_empty_fields_len = sizeof(StockLevel::static_empty_fields) - 1;
int StockLevel::static_session_data_len = sizeof(StockLevel::static_session_data) - 1;

/* Login */
char const Login::static_screen[] =
  POS( 1, 1) CLEAR_EOS
  POS(30, 3) "Please login."
  POS( 5, 5) "Warehouse:"
  POS(24, 5) "District:"

char const Login::static_empty_fields[] =
  POS( 16,5) "____" /* Warehouse */
  POS( 34,5) "____" /* District */

int Login::static_screen_len = sizeof(Login::static_screen) - 1;
int Login::static_empty_fields_len = sizeof(Login::static_empty_fields) - 1;

/* Menu */
char const Menu::static_screen[] =
  POS(1, 1) CLEAR_EOS
  "(1)New-Order (2)Payment (3)Order-Status (4)Delivery (5)StockLevel (9)Exit"

int Menu::static_screen_len = sizeof(Menu::static_screen) - 1;

/* end string */
char const Screen::end_str[] = "\033[H\n";
int Screen::end_str_len = sizeof(Screen::end_str) - 1;


```

### server.c

```

/*
 * server.c
 *
 * $Revision: 1.11 $
 * $Date: 1999/05/06 21:28:30 $
 * $Log: server.c,v $
 *
 * $TALog: server.c,v $
 * Revision 1.11 1999/05/06 21:28:30 oz
 * - Removed all the .. from the includes
 * - Added -I.. to the makefiles instead
 * - Moved all the thread related code and connection
 * selection to serverMon.c
 * [from r1.10 by delta oz-24309-TPCC-add-oracle8.1-code, r1.5]
 *
 * Revision 1.10 1999/04/19 20:14:48 oz

```

```

* - Moved all the simulated code to server.c
* - Created nulldb.c for compilation with no DB
* [from r1.8 by delta oz-24331-TPCC-move-sim-code-to-common-file, r1.1]
*
* Revision 1.9 1999/04/14 18:11:56 wenjian
* Make changes so that the web client data structures for transactions
* are same as the data structures used in SQL server. It is an important
* change to integrate with MS TPCC kit. It will also avoid copyin/copyout
* for each transaction.
* [from r1.8 by delta wenjian-24134-TPCC-make-client-data-structure-same-as-server, r1.1]
*
* Revision 1.8 1998/12/11 16:37:58 wenjian
* Move some common functions from client/client_utils.c to common/tpcc_utils.c.
* In this version, we only move time_diff_ms(). Need some work in order to
* move other functions like ERROUT.
*
* - Move time_diff_ms() to common/tpcc_utils.c
* [from r1.7 by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.2]
*
* Revision 1.7 1998/12/11 16:14:20 wenjian
* Add code for checking statistic data in a single variable and collecting
* statistic data based on iStatsFrequency.
*
* - Add time_diff_ms()
* [from r1.6 by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.1]
*
* Revision 1.6 1998/11/09 16:59:47 wenjian
* In this revision, most of the changes are related to the directory of header
* files after directory reorganization. Other changes include adding or removing
* files to put them in the right directories. Makefiles are written for NT
* platform so that nmake is working on NT now. Need a top level Makefile for all
* the directories.
* [from r1.5 by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.2]
*
* Revision 1.5 1998/11/09 14:48:24 wenjian
* In an effort to make a new directory structure for TPCC, this delta
* creates two directories: tpcc/client and tpcc/server. All the files
* for this revision are copied from tpcc/sp-tpcc without any change.
* Further change may be needed for some files due to the change of
* the directory structure.
* [added by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.1]
*
* Revision 1.13 1998/11/06 16:10:56 wenjian
* - Change num_mults from 5 to 20
* [from r1.12 by delta wenjian-23646-TPCC-clean-up-source-code, r1.1]
*
* Revision 1.12 1998/10/22 16:25:12 wenjian
* Multi-threaded version.
*
* - Define deliLog for the output of delivery server
* - Stop printing to stderr in err_printf() since it seems to be too
* expensive on NT. Print to file instead.
* [from r1.11 by delta wenjian-23529-TPCC-integrate-with-SQL-server, r1.2]
*
* Revision 1.11 1998/06/17 15:05:48 wenjian
* Somehow, read and write didn't work for socket on NT, although they
* are supposed to work. As a work-around way, use recv and send for
* NT in this revision. We may change them back if the problem is gone.
*
* Use recv and send for socket read and write.
* [from r1.10 by delta wenjian-21750-TPCC-changes-for-porting-on-NT, r1.2]
*
* Revision 1.10 1998/02/17 22:07:06 wenjian
* Define macros to deal with the different function names on NT
* [from r1.9 by delta wenjian-21750-TPCC-changes-for-porting-on-NT, r1.1]
*
* Revision 1.9 1998/01/23 15:08:48 oz
* - Updated the SP TPCC directory to the latest files used
* during the SP tpcc audit.
* [from r1.8 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
*
*/
/*
 * TPCC Server
 *
 * There are currently three versions of the TPCC benchmark
 * implemented here: An Encina monitor based benchmark,
 * an Encina Toolkit based benchmark and a DCE only benchmark.
 *
 * This file, server.c, contains all the code that is common to
 * all the versions. Each server has its own main file:
 * serverMon.c for the monitor server, serverTK.c for the toolkit
 * server and serverDce.c for the dce server.
 *
 * Each server is comprised of three main modules: the server specific
 * one (mentioned above), the common one, in this file, and the
 * server part, which is in the SQL files DBInfo.ec, dbInit.ec,
 * delivery.ec, newOrder.ec, orderStatus.ec, payment.ec, stockLevel.ec.
*/
#include <sys/types.h>
#ifdef WIN32
#include <sys/socket.h>
#include <sys/errno.h>
#else
#include <winsock.h>
#include <io.h>
#endif

```

```

#include <fcntl.h>
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <string.h>
#if defined (solaris)
#include <dce/pthread.h>
#else /* solaris */
#include <pthread.h>
#endif
#include <utils/trace.h>
#include <tpm/mon.h>

#include "common/utilities.h"
#include "server.h"
#include "common/tpcc_type.h"
#include "common/do_tpcc.h"
#include <time.h>

#define DEFINE_SERVER_DEBUG
#include "serverDebug.h"

#if defined(solaris)
extern int errno;
#endif

#ifdef WIN32
#define O_RDONLY _O_RDONLY
#define read(A,B,C) recv(A,B,C,0)
#define open _open
#define close _close
#endif

#define SIM_ERROR_CODE TPCC_SUCCESS

#ifdef WIN32
#define RANDOM rand
#else
#define RANDOM random
#endif

#define TPCC_HOME "/tmp"
#define TIME_PREFIX_LEN 50
extern char sys_errlist[];
extern time_diff_ms(struct timeval *, struct timeval *);

void dprint(char *format, ...);
/*
 * Global variables common to all types of servers
 */
FILE *server_logtrans = NULL;
FILE *deliLog = NULL;
int logtrans = -1;
FILE *dvry_log = NULL; /* FILE structure for delivery log */
int dvry_log_fd = -1; /* File descriptor for delivery log */
int status_log = -1; /* File descriptor for status log */
FILE *deliveryLog = NULL;
FILE *deliveryOut = NULL;
int serverIdNumber = 0; /* The ID of the server
 * This is used to identify output
 */
int serverPid = 0;
int num_mults = 80; /* The number of times the matrices are
 * multiplied (in order to spend some time)
 */
int server_null_test = 0;
int server_init = 0; /* The time (in seconds) the test started
 * This is used by the deferred delivery
 * which reports its times as elapsed time
 * since start time
 */
int null_with_sleep = 1; /* Sleep for some time when simulating trans */

void err_printf(char *format, ...);
void logprintf(char *format, ...);

void open_log_files()
{
/* open DVRY_LOG to keep delivery transactions logs*/
char logname[MAX_STR_LEN], fname[MAX_STR_LEN];
char buffer[MAX_STR_LEN];
char *tpcc_home;
char *log_dir;
int bytes;
int current_fp;
int current;

log_dir = getenv("DELIVERY_LOGS");
if (log_dir == NULL) {
fprintf(stderr, "DELIVERY_LOGS not specified, using %s\n",
TPCC_HOME);
log_dir = TPCC_HOME;
}
}
/*
sprintf(buffer, "%s/status.%d", log_dir, getpid());
status_log = creat(buffer, 0666 );
*/
tpcc_home = getenv("TPCC_HOME");
if (tpcc_home == NULL) {
fprintf(stderr, "TPCC_HOME not specified, using /tmp\n");
tpcc_home = "/tmp";
}

sprintf(fname, "%s/CURRENT", tpcc_home);
current_fp = open(fname, O_RDONLY);
bytes = read(current_fp, buffer, MAX_STR_LEN);

if (bytes == -1) {
fprintf(stderr, "Could not read CURRENT file.\n");
exit(1);
}
buffer[bytes] = '\0';
current = atoi(buffer);
close(current_fp);

dvry_log = NULL;
}
/*
 * logprintf() -- variable argument function used to print error
 * and debug statements. Function is called when
 * any of the debug macros (defined in serverDebug.h)
 * are used.
 */
/*
 * get_thread_id
 * A function that returns the thread ID of the current thread
 */
int get_thread_id()
{
pthread_t thread = pthread_self();
int thread_id = pthread_getunique_np(&thread);
return(thread_id);
}

void print_time_prefix(FILE *file)
{
time_t cur_time;
char time_str[30];

cur_time = time(&cur_time);
strftime(time_str, 29, "%X", localtime(&cur_time));

fprintf(file, "%4d %5d %4d %s - ",
serverIdNumber, serverPid, get_thread_id(), time_str);
}

char *get_time_prefix(char *buffer)
{
time_t cur_time;
char time_str[30];
int len;

cur_time = time(&cur_time);
strftime(time_str, 29, "%X", localtime(&cur_time));

len = sprintf(buffer, "%4d %5d %4d %s - ",
serverIdNumber, serverPid, get_thread_id(), time_str);
if (len >= TIME_PREFIX_LEN) {
fprintf(stderr, "TIME_PREFIX_LEN (%d) too small: %d\n",
TIME_PREFIX_LEN, len);
exit(12);
}
return(buffer);
}

void logprintf(char *format, ...)
{
char formatBuffer[200];
char *fmt = formatBuffer;
int fmtLen;
va_list ap;
va_start(ap, format);

fmtLen = TIME_PREFIX_LEN + strlen(format) + 2;
if (fmtLen > sizeof(formatBuffer)) {
fmt = (char *)malloc(fmtLen);
}
get_time_prefix(fmt);
strcat(fmt, format);
if (server_logtrans) {
vfprintf(server_logtrans, fmt, ap);
flush(server_logtrans);
} else {
vfprintf(stderr, fmt, ap);
}
if (fmt != formatBuffer) free(fmt);
va_end(ap);
}

void err_printf(char *format, ...)

```



```

char formatBuffer[200];
char *fmt = formatBuffer;
int fmtLen;
char timeBuffer[128];
va_list ap;
va_start(ap, format);

fmtLen = TIME_PREFIX_LEN + strlen(format) + 2;
if (fmtLen > sizeof(formatBuffer)) {
    fmt = (char *)malloc(fmtLen);
}
get_time_prefix(fmt);
strcat(fmt, format);
if (server_logtrans) {
    vfprintf(server_logtrans, fmt, ap);
    fflush(server_logtrans);
} else {
    vfprintf(stderr, fmt, ap);
}

if (fmt != formatBuffer) free(fmt);
va_end(ap);
}

/*
 * dprint() -- variable argument function used to print debug
 * statements; for use with DPRINT macro.
 */
void dprint(char *format, ...)
{
    va_list ap;
    va_start(ap, format);

    print_time_prefix(stderr);
    vfprintf(stderr, format, ap);

    va_end(ap);
}

*** Code that has to do with null servers and simulated DBs ***

void mat_mult(int);

#define ROWS 5
#define COLS 5

double matrix_a[ROWS][COLS] = {
    {1.2, 3.4, 2.3, 4.6, 5.2},
    {2.3, 4.5, 1.2, 9.4, 3.1},
    {3.4, 5.2, 3.8, 6.5, 1.6},
    {1.2, 5.3, 6.1, 2.9, 3.8},
    {2.4, 1.2, 3.4, 7.2, 1.0}
};

double matrix_b[ROWS][COLS] = {
    {3.4, 5.9, 2.8, 3.4, 5.6},
    {7.2, 9.3, 4.6, 5.2, 1.3},
    {6.4, 5.2, 8.3, 9.4, 2.3},
    {7.2, 3.4, 6.9, 8.1, 2.3},
    {2.3, 4.5, 7.2, 3.4, 5.8}
};

/* Num of ms to add to RT */
static int rt_increment = 0;

/* Num of ms to add to rt_increment after a certain time. */
static int more_srv_work = 0;

/* how often (in second) to add more_srv_work to rt_increment*/
static int period_to_add_rt = 7*60;

/* how often (in second) to check if there is transaction */
static int period_to_check_tran = 10;

static struct timespec *get_wait_time(struct timespec *timeP, int tran)
{
    int ran = RANDOM() % 1000;
    int wait;

    if (0) {
        if (ran > 998) {
            timeP->tv_sec = 10;
        } else if (ran > 990) {
            timeP->tv_sec = 5;
        } else if (ran > 970) {
            timeP->tv_sec = 1;
        } else {
            timeP->tv_sec = 0;
        }
        timeP->tv_nsec = 50000000;
        if (tran == NEWO_TRANS) {
            timeP->tv_nsec *= 2;
            timeP->tv_sec *= 2;
        }
    } else {
        int time_ms = 0;
        if (tran == NEWO_TRANS) {
            time_ms = 195;
        } else if (tran == PAYMENT_TRANS) {
            time_ms = 50;
        } else if (tran == ORDER_STAT_TRANS) {
            time_ms = 115;
        } else if (tran == STOCK_TRANS) {
            time_ms = 10;
        } else if (tran == DELIVERY_TRANS) {
            time_ms = 0;
        }
        time_ms += rt_increment;
        timeP->tv_sec = 0;
        timeP->tv_nsec = time_ms * 1000000;
    }

    return(timeP);
}

/** A simulated new order transaction */
void sim_new_order(dataP)
newOrder_data_t *dataP;
{
    int i;
    static int next_id = 100;
    struct timespec wait_time;
    static int lasttime = 0;
    struct timeval now;
    static int periods = 0;

    get_local_time(&now);
    if (now.tv_sec - lasttime > period_to_check_tran) {
        static trans[3]; /* Keep the counts for the last 5 periods */
        lasttime = now.tv_sec;

        if ((trans[1] - trans[0] < 2) &&
            (trans[2] - trans[1] < 2)) {
            rt_increment = 0;
            periods = 0;
            more_srv_work = getenv("TPCC_MORE_SERVER_WORK") ?
                atoi(getenv("TPCC_MORE_SERVER_WORK")) : 0;
            err_printf("Nothing much happening - resetting test\n");
        } else {
            periods++;
            if (periods % (period_to_add_rt / period_to_check_tran) == 0) {
                rt_increment += more_srv_work;
                err_printf("rt_increment now %d\n", rt_increment);
            }
            trans[0] = trans[1];
            trans[1] = trans[2];
            trans[2] = next_id;
        }
    }

    if (null_with_sleep)
        pthread_delay_np(get_wait_time(&wait_time, NEWO_TRANS));

    mat_mult(num_mults);
    sprintf((char *)dataP->c_last, "BARBARBAR");
    sprintf((char *)dataP->c_credit, "GC");
    dataP->c_discount = 0.33;
    dataP->o_id = next_id++;
    sprintf((char *)dataP->entry_date, "17-12-1995.12:33:56");
    dataP->total = 99.1;
    dataP->w_tax = 0.729;
    dataP->d_tax = 0.15;
    for (i=0; i<dataP->o_ol_cnt; i++) {
        dataP->item[i].price = dataP->item[i].ol_i_id % 1000;
        sprintf((char *)dataP->item[i].name_i, "item %d", i);
        dataP->item[i].s_quantity = i;
        dataP->item[i].brand_generic[0] = i%2 ? 'O' : 'E';
        dataP->item[i].brand_generic[1] = '0';
        dataP->item[i].ol_amount =
            dataP->item[i].price * dataP->item[i].ol_quantity;
    }

    if ((dataP->item[dataP->o_ol_cnt - 1].ol_i_id < 1) ||
        (dataP->item[dataP->o_ol_cnt - 1].ol_i_id > 1000000)) {
        dataP->header.returncode = INVALID_NEWO;
    } else if (RANDOM() % 90 == 0) {
        dataP->header.returncode = SIM_ERROR_CODE;
    } else {
        dataP->header.returncode = TPCC_SUCCESS;
    }
    return;
}

/** A simulated payment transaction */
void sim_payment(dataP)
payment_data_t *dataP;
{
    struct timespec wait_time;

    if (null_with_sleep)
        pthread_delay_np(get_wait_time(&wait_time, PAYMENT_TRANS));
    mat_mult(num_mults);

    dataP->c_id = 1;
    dataP->c_credit_lim = 100.9;
    dataP->c_discount = 0.2;
}

```

```

dataP->c_balance = 11.1;

sprintf(char *)dataP->c_first, "%-16s", "c_first");
sprintf(char *)dataP->c_middle, "%-2s", "MI");
sprintf(char *)dataP->c_last, "%-16s", "c_last");
sprintf(char *)dataP->c_street_1, "%-20s", "c_street_1");
sprintf(char *)dataP->c_street_2, "%-20s", "c_street_2");
sprintf(char *)dataP->c_city, "%-20s", "c_city");
sprintf(char *)dataP->c_state, "%-2s", "PA");
sprintf(char *)dataP->c_zip, "%-9s", "152111111");
sprintf(char *)dataP->c_phone, "%-16s", "6522573904218222");
sprintf(char *)dataP->c_date, "%-19s", "28-11-1995");
sprintf(char *)dataP->c_credit, "%-2s", "GC");
sprintf(char *)dataP->pay_date, "%-19s", "17-12-1995.12:39:13");
sprintf(char *)dataP->d_street_1, "%-20s", "d_street_1");
sprintf(char *)dataP->d_street_2, "%-20s", "d_street_2");
sprintf(char *)dataP->d_city, "%-20s", "d_city");
sprintf(char *)dataP->d_state, "%-2s", "PA");
sprintf(char *)dataP->d_zip, "%-9s", "152111111");
sprintf(char *)dataP->w_street_1, "%-20s", "w_street_1");
sprintf(char *)dataP->w_street_2, "%-20s", "w_street_2");
sprintf(char *)dataP->w_city, "%-20s", "w_city");
sprintf(char *)dataP->w_state, "%-2s", "OH");
sprintf(char *)dataP->w_zip, "%-9s", "142411111");

if(RANDOM() % 70 == 0) {
    dataP->header.returncode = SIM_ERROR_CODE;
} else {
    dataP->header.returncode = TPCC_SUCCESS;
}
}

```

```

/** A simulated stock level transaction */
void sim_stock_level(dataP)
stockLevel_data_t *dataP;
{
    struct timespec wait_time;

    if(null_with_sleep)
        pthread_delay_np(get_wait_time(&wait_time, STOCK_TRANS));

    mat_mult(num_mults);

    dataP->stock_count = 12;
    if(RANDOM() % 80 == 0) {
        dataP->header.returncode = SIM_ERROR_CODE;
    } else {
        dataP->header.returncode = TPCC_SUCCESS;
    }
}

```

```

/** A simulated delivery transaction */
void sim_delivery(dataP)
delivery_data_t *dataP;
{
    struct timespec wait_time;

    if(null_with_sleep)
        pthread_delay_np(get_wait_time(&wait_time, DELIVERY_TRANS));

    dataP->start_queue = 2.2;
    dataP->header.returncode = TPCC_SUCCESS;
}

```

```

/** A simulated order status transaction */
void sim_order_status(dataP)
orderStatus_data_t *dataP;
{
    int i;
    struct timespec wait_time;

    if(null_with_sleep)
        pthread_delay_np(get_wait_time(&wait_time, ORDER_STAT_TRANS));

    mat_mult(num_mults);

    dataP->c_id = dataP->c_id ? dataP->c_id : 99;
    strcpy(char *)dataP->c_first, "Jerome");
    strcpy(char *)dataP->c_middle, "LB");
    strcpy(char *)dataP->c_last, "Trevoe");
    dataP->c_balance = 90.78;
    dataP->o_id = 99;
    strcpy(char *)dataP->entry_date, "06-12-1995.16:42:28");
    dataP->o_carrier_id = 9;
    dataP->o_ol_cnt = 7;

    for (i=0; i<dataP->o_ol_cnt; i++) {
        dataP->item[i].ol_supply_w_id = 1;
        dataP->item[i].ol_i_id = dataP->w_id * 10 + dataP->d_id;
        dataP->item[i].ol_quantity = 10 * (i+1);
        dataP->item[i].ol_amount = dataP->item[i].ol_quantity * 10.1;
        strcpy(char *)dataP->item[i].delivery_date, "NOT DELIVR");
    }
}

```

```

if(RANDOM() % 90 == 0) {
    dataP->header.returncode = SIM_ERROR_CODE;
}

```

```

} else {
    dataP->header.returncode = 0;
}
}

/*
 * mat_mult
 *
 * Multiply the above two matrices
 */
static void mat_mult(iter)
    int iter;
{
    float res[ROWS][COLS];
    int i, j, k;
    int a_num_rows = ROWS;
    int a_num_columns = COLS;
    int b_num_rows = ROWS;
    int b_num_columns = COLS;

    for (; iter>0; iter--) {
        for (i=0; i<a_num_rows; i++) {
            for (j=0; j<b_num_columns; j++) {
                res[i][j] = 0;
                for (k=0; k<b_num_rows; k++) {
                    res[i][j] += matrix_a[i][k] * matrix_b[k][j];
                }
                matrix_a[i][j] = res[i][j];
            }
        }
        pthread_yield();
    }
}

```

## server.h

```

/*
 * server.h
 *
 * $Revision: 1.11 $
 * $Date: 1999/05/06 21:28:31 $
 * $Log: $
 *
 *
 * $TALog: server.h.v $
 * Revision 1.11 1999/05/06 21:28:31 oz
 * - Removed all the .. from the includes
 * - Added -I. to the makefiles instead
 * - Moved all the thread related code and connection
 * selection to serverMon.c
 * [from r1.9 by delta oz-24309-TPCC-add-oracle8.1-code, r1.5]
 *
 * Revision 1.9 1999/01/12 20:52:59 wenjian
 * Define MAPOBJNAMEFORMAT so that the server processes and dll can communicate
 * via the shared file mappings.
 * [from r1.8 by delta wenjian-23856-TPCC-integrate-with-NT-performance-monitor, r1.1]
 *
 * Revision 1.8 1998/12/14 20:27:57 wenjian
 * Made corresponding changes due to data structure change of tran_info_t.
 *
 * - change server_tran_t
 * [from r1.7 by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.3]
 *
 * Revision 1.7 1998/12/11 16:14:20 wenjian
 * Add code for checking statistic data in a single variable and collecting
 * statistic data based on iStatsFrequency.
 *
 * - Add server_tran_t and server_info_t
 * [from r1.6 by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.1]
 *
 * Revision 1.6 1998/11/09 16:59:48 wenjian
 * In this revision, most of the changes are related to the directory of header
 * files after directory reorganization. Other changes include adding or removing
 * files to put them in the right directories. Makefiles are written for NT
 * platform so that nmake is working on NT now. Need a top level Makefile for all
 * the directories.
 * [from r1.5 by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.2]
 *
 * Revision 1.5 1998/11/09 14:48:25 wenjian
 * In an effort to make a new directory structure for TPCC, this delta
 * creates two directories: tpcc/client and tpcc/server. All the files
 * for this revision are copied from tpcc/sp-tpcc without any change.
 * Further change may be needed for some files due to the change of
 * the directory structure.
 * [added by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.1]
 *
 * Revision 1.9 1998/10/22 15:33:05 wenjian
 * Make changes to Encina server code to connect with SQL server and add
 * callsql.c and sql directory.
 *
 * Add delivery_sql_t to deal with SYSTEMTIME struct used in SQL
 * [from r1.7 by delta wenjian-23529-TPCC-integrate-with-SQL-server, r1.1]
 *
 * Revision 1.7 1998/01/23 15:08:50 oz
 * - Updated the SP TPCC directory to the latest files used
 * during the SP tpcc audit.
 * [from r1.6 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]

```

```

*
*
*/
** server.h **

** Declarations common to all the server modules **

#ifndef TPCC_SERVER_H
#define TPCC_SERVER_H

#include "common/tpcc_type.h"

#define get_dbname_from_id(i) rmlist[i].dbName
#define MAPOBJNAMEFORMAT "srv_%s_PA%d"

#ifdef WIN32
typedef struct {
    delivery_data_t data;
    SYSTEMTIME queue_time;
} delivery_sql_t;
#endif

typedef enum {
    mon_server = 11
} server_type_t;

typedef struct {
    int num;
    double RTtotal;
    int RTcount;
} server_tran_t;

typedef struct {
    server_tran_t tran[MAX_TRAN_TYPE + 1];
    int total_trans;
} server_info_t;

extern int server_no_db;
extern int serverIdNumber;
extern int server_init;
extern server_type_t server_type;
extern int get_db_for_wh(int);

#endif /* TPCC_SERVER_H */

*
* serverDebug.h
*
* $Revision: 1.5 $
* $Date: 1998/11/09 14:48:25 $
* $Log: serverDebug.h,v $
* Revision 4.4 95/05/16 10:55:40 tpc (TPCC Benchmark)
* Added necessary RCS ident strings
*
*/
#ifndef SERVER_DEBUG
#define SERVER_DEBUG

#include <utils/trace.h>

#ifdef DEFINE_SERVER_DEBUG
long serverDebug = 0;
#else
extern long serverDebug;
#endif

#ifdef TRACE_TRANS
#define TRACETRAN(list) logprintf list
#else
#define TRACETRAN(list)
#endif

#ifdef DEBUG_SERVER
#define AUDITLOG(list) if (serverDebug & AUDIT_TRANS) UNCOND_EVENT list
#define NEWOLOG(list) if (serverDebug & DBG_NEWO) err_printf list
#define PAYLOG(list) if (serverDebug & DBG_PAY) err_printf list
#define OSLOG(list) if (serverDebug & DBG_OS) err_printf list
#define STKLOG(list) if (serverDebug & DBG_STK) err_printf list
#define DEBUGP(list) if (serverDebug) err_printf list
#else
#define AUDITLOG(list)
#define NEWOLOG(list)
#define PAYLOG(list)
#define OSLOG(list)
#define STKLOG(list)
#define DELLOG(list)
#define DEBUGP(list)
#endif

#define ERRLOG(list) err_printf list
#define SQL_RET_CODE(var, code) var = (code)

```

## serverDebug.h

```

/* Fix DPRINT to write on a debugging unit that can get set differently
for delivery */

#ifndef UNIT_TEST
#define DPRINT(list) dprint list
#define DELPRINT(list) delprint list
#else
#define DPRINT(list)
#define DELPRINT(list)
#endif

#define DBG_NEWO      0x0001
#define DBG_PAY      0x0002
#define DBG_OS       0x0004
#define DBG_STK      0x0008
#define DBG_DEL      0x0010
#define DBG_ERR      0x0020
#define AUDIT_TRANS  0x0100

#endif /* SERVER_DEBUG */

*
* serverMon.c
*
* $Revision: 1.23 $
* $Date: 1999/05/28 19:44:17 $
* $Log: serverEncina.c,v $
*
* $TALog: serverMon.c,v $
* Revision 1.23 1999/05/28 19:44:17 wenjian
* Add create_null_connection and clean_null_connection so that
* we can run with NULL DB.
* [from r1.22 by delta wenjian-24433-TPCC-clean-up-and-update, r1.4]
*
* Revision 1.22 1999/05/28 14:30:22 wenjian
* - Fix a bug for calling get_thread_data()
* [from r1.21 by delta wenjian-24433-TPCC-clean-up-and-update, r1.3]
*
* Revision 1.21 1999/05/26 16:29:59 wenjian
* Sync with Austin code, and sync code for Oracle DB and SQL server.
*
* - Fix some minor bugs
* [from r1.20 by delta wenjian-24433-TPCC-clean-up-and-update, r1.2]
*
* Revision 1.20 1999/05/06 21:28:31 oz
* - Removed all the .. from the includes
* - Added -I. to the makefiles instead
* - Moved all the thread related code and connection
* selection to serverMon.c
*
* - Added connection and thread related code.
* - pre_DB and post_DB always get called.
* - get_db_ready has only 2 parameters
* - added tran_info_t, total_tran_count_t, and thread_info_t
* - Added clean_thread_data and get_thread_data
* - Preallocate all the connections if necessary after
* initializing the DB.
* - A connection is created by calling the DB specific
* create_connection() which returns a void* handle to
* be passed to all the transactions.
* [from r1.18 by delta oz-24309-TPCC-add-oracle8.1-code, r1.5]
*
* Revision 1.18 1999/04/20 15:11:28 oz
* [merge of changes from 1.13 to 1.17 into 1.12]
*
* Revision 1.17 1999/04/19 20:14:49 oz
* - Moved all the simulated code to server.c
* - Created nulldb.c for compilation with no DB
* [from r1.13 by delta oz-24331-TPCC-move-sim-code-to-common-file, r1.1]
*
* Revision 1.12 1999/01/12 20:53:00 wenjian
* - Call initialization function perfSrvDataInit to create the shared file
* mapping for this server process
* - Change server_info to pServerInfo
* [from r1.11 by delta wenjian-23856-TPCC-integrate-with-NT-performance-monitor, r1.1]
*
* Revision 1.11 1998/12/14 20:27:58 wenjian
* Made corresponding changes due to data structure change of tran_info_t.
*
* - Made changes for server_tran_t
* - Add tran_type to FUNCTION_BEGIN, FUNCTION_END, pre_DB, and post_DB
* [from r1.10 by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.3]
*
* Revision 1.10 1998/12/11 16:14:20 wenjian
* Add code for checking statistic data in a single variable and collecting
* statistic data based on iStatsFrequency.
*
* - Change pre_oracle to pre_DB, post_oracle to post_DB
* - Add code to collect server RT in a global var
* [from r1.9 by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.1]
*
* Revision 1.9 1998/12/08 18:55:22 wenjian
* - Remove "statsFrequency=" command line argument.
* - Check rpc header for stats
* [from r1.8 by delta wenjian-23785-TPCC-pass-statsFrequency-from-client-to-server, r1.1]
*

```

## serverMon.c

\* Revision 1.8 1998/12/07 20:04:16 wenjian  
 \* Remove interfaces for explicit bindings  
 \* [from r1.7 by delta wenjian-23742-TPCC-update-with-Raleigh-code, r1.2]  
 \*

\* Revision 1.7 1998/11/24 21:46:03 wenjian  
 \* - Remove COLLECT\_TIMESTAMPS; use command line argument iStatsFrequency instead  
 \* - Take care of MULTIPLE\_INTERFACE and SINGLE\_INTERFACE  
 \* [from r1.6 by delta wenjian-23742-TPCC-update-with-Raleigh-code, r1.1]  
 \*

\* Revision 1.6 1998/11/09 16:59:48 wenjian  
 \* In this revision, most of the changes are related to the directory of header files after directory reorganization. Other changes include adding or removing files to put them in the right directories. Makefiles are written for NT platform so that nmake is working on NT now. Need a top level Makefile for all the directories.  
 \* [from r1.5 by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.2]  
 \*

\* Revision 1.5 1998/11/09 14:48:25 wenjian  
 \* In an effort to make a new directory structure for TPCC, this delta creates two directories: tpcc/client and tpcc/server. All the files for this revision are copied from tpcc/sp-tpcc without any change.  
 \* Further change may be needed for some files due to the change of the directory structure.  
 \* [added by delta wenjian-23677-TPCC-reorganize-directory-structure, r1.1]  
 \*

\* Revision 1.36 1998/11/06 16:10:56 wenjian  
 \* - Increase the range for the number of threads communicating to the DB.  
 \* - Print warning if the number of threads is out of the range.  
 \* [from r1.35 by delta wenjian-23646-TPCC-clean-up-source-code, r1.1]  
 \*

\* Revision 1.35 1998/10/22 21:30:47 wenjian  
 \* [merge of changes from 1.20 to 1.29 into 1.34]  
 \*

\* Revision 1.29 1998/10/08 14:18:02 dongfeng  
 \* Add codes for doing web-based tpcc.  
 \* [from r1.20 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.1]  
 \*

\* Revision 1.34 1998/10/22 19:43:38 wenjian  
 \* [merge of changes from 1.28 to 1.32 into 1.30]  
 \*

\* Revision 1.32 1998/10/22 16:25:13 wenjian  
 \* Multi-threaded version.  
 \*

\* - Open server\_logtrans for err\_printf() and deliLog for delivery server  
 \* - Call get\_time\_init() for get\_local\_time  
 \* - Changes for multi-threaded version  
 \* [from r1.31 by delta wenjian-23529-TPCC-integrate-with-SQL-server, r1.2]  
 \*

\* Revision 1.31 1998/10/22 15:33:05 wenjian  
 \* Make changes to Encina server code to connect with SQL server and add callsql.c and sql directory.  
 \*

\* - Add SYSTEMTIME \*queue\_time to deferred\_dvry\_t for NT  
 \* - Allocate space and set value for queue\_time  
 \* - Pass delivery\_sql\_t instead of delivery\_data\_t for SQLdel on NT  
 \* [from r1.28 by delta wenjian-23529-TPCC-integrate-with-SQL-server, r1.1]  
 \*

\* Revision 1.30 1998/10/08 18:03:03 gerstl  
 \* Changes to allow configurations where some servers only service specific transaction types. Split transaction interfaces by type.  
 \*

\* Transaction interface support is based upon a bitmap passed to the server.  
 \* [from r1.28 by delta gerstl-23515-TPCC-allow-separate-online-transaction-interfaces, r1.1]  
 \*

\* Revision 1.28 1998/10/07 15:49:49 gerstl  
 \* [merge of changes from 1.22 to 1.26 into 1.27]  
 \*

\* Revision 1.26 1998/09/03 20:22:02 wenjian  
 \* Sync with Austin code: mostly use servrMon.c in Austin.  
 \* [from r1.25 by delta wenjian-23183-TPCC-sync-AIX-code-with-Austin, r1.4]  
 \*

\* Revision 1.25 1998/09/03 16:07:12 wenjian  
 \* Remove UNCOND\_EVENT which is not in austin code.  
 \* [from r1.24 by delta wenjian-23183-TPCC-sync-AIX-code-with-Austin, r1.3]  
 \*

\* Revision 1.24 1998/09/02 15:43:30 wenjian  
 \* Define num\_worker\_threads.  
 \* [from r1.23 by delta wenjian-23183-TPCC-sync-AIX-code-with-Austin, r1.2]  
 \*

\* Revision 1.23 1998/08/28 18:30:02 wenjian  
 \* This delta sync the TPCC code with Austin.  
 \*

\* - Take care of 1 thread per PA  
 \* - Update with Austin code  
 \* - Remove the old code starting from #ifdef THIS\_WAS\_THE\_OLD\_CODE  
 \* - Remove impTPCCDvryInfo()  
 \* [from r1.22 by delta wenjian-23183-TPCC-sync-AIX-code-with-Austin, r1.1]  
 \*

\* Revision 1.27 1998/09/26 10:27:33 oz  
 \* Changes for NT.  
 \* [from r1.22 by delta oz-23339-TPCC-update-for-NT, r1.1]  
 \*

\* Revision 1.22 1998/08/18 14:38:43 wenjian  
 \* Minor change  
 \* [from r1.20 by delta wenjian-21750-TPCC-changes-for-porting-on-NT, r1.4]  
 \*

\* Revision 1.20 1998/02/17 22:07:06 wenjian

\* Minor changes for NT  
 \* [from r1.19 by delta wenjian-21750-TPCC-changes-for-porting-on-NT, r1.1]  
 \*

\* Revision 1.19 1998/01/30 15:12:28 oz  
 \* - Remove the explicit binding functions from the tidl files and from serverMon.c  
 \* [from r1.18 by delta oz-21697-TPCC-remove-explicit-binding-code, r1.2]  
 \*

\* Revision 1.18 1998/01/24 14:17:06 oz  
 \* - User server name to identify server and name delivery file  
 \* - Use env variable HOME instead of /home/encina if HOME is set  
 \*

\* - Removed the machine list  
 \* The server ID is computed from the first number found in the host name  
 \* [from r1.17 by delta oz-21687-TPCC-use-server-name-to-identify-process, r1.1]  
 \*

\* Revision 1.17 1998/01/23 21:59:00 oz  
 \* - In order to simplify the Encina TPCC code: Merge the four online transactions into 1 interface  
 \* - Moved all the scripts to a scripts subdirectory  
 \* - Removed unused files  
 \* [from r1.16 by delta oz-21671-TPCC-merge-online-transaction-interfaces, r1.1]  
 \*

\* Revision 1.16 1998/01/23 15:08:53 oz  
 \* - Updated the SP TPCC directory to the latest files used during the SP tpcc audit.  
 \* [from r1.15 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]  
 \*

serverMon.c

\* Code that is monitor specific.  
 \*/

```
#include <sys/types.h>
#ifdef WIN32
#include <unistd.h>
#else
#include <process.h>
#include <winsock.h>
#endif
#include <stdio.h>
#include <stdarg.h>
#include <time.h>
#include <tmx/tmx.h>
#include <tc/tc.h>
#include <tpm/mon/mon_server.h>

#ifdef solaris
#include <dce/pthread.h>
#else /* solaris */
#include <pthread.h>
#endif /* solaris */
#include <utils/trace.h>

#include "common/databuf.h"
#include "common/utilities.h"
#include "serverDebug.h"
#include "common/delivery.h"
#ifdef MULTIPLE_INTERFACE
#include "common/neworder.h"
#include "common/payment.h"
#include "common/stocklevel.h"
#include "common/orderstatus.h"
#else
#include "common/tpcc_trans.h"
#endif
#include "server.h"

#define FUNCTION_BEGIN(name, dataP, tran_type, infoP) \
pre_DB(name, &(dataP)->header, tran_type, infoP);

#define FUNCTION_END(name, dataP, tran_type, infoP) \
post_DB(name, &(dataP)->header, tran_type, infoP); \

#ifdef WIN32
#define CASECMP(x,y) _stricmp(x,y) == 0
#define getpid _getpid
extern void TPCexit();
#else
#define CASECMP(x,y) streasecmp(x,y) == 0
extern void TPCexit();
#endif

extern void *create_connection();
extern void clean_connection(void *);
extern int get_db_ready(char *, int);

inModule("serverMon");

static void start_deferred_delivery_threads();
static void queue_delivery(delivery_data_t *dataP);
static void *create_null_connection();
static void clean_null_connection(void *ptr);

extern int server_null_test;
```

```

extern void err_printf(char *format, ...);
extern int get_db_ready(char *, int);
extern void logprintf(char *format, ...);
extern server_info_t *perfSrvDataInit(char *, int);
extern void *start_bg_thread();

static void get_mon_server_env();

server_type_t server_type = mon_server;

char *tpcc_serverName = NULL;
char *dbName = NULL;
int total_num_warehouses;
int num_deferred_dvry_threads = 1;
int num_worker_threads = 1;
int dvry_queue_size = 3000;
server_info_t *pServerInfo = NULL;
char oracle_home[256]; /* will be used in tpccpl.c */

typedef struct {
    pthread_mutex_t    lock;
    pthread_cond_t     q_cond;
    pthread_cond_t     work_cond;

    int                num_waiters; /* Number of new requests waiting */

    int                head, tail;
    int                allocated; /* Total size of the queue */
    int                size; /* Num elements currently there */
#ifdef WIN32
    SYSTEMTIME         *queue_time;
#endif
    delivery_data_t    *data;
} deferred_dvry_t;

static deferred_dvry_t deferred_dvry_data;
#define MAX_DVRY_QUEUE deferred_dvry_data.allocated

/*
 * Information about one transaction type
 */
typedef struct {
    int num;
    int errs;
    double RT;
} tran_info_t;

/*
 * total_tran_count_t
 *
 * structure that holds the total count of transaction of each type
 * as well as the reponse times.
 */
typedef struct {
    tran_info_t tran[MAX_TRAN_TYPE + 1];
    int errors;
} total_tran_count_t;

typedef struct {
    void *cnP; /* DB specific connection to be used by this thread */

    int calls; /* Number of times it was used */
    int errors; /* Total number of errors on this connection */
    int calls_last_err; /* Number of calls when the last error occurred */
    int consecutive_errs; /* Number of consecutive errs */
    int connect_time; /* Time (seconds) connections was created */

    /* For debug */
    int state; /* State of the connection */
    struct timeval tran_time; /* Time this tran started */
    int cur_tran_type;
    void *cur_tran_dataP;
    total_tran_count_t stat;
    int printed;
} thread_info_t;

#define SVR_STATE_NONE 0
#define SVR_STATE_SENT 1
#define SVR_STATE_REPLIED 2
#define SVR_STATE_ERR 3

/* Connection related data structures */
static void clean_thread_data(void *ptr);

pthread_key_t thread_key;
pthread_mutex_t init_lock;
thread_info_t *info_array = NULL; /* Array of thread data */
int num_threads = 0; /* number of threads that have already been init */
int next_thread = 0; /* next thread id: next entry in the array */

int preallocate_cn = 1; /* Should all connections be preallocated */
int num_connections = 0;
int num_allocated = 0;

static thread_info_t *get_thread_data();

static void display_mon_env()
{
    char *env_str;
    char envMsg[64];

#define DISPLAY_ENV_VAR(var) \
    if ((env_str = getenv(var)) != NULL) { \
        UNCOND_EVENT("%s == %s\n", var, env_str); \
    } else { \
        UNCOND_EVENT("%s not set\n", var); \
    }

    UNCOND_EVENT("TPCC Server display env. ID: %d\n", serverIdNumber);

    /*
     * For debugging purpose: have the first PA
     * display the following information
     */

    if ((serverIdNumber & 0xff) == 0) {
        DISPLAY_ENV_VAR("RPC_SUPPORTED_PROTSEQS");
        DISPLAY_ENV_VAR("RPC_UNSUPPORTED_NETADDRS");
        DISPLAY_ENV_VAR("ENCINA_BINDING_TIMEOUT");
        DISPLAY_ENV_VAR("ENCINA_THREAD_POOL_QUEUE_LENGTH");
        DISPLAY_ENV_VAR("ENCINA_THREAD_POOL_QUEUE_LENGTH");
        DISPLAY_ENV_VAR("ENCINA_TPOOL_SIZE");
        DISPLAY_ENV_VAR("ENCINA_RPC_THREAD_STACK_SIZE");
    }
}

/* get_server_index() -- This is used for debug purposes only
 *
 * Return the server index for this server.
 * By convention, all the client machines hvae similar
 * names with different numbers, as in client1 client2, ...
 * If the convention is followed the server index is the first
 * number found. Otherwise, it is 0.
 */
static int get_server_index()
{
    int i, ind;
    char host_name[128];
    if (0 == gethostname(host_name, sizeof(host_name))) {
        err_printf("Machine is on host %s\n", host_name);
        ind = strtospn(host_name, "0123456789");
        return(atol(host_name + ind));
    }

    return(0);
}

static parse_cmd_line(int argc, char *argv[], char **scheduling, int *interface_type)
{
    int nextInd = 1;
    char usageStr[128];
    int envRetrieval;
    if ((nextInd + 3) > argc) {
        sprintf(usageStr,
            "Not enough parameters. Usage: %s [-no_db] interfaces schedulingPolicy envRetrievalFlag [dvry=#] [debugFlag] [db:<dbName>] [cn=#].", argv[0]);
        fprintf(stderr, "%s\n", usageStr);
        mon_TerminateServer(usageStr);
    } else {
        if (strcmp(argv[nextInd], "-no_db") == 0) {
            server_null_test = 1;
            fprintf(stderr, "----= NULL test ----\n");
            nextInd++;
        }
        *interface_type = strtol(argv[nextInd++], NULL, 0);
        *scheduling = argv[nextInd++];
        envRetrieval = atoi(argv[nextInd++]);

        while (nextInd < argc) {
            if (strcmp(argv[nextInd], "db:") == 0) {
                dbName = argv[nextInd] + 3;
                nextInd++;
            } else if (strcmp(argv[nextInd], "dvry=") == 0) {
                num_deferred_dvry_threads = atoi(argv[nextInd] + 5);
                if (num_deferred_dvry_threads < 0 || num_deferred_dvry_threads > 200) {
                    err_printf("num_deferred_dvry_threads was %d (>200), reset to 10\n",
                        num_deferred_dvry_threads);
                    num_deferred_dvry_threads = 10;
                }
                nextInd++;
            } else if (strcmp(argv[nextInd], "dvryQ=") == 0) {
                dvry_queue_size = atoi(argv[nextInd] + 6);
                if (dvry_queue_size < 1 || dvry_queue_size > 200000)
                    dvry_queue_size = 10;
                nextInd++;
            }
        }
    }
}

```

```

    } else {
        serverDebug = atol(argv[nextInd++]);
    }
}
}
}

static void set_scheduling(char *scheduling)
{
    mon_paAccess_t paAccess;
    UNCOND_EVENT("Setting Scheduling Policy: %s\n", scheduling);

    if(CASECMP(scheduling, "MON_CONCURRENT_SHARED")) {
        paAccess = MON_CONCURRENT_SHARED;
    } else if(CASECMP(scheduling, "MON_EXCLUSIVE")) {
        num_deferred_dvry_threads = 1;
        paAccess = MON_EXCLUSIVE;
    } else if(CASECMP(scheduling, "MON_SHARED")) {
        num_deferred_dvry_threads = 1;
        paAccess = MON_SHARED;
    } else {
        err_printf("Invalid Policy: %s\n", scheduling);
        mon_TerminateServer("Invalid scheduling policy specified.");
    }

    ENCINA_CALL("mon_SetSchedulingPolicy",
        mon_SetSchedulingPolicy(paAccess));
}

static void register_interfaces(int interface_type)
{
    extern FILE *deliLog;
    char *env_str;
    int env_val;

    UNCOND_EVENT("Registering interfaces\n");

    num_worker_threads = 0;

    /* interface_type is a bitmap of the interfaces this
    * server needs to support.
    */
#ifdef MULTIPLE_INTERFACE
    if (interface_type & NEW_INTERFACE) {
        ENCINA_CALL("mon_InitServerInterface",
            mon_InitServerInterface(MON_SERVER_INTERFACE(neworder,1,0));
    }
    if (interface_type & PAYMENT_INTERFACE) {
        ENCINA_CALL("mon_InitServerInterface",
            mon_InitServerInterface(MON_SERVER_INTERFACE(payment,1,0));
    }
    if (interface_type & ORDER_STAT_INTERFACE) {
        ENCINA_CALL("mon_InitServerInterface",
            mon_InitServerInterface(MON_SERVER_INTERFACE(orderstatus,1,0));
    }
    if (interface_type & STOCK_INTERFACE) {
        ENCINA_CALL("mon_InitServerInterface",
            mon_InitServerInterface(MON_SERVER_INTERFACE(stocklevel,1,0));
    }
#else
    if (interface_type & ONLINE_INTERFACES) {
        ENCINA_CALL("mon_InitServerInterface",
            mon_InitServerInterface(MON_SERVER_INTERFACE(tpccTrans,1,0));
    }
#endif

    if (interface_type & DELIVERY_INTERFACE) {
#ifdef WIN32
        deliLog = fopen("deliLog.out", "w");
#endif
        if (num_deferred_dvry_threads > 0) {
            start_deferred_delivery_threads();
        }
        ENCINA_CALL("mon_InitServerInterface",
            mon_InitServerInterface(MON_SERVER_INTERFACE(delivery,1,0));
    } else {
        num_deferred_dvry_threads = 0;
    }

    /* ENCINA_TPOOL_SIZE and ENCINA_APPL_TPOOL_SIZE
    * are set in tpccCommon.tcl for each
    * server started. If we are delivery only, we don't care
    * about it, otherwise we need to adjust num_worker_threads
    */
    if (interface_type & ONLINE_INTERFACES) {
        if ((env_str = getenv("ENCINA_APPL_TPOOL_SIZE")) != NULL) {
            env_val = atol(env_str);
            if (env_val >= 0 && env_val < 1000)
                num_worker_threads += env_val;
            else {
                err_printf("ENCINA_APPL_TPOOL_SIZE was %d, reset to 10\n", env_val);
                num_worker_threads += 10;
            }
        }
        if ((env_str = getenv("ENCINA_TPOOL_SIZE")) != NULL) {
            env_val = atol(env_str);
            if (env_val >= 0 && env_val < 1000)
                num_worker_threads += env_val;
        }
    }
}

```

```

    else {
        err_printf("ENCINA_TPOOL_SIZE was %d, reset to 10\n", env_val);
        num_worker_threads += 5;
    }
}
}
}

if (num_worker_threads < 1) num_worker_threads = 1;
}

}

void main(argc,argv)
int argc;
char *argv[];
{
    extern FILE *server_logtrans;
    int rc;
    int pa_num;
    char *scheduling = "";
    int rmlid;
    char intermediary[256];
    extern int serverPid;
    int interface_type = ALL_INTERFACE;
    int status;

    inFunction("server_Init");

    /* hard code first for a quick test */
    /* getenv didn't work, though we have ORACLE_HOME defined */
    /* strcpy(oracle_home,getenv("ORACLE_HOME")); */
    strcpy(oracle_home, "/home/oracle815/app/oracle/product/8.1.5");

    server_logtrans = fopen("server_print.out", "w");

    get_time_init();

    serverPid = getpid();
    UNCOND_EVENT("TPCC Server Starting\n");

    /* Use the top 8 bits of the serverIdNumber to store the server index */
    serverIdNumber = (get_server_index() & 0xff) * 1000;

    parse_cmd_line(argc, argv, &scheduling, &interface_type);

    display_mon_env();

    DEBUGP("Debug level set at %d\n", serverDebug);

    DEBUGP("Creating thread data key");
    if(status = pthread_keycreate(&thread_key, clean_thread_data)) {
        fprintf(stderr, "init_global_data : pthread_keycreate failed: %d\n", status);
        mon_TerminateServer("Cannot create a key for the thread data");
    }

    mon_RetrieveEnable(FALSE);

    err_printf("Setting scheduling %s\n", scheduling);
    set_scheduling(scheduling);
    err_printf("Registering interfaces \n");
    register_interfaces(interface_type);

    err_printf("Calling mon_init\n");
    ENCINA_CALL("mon_InitServer", mon_InitServer());
    ENCINA_CALL("mon_SetHandleCacheRefreshInterval",
        mon_SetHandleCacheRefreshInterval(300));

    pa_num = mon_RetrievePaNum();
    tpcc_serverName = mon_RetrieveServerId();
    if (pa_num > 0)
        serverIdNumber += pa_num;
    err_printf("PA Number %d, serverId %d (%s)\n",
        pa_num, serverIdNumber, tpcc_serverName);

    num_connections = num_deferred_dvry_threads + num_worker_threads;

    if ((rc = get_db_ready(dbName, 0)) != 0) {
        char msg[128];
        sprintf(msg, "failed to open database tpcc/tpcc: %d", rc);
        WARNING("%s\n", msg);
        err_printf("%s\n", msg);
        mon_TerminateServer(msg);
    }
    if (preallocate_cn || num_connections == 1) {
        int i;
        thread_info_t *curP;
        /* Preallocate all the desired connections */
        logprintf("Preallocating %d connections to the DB\n", num_connections);
        info_array = (thread_info_t *)calloc(num_connections, sizeof(*info_array));

        for (i=0, curP = info_array; i<num_connections; i++, curP++) {
            if (server_null_test)
                curP->cnP = create_null_connection();
            else
                curP->cnP = create_connection();
        }
        num_allocated = num_connections;
    }
}

```

```

/* initialize pServerInfo */
#ifdef WIN32
pServerInfo = perfSrvDataInit(tpcc_serverName, pa_num);
#endif
if (pServerInfo == NULL)
    pServerInfo = malloc(sizeof(server_info_t));
memset(pServerInfo, 0, sizeof(server_info_t));
#ifdef WIN32
start_bg_thread();
#endif

err_printf(">> Calling mon_BeginService()\n");

ENCINA_CALL("mon_BeginService", mon_BeginService());

fprintf(stderr, "mon_BeginService returned ... terminating\n");
TPCexit();
}

static void clean_thread_data(void *ptr) {
thread_info_t *threadP = (thread_info_t *)ptr;
if (server_null_test)
    clean_null_connection(threadP->cnP);
else
    clean_connection(threadP->cnP);
err_printf("Closing connection 0x%p. Called %d, %d errors\n",
    threadP->cnP, threadP->calls, threadP->errors);
}

/*
 * The routine executed by the deferred delivery thread
 */
/* Logic:
 * Wait until there is a valid request in the deferred delivery data.
 * After processing the request data_valid is set to FALSE
 * (allowing new requests to be queued).
 * This is a simple fixed size queue implemented in a cyclic array
 */
static void deferred_delivery()
{
thread_info_t *infoP;
pthread_mutex_lock(&deferred_dvry_data.lock);

while (1) {
if (deferred_dvry_data.size > 0) {
/*
 * There is a request to be processed
 */
#ifdef WIN32
int ind = deferred_dvry_data.head % MAX_DVRY_QUEUE;
delivery_sql_t dbData;
dbData.data = deferred_dvry_data.data[ind];
dbData.queue_time = deferred_dvry_data.queue_time[ind];
#else
delivery_data_t data = deferred_dvry_data.data[ind];
#endif

deferred_dvry_data.head++;
deferred_dvry_data.size--;

if (deferred_dvry_data.num_waiters > 0)
pthread_cond_signal(&deferred_dvry_data.q_cond);

if (deferred_dvry_data.head % 1000 == 0) {
err_printf("Processed %d deferred deliveries so far, queue size %d\n",
    deferred_dvry_data.head,
    deferred_dvry_data.size);
}

if (deferred_dvry_data.head > deferred_dvry_data.tail) {
err_printf("Error: Deferred Queue: head %d > tail %d\n",
    deferred_dvry_data.head,
    deferred_dvry_data.tail);

continue;
}

pthread_mutex_unlock(&deferred_dvry_data.lock);
#ifdef WIN32
if (server_null_test) {
sim_delivery(&dbData);
} else {
infoP = get_thread_data();
do_delivery(infoP->cnP, &dbData);
}
#else
if (server_null_test) {
sim_delivery(&data);
} else {
infoP = get_thread_data();
do_delivery(infoP->cnP, &data);
}
#endif

DPRINTF(("Deferred: Locking\n"));
pthread_mutex_lock(&deferred_dvry_data.lock);
} else {
/*
 * Wait for a request to be queued
 */
DPRINTF(("Deferred delivery waiting\n"));
pthread_cond_wait(&deferred_dvry_data.work_cond,
    &deferred_dvry_data.lock);

DPRINTF(("Deferred: Awake\n"));
}
}

/*
 * queue_delivery
 *
 * Queue a delivery request to be processed in the background
 * The queue is implemented as a simple queue of size 1.
 * If data_valid is true: there is already a request waiting in the queue
 * Sleep on a condition variable until the queue is empty.
 * Once the queue is empty put the request in the queue, wake up the
 * background thread and leave.
 */

static void queue_delivery(dataP
delivery_data_t *dataP;
{
struct timeval now;
int waited = 0;
static int last_report_time = 0;
#ifdef WIN32
SYSTEMTIME queue_time;
#endif

DPRINTF(("queue: Locking\n"));
pthread_mutex_lock(&deferred_dvry_data.lock);
DPRINTF(("queue: Locked\n"));

while (deferred_dvry_data.size >= MAX_DVRY_QUEUE) {
/* The request queue is full
 * Wait until a request is processed and removed from the queue.
 */
deferred_dvry_data.num_waiters++;
DPRINTF((">> queue_delivery: %d waiters, size %d\n",
    deferred_dvry_data.num_waiters, deferred_dvry_data.size));
DPRINTF(("Queue Delivery waiting, %d waiters\n",
    deferred_dvry_data.num_waiters));
pthread_cond_wait(&deferred_dvry_data.q_cond,
    &deferred_dvry_data.lock);
deferred_dvry_data.num_waiters--;
waited++;
}
DPRINTF(("Queueing delivery\n"));
/*
 * There is room in the queue.
 * Enter the request and wake up the background thread
 */
#ifdef WIN32
GetLocalTime(&queue_time);
deferred_dvry_data.size++;
deferred_dvry_data.data[deferred_dvry_data.tail % MAX_DVRY_QUEUE] = *dataP;
deferred_dvry_data.queue_time[deferred_dvry_data.tail % MAX_DVRY_QUEUE] = queue_time;
#else
get_local_time(&now);
dataP->start_queue = (double)now.tv_sec + (now.tv_usec / 1000000.0);
deferred_dvry_data.size++;
deferred_dvry_data.data[deferred_dvry_data.tail % MAX_DVRY_QUEUE] = *dataP;
if (now.tv_sec - last_report_time > 29) {
err_printf("queue_delivery - %d waiters, size %d\n",
    deferred_dvry_data.num_waiters, deferred_dvry_data.size);
last_report_time = now.tv_sec;
}
}
#endif

deferred_dvry_data.tail++;
DPRINTF(("queue_delivery: Signalling\n"));
pthread_cond_signal(&deferred_dvry_data.work_cond);

DPRINTF(("queue_delivery: Unlocking. Tail %d, size %d Max %d\n",
    deferred_dvry_data.tail,
    deferred_dvry_data.size,
    MAX_DVRY_QUEUE));
pthread_mutex_unlock(&deferred_dvry_data.lock);

if (waited) err_printf(">> queue_delivery waited %d times\n", waited);
dataP->header.returncode = TPCC_SUCCESS;
}

/*
 * start_deferred_delivery_threads
 *
 * Initialize the deferred delivery data structure and start
 * a background thread to process the delivery requests
 */
static void start_deferred_delivery_threads()
{
pthread_t thread;
int i;
int rc;

pthread_mutex_init(&deferred_dvry_data.lock, pthread_mutexattr_default);
pthread_cond_init(&deferred_dvry_data.work_cond, pthread_condattr_default);
pthread_cond_init(&deferred_dvry_data.q_cond, pthread_condattr_default);

```

```

deferred_dvry_data.num_waiters = 0;
deferred_dvry_data.head = 0;
deferred_dvry_data.tail = 0;
deferred_dvry_data.size = 0;
deferred_dvry_data.allocated = dvry_queue_size;
deferred_dvry_data.data =
    (delivery_data_t *)malloc(dvry_queue_size * sizeof(delivery_data_t));
#ifdef WIN32
deferred_dvry_data.queue_time =
    (SYSTEMTIME *)malloc(dvry_queue_size * sizeof(SYSTEMTIME));
#endif

/*
 * Create the background delivery thread.
 */
err_printf("Starting %d deferred delivery threads, queue size %d\n",
    num_deferred_dvry_threads,
    dvry_queue_size);
for (i=0; i<num_deferred_dvry_threads; i++) {
    if ((rc = pthread_create(&thread,
        pthread_attr_default,
        (pthread_startoutine_t)deferred_delivery,
        (pthread_addr_t)0) != 0) {
        WARNING("Failed to create delivery thread rc=%d\n", rc);
        exit(1);
    }
    (void)pthread_detach(&thread);
}

void exit_program(code)
    int code;
{
    char errMsg[55];
    sprintf(errMsg, "exit_program called with code %d", code);
    fprintf(stderr, "%s\n", errMsg);

    TPCexit();

    mon_TerminateServer(errMsg);
}

static char *thread_state_to_str(int state)
{
    char *retval;
    switch(state) {
        case SVR_STATE_NONE: retval = "None"; break;
        case SVR_STATE_SENT: retval = "Sent"; break;
        case SVR_STATE_REPLIED: retval = "Replied"; break;
        case SVR_STATE_ERR: retval = "Err"; break;
        default: retval = "unknown"; break;
    }
    return retval;
}

static thread_info_t *get_thread_data() {
    thread_info_t *dataP;
    struct timeval cur_time;

    /* Get a thread structure.
     * Each thread always uses the same connection.
     * The first time the thread tries to talk to the DB it creates
     * a connection, initializes it and stores it in a thread global
     * data structure.
     *
     * There is a special case for the single connection case: If there
     * is exactly one connection then it is global and not per thread.
     * There may be many threads but it is assumed that the application is
     * responsible for synchronizing the threads so that no two threads
     * ever use the connection at the same time.
     */
    if (num_connections == 1) {
        dataP = &info_array[0];
    } else {
        pthread_getspecific(thread_key, (pthread_addr_t *)&dataP);
    }
    if (dataP == NULL) { /* No connection assigned to this thread */
        pthread_mutex_lock(&init_lock); /* Initialize a connection */
        get_local_time(&cur_time);

        fprintf(stderr, "get_cn> initializing thread slot\n");

        if (preallocate_cn) {
            if (next_thread >= num_allocated) {
                fprintf(stderr, "Too many threads, not enough connections\n");
                mon_TerminateServer("Too many threads, not enough connections");
            }
            dataP = &info_array[next_thread++];
        } else {
            dataP = (thread_info_t *)malloc(sizeof(thread_info_t));
            memset(dataP, (char)0, sizeof(*dataP));
            if (server_null_test)
                dataP->cnP = create_null_connection();
            else
                dataP->cnP = create_connection();
        }
        pthread_setspecific(thread_key, dataP); /* Store it */
    }
}

```

```

fprintf(stderr, "get_cn> initialized connection 0x%x\n", dataP);
pthread_mutex_unlock(&init_lock);
}
return dataP;
}

static void pre_DB(char *name, data_header *headerP,
    int tran_type, thread_info_t *infoP)
{
    struct timeval tp;
    DPRINT("> %s", name);
    get_local_time(&tp);
    if (infoP != NULL) {
        infoP->cur_tran_type = tran_type;
        infoP->calls++;
        infoP->state = SVR_STATE_SENT;
        infoP->tran_time = tp;
    }

    headerP->start_time.sec = tp.tv_sec;
    headerP->start_time.usec = tp.tv_usec;
}

static void post_DB(char *name, data_header *headerP,
    int tran_type, thread_info_t *infoP)
{
    struct timeval tp;
    DPRINT("< %s\n", name);
    get_local_time(&tp);
    headerP->end_time.sec = tp.tv_sec;
    headerP->end_time.usec = tp.tv_usec;
    headerP->dtype = serverIdNumber;

    if (infoP != NULL) {
        infoP->tran_time = tp;
        infoP->state = SVR_STATE_REPLIED;
    }

    pServerInfo->tran[tran_type].num++;
    /* store the RT info for this server */
    if (tran_type <= MAX_TRAN_TYPE && tran_type > 0) {
        pServerInfo->tran[tran_type].RTtotal +=
            time_diff_ms(&(headerP->end_time), &(headerP->start_time));
        pServerInfo->tran[tran_type].RTcount++;
    }
}

/*
 * ----- The following are the entry points
 * for the RPCs arriving at the Server
 */

void impTPCCDbInfo(dataP, trpcStatus)
    dbInfo_data_t *dataP;
    trpc_status_t *trpcStatus;
{
    UNCOND_EVENT("> impTPCCDbInfo");
    err_printf("> impTPCCDbInfo");
    dataP->server_id = serverIdNumber;
    err_printf("< impTPCCDbInfo");
}

void impTPCCNOInfo(dataP, trpcStatus)
    dbInfo_data_t *dataP;
    trpc_status_t *trpcStatus;
{
    impTPCCDbInfo(dataP, trpcStatus);
}

void impTPCCNewOrder(dataP, trpcStatus)
    newOrder_data_t *dataP;
    trpc_status_t *trpcStatus;
{
    static int numCalls = 0;
    thread_info_t *infoP = get_thread_data();
    FUNCTION_BEGIN("NewOrder", dataP, NEWO_TRANS, infoP);
    if (server_null_test) {
        sim_new_order(dataP);
    } else {
        do_new_order(infoP->cnP, dataP);
    }

    if ((dataP->header.returncode != TPCC_SUCCESS) &&
        (dataP->header.returncode != INVALID_NEWO)) {
        logprintf("< impTPCCNewOrder; rc=%d, sql=%d, isam=%d\n",
            dataP->header.returncode,
            dataP->header.sql_code,
            dataP->header.isam_code);
    } else if (dataP->header.returncode == INVALID_NEWO) {
        DPRINT("< impTPCCNewOrder INVALID_NEWO\n");
    }
    if (++numCalls % 10000 == 0) {
}

```



```

err_printf("impTPCCNewOrder so far %d\n", numCalls);
}
FUNCTION_END("NewOrder", dataP, NEWO_TRANS, infoP);
}

void impTPCCPayment(dataP, trpcStatus)
payment_data_t *dataP;
trpc_status_t *trpcStatus;
{
static int numCalls = 0;
thread_info_t *infoP = get_thread_data();
FUNCTION_BEGIN("Payment", dataP, PAYMENT_TRANS, infoP);
if (server_null_test) {
sim_payment(dataP);
} else {
do_payment(infoP->cnP, dataP);
}

if (dataP->header.returncode != TPCC_SUCCESS) {
logprintf("< impTPCCPayment; rc=%d, sql=%d, isam=%d\n",
dataP->header.returncode,
dataP->header.sql_code,
dataP->header.isam_code);
}

if (++numCalls % 10000 == 0) {
err_printf("impTPCCPayment so far %d\n", numCalls);
}
FUNCTION_END("Payment", dataP, PAYMENT_TRANS, infoP);
}

void impTPCCOrderStatus(dataP, trpcStatus)
orderStatus_data_t *dataP;
trpc_status_t *trpcStatus;
{
thread_info_t *infoP = get_thread_data();
FUNCTION_BEGIN("OrderStatus", dataP, ORDER_STAT_TRANS, infoP);
if (server_null_test) {
sim_order_status(dataP);
} else {
do_order_status(infoP->cnP, dataP);
}

if (dataP->header.returncode != TPCC_SUCCESS) {
logprintf("< impTPCCOrderStatus; rc=%d, sql=%d, isam=%d\n",
dataP->header.returncode,
dataP->header.sql_code,
dataP->header.isam_code);
}
FUNCTION_END("OrderStatus", dataP, ORDER_STAT_TRANS, infoP);
}

void impTPCCStockLevel(dataP, trpcStatus)
stockLevel_data_t *dataP;
trpc_status_t *trpcStatus;
{
thread_info_t *infoP = get_thread_data();
FUNCTION_BEGIN("StockLevel", dataP, STOCK_TRANS, infoP);
if (server_null_test) {
sim_stock_level(dataP);
} else {
do_stock_level(infoP->cnP, dataP);
}

if (dataP->header.returncode != TPCC_SUCCESS) {
logprintf("< impTPCCStockLevel; rc=%d, sql=%d, isam=%d\n",
dataP->header.returncode,
dataP->header.sql_code,
dataP->header.isam_code);
}
FUNCTION_END("StockLevel", dataP, STOCK_TRANS, infoP);
}

void impTPCCDelivery(dataP, trpcStatus)
delivery_data_t *dataP;
trpc_status_t *trpcStatus;
{
#ifdef WIN32
delivery_sql_t dbData;
#endif

thread_info_t *infoP = NULL;
FUNCTION_BEGIN("DELIVERY", dataP, DELIVERY_TRANS, infoP);
if (num_deferred_dvry_threads > 0) {
queue_delivery(dataP);
} else {
#ifdef WIN32
if (server_null_test) {
sim_delivery(&dbData);
} else {
infoP = get_thread_data();
do_delivery(infoP->cnP, &dbData);
}
#else
if (server_null_test) {
sim_delivery(dataP);
} else {
infoP = get_thread_data();
do_delivery(infoP->cnP, dataP);
}
#endif
}
}

```

```

}
#endif
}

if (dataP->header.returncode != TPCC_SUCCESS) {
logprintf("< impTPCCDelivery; rc=%d, sql=%d, isam=%d\n",
dataP->header.returncode,
dataP->header.sql_code,
dataP->header.isam_code);
}
FUNCTION_END("DELIVERY", dataP, DELIVERY_TRANS, infoP);
}

/* functions in order to run with NULL database */
static void *create_null_connection() {
static cn_num = 0;
int *id = (int *)malloc(sizeof(int));
*id = cn_num++;
return id;
}

static void clean_null_connection(void *ptr) {
free(ptr);
return;
}

```

## stocklevel.tacf

```

/*
 * Copyright (C) 1991, 1990 Transarc Corporation
 * All Rights Reserved
 */
/*
 * stocklevel.tacf -- attribute configuration file for tpcc server.
 * used for transparent binding
 *
 * $Revision: 1.1 $
 * $Date: 1998/11/06 21:10:16 $
 * $Log: tpcc.tacf,v $
 *
 * $TALog: stocklevel.tacf,v $
 * Revision 1.1 1998/11/06 21:10:16 dongfeng
 * - Move all files common to client and server to tpcc/common
 * directory
 * [added by delta dongfeng-23677-TPCC-new-directory-structures, r1.1]
 *
 * Revision 1.2 1998/10/08 18:03:04 gerstl
 * Changes to allow configurations where some servers only service
 * specific transaction types. Split transaction interfaces by type.
 * [added by delta gerstl-23515-TPCC-allow-separate-online-transaction-interfaces, r1.1]
 *
 *
 *
 */

```

```

[implicit_handle (mon_handle_t handle)]
interface stocklevel
{
}

```

## stocklevel.tidl

```

/*
 * id: $id: $
 *
 * component_name: encina benchmarks
 *
 * the following functions list may not be complete.
 * functions defined by/via macros may not be included.
 *
 * functions:
 * <fill_me_in>
 *
 * origins: transarc corp.
 *
 * (c) copyright transarc corp. 1995, 1993
 * all rights reserved
 * licensed materials - property of transarc
 *
 * us government users restricted rights - use, duplication or
 * disclosure restricted by gsa adp schedule contract with transarc corp
 */
/*
 * history
 * $Talog: $
 */
/*
 * stocklevel.tidl -- interface definition file for tpccserver.
 *
 * $Revision: 1.0 $
 * $Date: 1995/10/20 21:55:05 $
 * $Log: tpcc.tidl,v $
 */
[
uid(1dda58e8-5e05-11d2-bd18-9e621208aa77),

```

```

version(1.0)
]
interface stocklevel
{
import "tpm/mon/mon_handle.idl";
import "tpcc_type.idl";

[nontransactional] void
    impTPCCStockLevel([in,out] stockLevel_data_t *dataP,
                      [out] trpc_status_t *trpcStatus);
}

                tpcc.h

#ifndef TPCC_H_INCLUDED
#define TPCC_H_INCLUDED
/*
 * File: tpcc.h
 * created: 8-26-91
 * program description:
 * This module contains global variables and data definitions
 * for the tpcc application.
 */
#include "../common/tpcc_type.h"

#define TPCC_H

/*
 * Global numbers, constants,...
 */

#define INVALID_ITEM 100
#define TRAN_OK 0
#define REMOTE_WAREHOUSE 17

#define FORM_DATE 1
#define FORM_DATETIME 2

#define MAX_ITEMS 15

/*
 * transaction structures
 */

typedef orderStatus_data_t OrderStatus_data;
typedef newOrder_data_t NewOrder_data;
typedef stockLevel_data_t StockLevel_data;
typedef delivery_data_t Delivery_data;
typedef payment_data_t Payment_data;

Compatibility for older .sqc files
*****
#define s_C_BALANCE c_balance
#define s_C_CITY c_city
#define s_C_CREDIT c_credit
#define s_C_CREDIT_LIM c_credit_lim
#define s_C_DATA c_data
#define s_C_DISCOUNT c_discount
#define s_C_D_ID c_d_id
#define s_C_FIRST c_first
#define s_C_ID c_id
#define s_C_LAST c_last
#define s_C_MIDDLE c_middle
#define s_C_PHONE c_phone
#define s_C_SINCE c_date
#define s_C_STATE c_state
#define s_C_STREET_1 c_street_1
#define s_C_STREET_2 c_street_2
#define s_C_W_ID c_w_id
#define s_C_ZIP c_zip
#define s_D_CITY d_city
#define s_D_ID d_id
#define s_D_STATE d_state
#define s_D_STREET_1 d_street_1
#define s_D_STREET_2 d_street_2
#define s_D_TAX d_tax
#define s_D_ZIP d_zip
#define s_H_AMOUNT h_amount
#define s_H_DATE pay_date
#define s_I_NAME name_i
#define s_I_PRICE price
#define s_OL_AMOUNT ol_amount
#define s_OL_DELIVERY_D delivery_date
#define s_OL_I_ID ol_i_id
#define s_OL_QUANTITY ol_quantity
#define s_OL_SUPPLY_W_ID ol_supply_w_id
#define s_O_CARRIER_ID o_carrier_id
#define s_O_ENTRY_D entry_date
#define s_O_ID o_id
#define s_O_OL_CNT o_ol_cnt

```

```

#define s_S_QUANTITY s_quantity
#define s_QUANTITY quantity
#define s_W_CITY w_city
#define s_W_ID w_id
#define s_W_STATE w_state
#define s_W_STREET_1 w_street_1
#define s_W_STREET_2 w_street_2
#define s_W_TAX w_tax
#define s_W_ZIP w_zip
#define s_all_local o_all_local
#define s_brand_generic brand_generic
#define s_exec_status exec_status
#define s_low_stock stock_count
#define s_ol_cnt o_ol_cnt
#define s_queued_time queued_time
#define s_status_line statusline
#define s_threshold threshold
#define s_total_amount total
#define s_transtatus header.returncode

#ifndef TPCC_H_INCLUDED
#define TPCC_H_INCLUDED
/*
 * Copyright (C) 1991, 1990 Transarc Corporation
 * All Rights Reserved
 */
/*
 * tpcc.tacf -- attribute configuration file for tpcc server.
 * used for transparent binding
 */
/* $Revision: 1.1 $
 * $Date: 1998/11/06 21:10:17 $
 * $Log: tpcc.tacf,v $
Revision 4.2 95/05/16 10:55:49 10:55:49 tpcc (TPCC Benchmark)
Added necessary RCS ident strings
 */

[implicit_handle (mon_handle_t handle)]
interface tpccTransactions
{
}

                tpcc.tacf

/*
 * Copyright (C) 1991, 1990 Transarc Corporation
 * All Rights Reserved
 */
/*
 * neworder.tacf -- attribute configuration file for tpcc server.
 * used for transparent binding
 */
/* $Revision: 1.1 $
 * $Date: 1998/11/06 21:10:17 $
 * $Log: tpcc.tacf,v $
 * $TALog: tpcc_trans.tacf,v $
 * Revision 1.1 1998/11/06 21:10:17 dongfeng
 * - Move all files common to client and server to tpcc/common
 * directory
 * [added by delta dongfeng-23677-TPCC-new-directory-structures, r1.1]
 */
/* Revision 1.1 1997/06/16 22:04:48 oz
 * - Integration with Data Dependent Routing: Phase 1
 * - Separated the all the binding related code into its own files.
 * - Added mon_client_utils.[ch] that export binding related calls.
 * - Added a TPCC_USE_DDR compile time switch
 * - Added tpcc_trans.tidl: All the functions in one interface.
 * [added by delta oz-20170-TPCC-add-data-dependent-routing, r1.1]
 */
/*
 */
[implicit_handle (mon_handle_t handle)]
interface tpccTrans
{
}

```

## tpcc\_trans.tidl

```
*
* id: Sid: $
*
* component_name: encina benchmarks
*
* the following functions list may not be complete.
* functions defined by/via macros may not be included.
*
* functions:
* <fill_me_in>
*
* origins: transarc corp.
*
* (c) copyright transarc corp. 1995, 1993
* all rights reserved
* licensed materials - property of transarc
*
* us government users restricted rights - use, duplication or
* disclosure restricted by gsa adp schedule contract with transarc corp
*/
*
* history
* Stalog: $
*/
*
* tpcc_trans.tidl -- interface definition file for tpccserver.
*
* $Revision: 1.11 $
* $date: 1995/10/20 21:55:05 $
* $Log: tpcc.tidl,v $
*/
[uuid(955d7288-e672-11d0-bcef-9e621234aa77), version(1.0)]
interface tpccTrans
{
import "tpm/mon/mon_handle.idl";
import "tpcc_type.idl";

[nontransactional] void
    impTPCCNewOrder([in,out] newOrder_data_t *dataP,
                    [out] trpc_status_t * trpcStatus);

[nontransactional] void
    impTPCCPayment([in,out] payment_data_t *dataP,
                  [out] trpc_status_t * trpcStatus);

[nontransactional] void
    impTPCCOrderStatus([in,out] orderStatus_data_t *dataP,
                       [out] trpc_status_t * trpcStatus);

[nontransactional] void
    impTPCCStockLevel([in,out] stockLevel_data_t *dataP,
                      [out] trpc_status_t * trpcStatus);

[nontransactional] void
    impTPCCNOInfo([out] dbInfo_data_t *dataP,
                 [out] trpc_status_t * trpcStatus);
}
}
```

## tpcc\_type.idl

```
*
* tpcc_type.idl
*
* $Revision: 1.2 $
* $Date: 1998/12/08 18:55:21 $
* $Log: $
*
* $TALog: tpcc_type.idl,v $
* Revision 1.2 1998/12/08 18:55:21 wenjian
* Add "int stats" to data_header structure
* [from r1.1 by delta wenjian-23785-TPCC-pass-statsFrequency-from-client-to-server, r1.1]
*
* Revision 1.1 1998/11/06 21:10:17 dongfeng
* - Move all files common to client and server to tpcc/common
* directory
* [added by delta dongfeng-23677-TPCC-new-directory-structures, r1.1]
*
* Revision 1.11 1998/01/24 14:17:07 oz
* - User server name to identify server and name delivery file
* - Use env variable HOME instead of /home/encina if HOME is set
*
* - Added const ONLINE_INTERFACES
* [from r1.10 by delta oz-21687-TPCC-use-server-name-to-identify-process, r1.1]
*
* Revision 1.10 1998/01/23 15:09:11 oz
* - Updated the SP TPCC directory to the latest files used
* during the SP tpcc audit.
* [from r1.9 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
```

```
*
*
*
*/
[
    uuid(008c6338-2b0a-1001-a9ab-02608c2f015a), version(1)
]
interface tpcc_types {

const long NAME_LENGTH = 32;

const long NEWO_INTERFACE = 0x01;
const long PAYMENT_INTERFACE = 0x02;
const long ORDER_STAT_INTERFACE = 0x04;
const long DELIVERY_INTERFACE = 0x08;
const long STOCK_INTERFACE = 0x10;
const long ONLINE_INTERFACES = NEWO_INTERFACE | PAYMENT_INTERFACE |
ORDER_STAT_INTERFACE | STOCK_INTERFACE;
const long ALL_INTERFACE = 0xffff;

const long NEWO_TRANS = 1;
const long PAYMENT_TRANS = 2;
const long ORDER_STAT_TRANS = 3;
const long DELIVERY_TRANS = 4;
const long STOCK_TRANS = 5;
const long MAX_TRAN_TYPE = 5;

typedef struct {
    long int sec;
    long int usec;
} time_type;

typedef struct {
    short int dtype;
    short int returncode;
    long int sql_code;
    long int isam_code;
    long int num_rms;

    short int stats; /* For instrument only */
    time_type start_time; /* For Debug Purposes only */
    time_type end_time; /* For Debug Purposes only */
} data_header;

/* Definitions for payment transaction
*
* payment_data_t
*
* An in-out structure for payment transaction.
* It contains all the input parameters as well as the output parameters.
*/
typedef struct {
    data_header header;
    short int w_id;
    short int d_id;
    short int c_id;
    short int c_w_id;
    short int c_d_id;
    short int byname;
    double h_amount;
    char pay_date[20];

    char w_name[11];
    char w_street_1[21];
    char w_street_2[21];
    char w_city[21];
    char w_state[3];
    char w_zip[10];

    char d_name[11];
    char d_street_1[21];
    char d_street_2[21];
    char d_city[21];
    char d_state[3];
    char d_zip[10];

    char c_first[17]; /* was C_LAST_LEN already includes +1 */
    char c_middle[3];
    char c_last[17];
    char c_phone[17];
    char c_credit[3];
    char c_street_1[21];
    char c_street_2[21];
    char c_city[21];
    char c_state[3];
    char c_zip[10];
    double c_credit_lim;
    double c_balance;
    double c_discount;
    double c_ytd_payment;
    short int c_payment_cnt;
    char c_date[20];
    char c_data[201];
} payment_data_t;

/* Definitions for new order transaction */
typedef struct {
    short int ol_supply_w_id;
```

<pre> short int ol_quantity; short int s_quantity; long int ol_i_id; char name_i[25]; char brand_generic[2]; double price; double ol_amount; long int s_idx; char s_dist[25]; } OL_TABLE, newOrder_item_t;  typedef struct {     data_header header;     short int w_id;     short int d_id;     short int c_id;     short int o_ol_cnt;     short int o_all_local;     short int items_valid; /* true if all valid */     short int total_items;     long int o_id;     double w_tax;     double d_tax;     double total;     double c_discount;     char entry_date[20];     char c_last[17];     char c_credit[3];     char statusline[26];     OL_TABLE item[15]; } newOrder_data_t;  /* Definitions for order status transaction */  typedef struct {     long int ol_i_id;     short int ol_supply_w_id;     short int ol_quantity;     double ol_amount;     char delivery_date[20]; } orderStatusItem_t;  typedef struct {     data_header header;     short int w_id;     short int d_id;     short int c_id;     short int o_id;     short int o_ol_cnt;     short int byname;     short int o_carrier_id;     char c_last[17];     char c_first[17];     char c_middle[3];     char entry_date[20];     double c_balance;     orderStatusItem_t item[15]; } orderStatus_data_t;  /* Definitions for stock level transaction */  typedef struct {     data_header header;     short int w_id;     short int d_id;     short int threshold;     long int stock_count; } stockLevel_data_t;  /* Definitions for delivery transaction */  typedef struct {     data_header header;     short int w_id;     short int o_carrier_id;     long int queued_time;     short status;     char exec_status[50];     double start_queue; } delivery_data_t;  typedef struct {     long int first_wh;     long int last_wh;     long int server_id; } dbInfo_data_t;  /*  * A union of all the transactions  */ typedef union switch(long int tran_type) data {     case NEWO_TRANS: newOrder_data_t new_order;     case PAYMENT_TRANS: payment_data_t payment;     case ORDER_STAT_TRANS: orderStatus_data_t order_status;     case DELIVERY_TRANS: delivery_data_t delivery;     case STOCK_TRANS: stockLevel_data_t stock_level; } tpc_data_t; </pre>	<pre> }  <b><u>tpcc_utils.c</u></b>  /*  *  * tpc_utils.c  *  * \$Revision: 1.2 \$  * \$Date: 1998/12/14 20:27:57 \$  * \$Log: \$  *  *  *  * \$TALog: tpcc_utils.c,v \$  * Revision 1.2 1998/12/14 20:27:57 wenjian  * Made corresponding changes due to data structure change of tran_info_t.  *  * - Add header file winsock.h for NT platform  * [from r1.1 by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.3]  *  * Revision 1.1 1998/12/11 16:37:58 wenjian  * Move some common functions from client/client_utils.c to common/tpcc_utils.c.  * In this version, we only move time_diff_ms(). Need some work in order to  * move other functions like ERROUT.  *  * - A file including utility functions for both client and server  * [added by delta wenjian-23788-TPCC-use-single-stats-var-for-each-client-and-server, r1.2]  *  *  *  * tpc_utils.c  * Generic utilities used by the client and server processes  */  #include &lt;stdio.h&gt; #include &lt;time.h&gt; #include &lt;string.h&gt; #include &lt;stdarg.h&gt;  #ifdef (solaris) #include &lt;dce/pthread.h&gt; #else /* solaris */ #include &lt;pthread.h&gt; #endif  #include "databuf.h" #include "do_tpcc.h" #include "tpcc_type.h"  #ifdef WIN32 #include &lt;winsock.h&gt; #endif  /*  * time_diff_ms  * Return the difference in milliseconds between two times  */ int time_diff_ms(t2, t1) struct timeval *t2, *t1; {     int t_diff;      t_diff = (t2-&gt;tv_usec + 1000000 - t1-&gt;tv_usec + 500) / 1000 +             (t2-&gt;tv_sec - t1-&gt;tv_sec - 1) * 1000;      return(t_diff); }  <b><u>util.h</u></b>  #ifdef LOCAL_UTIL_H #define LOCAL_UTIL_H  #include "util_token.h"  #define UTIL_ALLOC(ptr, type, size) \     ptr = (type)malloc(size); \     if (ptr==NULL) { \         fprintf(stderr, "UTIL_ALLOC failed\n"); \         exit(1); \     } #endif  <b><u>util_alloc.h</u></b>  /*  * util_alloc.h  *  * \$Revision: 1.1 \$  * \$Date: 1998/11/06 21:10:18 \$  * \$Log: util_alloc.h,v \$ </pre>
--	--

```

* Revision 4.2 95/05/16 10:55:43 10:55:43 tpec (TPCC Benchmark)
* Added necessary RCS ident strings
*
*
*/

#ifndef TRANSARC_UTIL_ALLOC_H
#define TRANSARC_UTIL_ALLOC_H

*
* UTIL_[ALLOC, REALLOC, NEW, FREE] -- macros that wrap calls to
* malloc, realloc, free. The allocation macros check the return
* value, a NULL pointer is converted into a fatal error.
*/
#define UTIL_ALLOC_ROBUST(ptr, type, size) \
((ptr) = (type) malloc(size))

#define UTIL_ALLOC(ptr, type, size) \
do { \
    if (UTIL_ALLOC_ROBUST(ptr, type, size) == 0) \
        util_MemoryError("UTIL_ALLOC", __FILE__, __LINE__); \
} while (0)

#define UTIL_REALLOC_ROBUST(ptr, type, size) \
(ptr = (type) realloc((void *) ptr, size))

#define UTIL_REALLOC(ptr, type, size) \
do { \
    if (UTIL_REALLOC_ROBUST(ptr, type, size) == 0) \
        util_MemoryError("UTIL_REALLOC", __FILE__, __LINE__); \
} while (0)

#define UTIL_FREE(ptr) \
do { \
    if (!ptr) { \
        util_MemoryError("UTIL_FREE", __FILE__, __LINE__); \
    } \
    free((void *) (ptr)); \
    ptr = 0; /* Make all free'd pointers zero. */ \
} while (0)

#define UTIL_ALLOC_ARRAY_ROBUST(ptr, type, number) \
((ptr) = (type *) malloc(sizeof(type) * (number)))

#define UTIL_ALLOC_ARRAY(ptr, type, number) \
do { \
    if (UTIL_ALLOC_ARRAY_ROBUST(ptr, type, number) == 0) \
        util_MemoryError("UTIL_ALLOC_ARRAY", __FILE__, __LINE__); \
} while (0)

#define UTIL_COPY_STRING_ROBUST(to, from) \
(((to) = (char *) malloc(strlen((char *) (from)) + 1)) ? \
strcpy((char *) (to), (char *) (from)) : 0)

#define UTIL_COPY_STRING(to, from) \
do { \
    if (UTIL_COPY_STRING_ROBUST(to, from) == 0) \
        util_MemoryError("UTIL_COPY_STRING", __FILE__, __LINE__); \
} while (0)

#endif /* TRANSARC_UTIL_ALLOC_H */


```

### util\_token.h

```

*
* ID: $Id: util_token.h,v 1.1 1998/11/06 21:10:18 dongfeng Exp $
*
* COMPONENT_NAME: Encina Toolkit Executive
*
* The following functions list may not be complete.
* Functions defined by/via macros may not be included.
*
* FUNCTIONS:
*
* ORIGINS: Transarc Corp.
*
* (C) COPYRIGHT Transarc Corp. 1995, 1994, 1993, 1990
* All Rights Reserved
* Licensed Materials - Property of Transarc
*
* US Government Users Restricted Rights - Use, duplication or
* disclosure restricted by GSA ADP Schedule Contract with Transarc Corp
*/
*
* HISTORY
* $TALog: util_token.h,v $
* Revision 1.1 1998/11/06 21:10:18 dongfeng
* - Move all files common to client and server to tpcc/common
* directory

```

```

* [added by delta dongfeng-23677-TPCC-new-directory-structures, r1.1]
*
* Revision 1.7 1995/01/30 12:50:13 barry
* Update copyrights for Encina 1.2.
* [from r1.6 by delta barry-0000-update-copyrights-for-1.2, r1.1]
*
* Revision 1.6 1995/01/13 14:12:51 psu
* fix comment to conform to coding standards.
* [from r1.5 by delta psu-13196-client-interoperate-with-oracle-pro-c-2, r1.3]
*
* Revision 1.5 1995/01/12 20:34:22 psu
* put comment on ## fix in the code
*
* try to make sure people don't change the use of ##
* [from r1.4 by delta psu-13196-client-interoperate-with-oracle-pro-c-2, r1.2]
*
* Revision 1.4 1995/01/12 15:48:28 psu
* fix use of ## to make pro*c happy
*
* pro*c 2.0 can't deal with a ## b ## c, change to a ## b##c.
* [from r1.3 by delta psu-13196-client-interoperate-with-oracle-pro-c-2, r1.1]
*
* Revision 1.3 1994/02/04 17:22:22 pinaki
* Update copyright.
* [from r1.2 by delta pinaki-0000-update-copyright-for-1.1, r1.1]
*
* Revision 1.2 1993/12/18 22:06:57 mwyong
* [from r1.1 by delta mwyong-10043-util-always-offer-UTIL_IDENT, r1.1]
*
* Revision 1.1 1993/12/03 22:00:04 mwyong
* Split the various features into separate files, so that they can
* be included separately.
* [added by delta mwyong-9848-utils-split-util.h-into-separately-usable-parts, r1.1]
*
*/
#ifndef TRANSARC_UTIL_TOKEN_H
#define TRANSARC_UTIL_TOKEN_H

#include <encina/c_prologue.h>

/* UTIL_IDENT -- the identity function */
#define UTIL_IDENT(a) a

/*
* UTIL_[STRING, CONCAT, CONCAT3] -- macros for converting into, and
* concatenating together, strings.
*/

/* Note, the a ## b##c is needed to make some broken cpp's work correctly.
* This was originally put here for Oracle Pro*C 2.0, but other compilers
* may have similar problems.
*/

#if ENCINA_C_ANSI_STRING_TOKEN_SUPPORT
#define UTIL_STRING(a) # a
#define UTIL_CONCAT(a, b) a ## b
#define UTIL_CONCAT3(a,b,c) a ## b##c
#else /* ENCINA_C_ANSI_STRING_TOKEN_SUPPORT */
#define UTIL_STRING(a) "a"
#define UTIL_CONCAT(a, b) UTIL_IDENT(a)b
#define UTIL_CONCAT3(a,b,c) UTIL_CONCAT(a,b)c
#endif /* ENCINA_C_ANSI_STRING_TOKEN_SUPPORT */

#include <encina/c_epilogue.h>
#endif /* TRANSARC_UTIL_TOKEN_H */


```

### utilities.h

```

*
* ID: $Id: utilities.h,v 1.1 1998/11/06 21:10:19 dongfeng Exp $
*
* COMPONENT_NAME: Encina Toolkit Server Core
*
* ORIGINS: Transarc Corp.
*
* (C) COPYRIGHT Transarc Corp. 1995
* All Rights Reserved
* Licensed Materials - Property of Transarc
*
* US Government Users Restricted Rights - Use, duplication or
* disclosure restricted by GSA ADP Schedule Contract with Transarc Corp
* $Revision: 1.1 $
* $Log: utilities.h,v $
*
* $TALog: utilities.h,v $
* Revision 1.1 1998/11/06 21:10:19 dongfeng
* - Move all files common to client and server to tpcc/common
* directory
* [added by delta dongfeng-23677-TPCC-new-directory-structures, r1.1]
*
* Revision 1.7 1998/10/27 14:57:52 dongfeng
* Change enc_status to a data structure that has fields:
* - Status code
* - Line Number
* - File Name
* - Encina Error Code

```

```

* - Error Msg
* Remove statusMsgs in web_tpc.c
* [from r1.6 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.6]
*
* Revision 1.6 1998/10/22 19:18:37 dongfeng
* [from r1.5 by delta dongfeng-23067-TPCC-add-web-based-tpcc-client, r1.2]
*
* Revision 1.5 1998/01/23 15:09:16 oz
* - Updated the SP TPCC directory to the latest files used
* during the SP tpc audit.
* [from r1.4 by delta oz-20774-TPCC-update-to-latest-SP-version-11-27, r1.1]
*
*/
*
* utilities.h -- holds declarations, macros, and constants used by the
* telshop/merchandise client-server program.
*
* SDate: 1998/11/06 21:10:19 S
*/

#ifndef UTILITIES_H
#define UTILITIES_H

#include <dce/rpc.h>
#include <dce/dce_error.h>
#include <encina/encina.h>
#include <stdlib.h>

#include <utils/trace.h>
#include "util_alloc.h"

/* Boolean type, and its constants */

#define FALSE 0
#define TRUE 1

#if ENCINA_C_ANSI_STRING_TOKEN_SUPPORT
#define UTIL_STRING(a) #a
#define UTIL_CONCAT(a, b) a ## b
#define UTIL_CONCAT3(a,b,c) a ## b##c
#else /* ENCINA_C_ANSI_STRING_TOKEN_SUPPORT */
#define UTIL_STRING(a) "a"
#define UTIL_CONCAT(a, b) UTIL_IDENT(a)b
#define UTIL_CONCAT3(a,b,c) UTIL_CONCAT(a,b)c
#endif /* ENCINA_C_ANSI_STRING_TOKEN_SUPPORT */

/* ENCINA_CALL: Make fail-fast calls on the various services. */

/* Macro delimiters */

#define BEGIN_MACRO do {
#define END_MACRO } while (0)

/* FATAL -- Failure. Print error message and exit the program */

void exit_program();

#ifndef FATAL
#define FATAL(args)
BEGIN_MACRO
printf args;
exit(1);
END_MACRO
#endif

/* ENCINA_CALL: Make fail-fast calls on the various services. */

#define ENCINA_CALL_RC(proc_name,call,rc)
BEGIN_MACRO
char _errorMsg[ENCINA_MAX_STATUS_STRING_SIZE];
rc = (call);
if (rc) {
encina_StatusToString(rc, ENCINA_MAX_STATUS_STRING_SIZE,
_errorMsg);

err_printf( "%x \n", rc);
err_printf( "%s \n", _errorMsg);
err_printf( "%s \n", proc_name);
}
END_MACRO

#define ENCINA_CALL(proc_name,call)
BEGIN_MACRO
unsigned long _status;
ENCINA_CALL_RC(proc_name,call,_status);
if (_status) exit_program(_status);
END_MACRO

typedef enum {
action_exit,
action_continue
} error_action_t;

```

```

#define CHECK_DCE_STATUS(_status, _msg, _action)
{
int error_stat;
unsigned long _rc = (_status);
unsigned char error_string[dce_c_error_string_len];
if ((_status) != rpe_s_ok) {
dce_error_inq_text(_rc, error_string, &error_stat);
err_printf("%s failed, error: %s (%d)\n", _msg, error_string, _rc);
if ((_action) == action_exit)
exit(-1);
}
}

#define DCE_CALL(call, args)
{
call args;
CHECK_DCE_STATUS(status, UTIL_STRING(call), action_exit);
}

/* MALLOC_CHECK -- Make sure there is memory to be allocated;
* fail if there is not. */

#define MALLOC_CHECK(memP)
BEGIN_MACRO
if (!memP)
FATAL("Out of memory.\n");
END_MACRO

/* ASSERT -- internal checks that assure the program is running correctly.
* Use to check program correctness, not user input. */

#ifndef ASSERT
#define ASSERT(condition)
BEGIN_MACRO
if (!(condition))
FATAL("'%s (%d): Assertion failed.\n", __FILE__, __LINE__);
END_MACRO
#endif

#define RAND(lim1, lim2) ((int)(drand48()*((lim2)-(lim1)+1)) + (lim1))

#ifndef BAD_STATUS
#define BAD_STATUS(call, status)
BEGIN_MACRO
char _errorMsg[ENCINA_MAX_STATUS_STRING_SIZE];
encina_StatusToString(status, ENCINA_MAX_STATUS_STRING_SIZE,
_errorMsg);
logprintf("%s: %s (%d)\n", UTIL_STRING(call), _errorMsg, status);
exit(1);
END_MACRO
#endif

#ifndef boolean_t
#define boolean_t int
#endif

#ifndef EXPORT
#define EXPORT
#endif

#ifndef IMPORT
#define IMPORT extern
#endif

/* For web_tpc_client */
#define CHK_STATUS(st, val, _errMsg)
BEGIN_MACRO
if(st) {
enc_status.status=val;
strcpy(enc_status.file, __FILE__);
enc_status.line= __LINE__;
enc_status.encinaError = st;
if(_errMsg)strcpy(enc_status.errorMsg, _errMsg);
if(st!=1) return;
}
END_MACRO

#endif /* UTILITIES_H */

```

## A.2 Client Transaction Code

initpay.sql

```

CREATE OR REPLACE PACKAGE initpay
AS
TYPE rowidarray IS TABLE OF ROWID INDEX BY BINARY_INTEGER;
row_id          rowidarray;
cust_rowid     ROWID;
dist_name      VARCHAR2(11);
ware_name      VARCHAR2(11);
c_num          BINARY_INTEGER;
PROCEDURE pay_init;
END initpay;
/
CREATE OR REPLACE PACKAGE BODY initpay AS
PROCEDURE pay_init IS
BEGIN
NULL;
END pay_init;
END initpay;
/
exit

```

## 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 warehouse
SET w_ytd = w_ytd+h_amount
WHERE w_id = :w_id
RETURNING w_name,
w_street_1, w_street_2, w_city, w_state, w_zip
INTO initpay.ware_name,
:w_street_1, :w_street_2, :w_city, :w_state, :w_zip;

--Bulk fetch
SELECT rowid
BULK COLLECT INTO initpay.row_id
FROM customer
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;

--Store number of rows processed
initpay.c_num := sql%rowcount;
initpay.cust_rowid := initpay.row_id((initpay.c_num) / 2);

UPDATE customer
SET c_balance = c_balance - :h_amount,
c_ytd_payment = c_ytd_payment + :h_amount,
c_payment_cnt = c_payment_cnt+1
WHERE rowid = initpay.cust_rowid
RETURNING
c_id, c_first, c_middle, c_last, c_street_1, c_street_2,
c_city, c_state, c_zip, c_phone,
c_since, c_credit, c_credit_lim,
c_discount, c_balance
INTO :c_id, :c_first, :c_middle, :c_last,
:c_street_1, :c_street_2, :c_city, :c_state,
:c_zip, :c_phone, :c_since, :c_credit,
:c_credit_lim, :c_discount, :c_balance;

:c_data := '';
IF :c_credit = 'BC' THEN
UPDATE customer
SET c_data = substr((to_char(:c_id) || '' ||
to_char(:c_d_id) || '' ||
to_char(:c_w_id) || '' ||
to_char(:d_id) || '' ||
to_char(:w_id) || '' ||
to_char(:h_amount/100, '9999.99') || ''')
|| c_data, 1, 500)
WHERE rowid = initpay.cust_rowid
RETURNING substr(c_data,1, 200)
INTO :c_data;

END IF;

UPDATE district
SET d_ytd = d_ytd+h_amount
WHERE d_id = :d_id
AND d_w_id = :w_id
RETURNING d_name, d_street_1, d_street_2, d_city,
d_state, d_zip
INTO initpay.dist_name, :d_street_1, :d_street_2, :d_city,
:d_state, :d_zip;

INSERT INTO history (h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
h_amount, h_date, h_data)
VALUES (:c_id, :c_d_id, :c_w_id, :d_id, :w_id, :h_amount,
:cr_date, initpay.ware_name || ' ' || initpay.dist_name);

```

```

--Sanjay-No commit needed iff Commit on Success done
-- COMMIT;
EXIT;

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

END LOOP;
END;

```

## paynz.sql

```

DECLARE /* paynz */
-- cust_rowid ROWID;
-- dist_name VARCHAR2(11);
-- ware_name VARCHAR2(11);
not_serializable EXCEPTION;
PRAGMA EXCEPTION_INIT(not_serializable,-8177);
deadlock EXCEPTION;
PRAGMA EXCEPTION_INIT(deadlock,-60);
snapshot_too_old EXCEPTION;
PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
BEGIN
LOOP BEGIN
UPDATE warehouse
SET w_ytd = w_ytd + :h_amount
WHERE w_id = :w_id
RETURNING w_name, w_street_1, w_street_2, w_city, w_state, w_zip
INTO initpay.ware_name, :w_street_1, :w_street_2, :w_city,
:w_state, :w_zip;

UPDATE customer
SET c_balance = c_balance - :h_amount,
c_ytd_payment = c_ytd_payment + :h_amount,
c_payment_cnt = c_payment_cnt+1
WHERE c_id = :c_id AND c_d_id = :c_d_id AND
c_w_id = :c_w_id
RETURNING rowid, c_first, c_middle, c_last, c_street_1,
c_street_2, c_city, c_state, c_zip, c_phone,
c_since, c_credit, c_credit_lim,
c_discount, c_balance
INTO initpay.cust_rowid, :c_first, :c_middle, :c_last, :c_street_1,
:c_street_2, :c_city, :c_state, :c_zip, :c_phone,
:c_since, :c_credit, :c_credit_lim,
:c_discount, :c_balance;
IF SQL%NOTFOUND THEN
raise NO_DATA_FOUND;
END IF;

-- :c_data := '';

IF :c_credit = 'BC' THEN
UPDATE customer
SET c_data = substr((to_char(:c_id) || '' ||
to_char(:c_d_id) || '' ||
to_char(:c_w_id) || '' ||
to_char(:d_id) || '' ||
to_char(:w_id) || '' ||
to_char(:h_amount, '9999.99') || ''')
|| c_data, 1, 500)
WHERE rowid = initpay.cust_rowid
RETURNING substr(c_data,1, 200)
INTO :c_data;

END IF;

UPDATE district
SET d_ytd = d_ytd + :h_amount
WHERE d_id = :d_id
AND d_w_id = :w_id
RETURNING d_name, d_street_1, d_street_2, d_city, d_state, d_zip
INTO initpay.dist_name, :d_street_1, :d_street_2, :d_city, :d_state,
:d_zip;
IF SQL%NOTFOUND THEN
raise NO_DATA_FOUND;
END IF;

INSERT INTO history (h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
h_amount, h_date, h_data)
VALUES
(:c_id, :c_d_id, :c_w_id, :d_id, :w_id, :h_amount,
:cr_date, initpay.ware_name || ' ' || initpay.dist_name);
-- COMMIT;
-- :h_date := to_char(:cr_date, 'DD-MM-YYYY.HH24:MI:SS');
EXIT;

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

END LOOP;

```

```
END;
```

## pldel.c

```
#ifndef RCSID
static char *RCSID =
    "Header: /afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/server/ora8.1_mt/RCS/pldel.c,v
1.2 1999/04/15 12:16:51 oz Exp $ 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"
#include "plora.h"
#ifdef TUX
#include <userlog.h>
#endif

#include "tpccflags.h"

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

#ifdef PLSQDEL
#define SQLTXT "BEGIN delivery.deliver (:w_id, :carrier_id, :order_id, \
:retry); END;"
#else
#ifdef DMLRETDEL
#define SQLTXT1 "DELETE FROM new_order WHERE no_d_id = :d_id \
AND no_w_id = :w_id AND rownum <= 1 \
RETURNING no_o_id into :o_id"
#else
#define SQLTXT1A "\
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 1, no_o_id, new_order.rowid, o_c_id, \
orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 1 AND o_w_id = :w_id AND o_d_id = 1 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
"

#define SQLTXT1B "\
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 2, no_o_id, new_order.rowid, o_c_id, \
orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 2 AND o_w_id = :w_id AND o_d_id = 2 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
"

#define SQLTXT1C "\
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 3, no_o_id, new_order.rowid, o_c_id, \
orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 3 AND o_w_id = :w_id AND o_d_id = 3 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
"

#define SQLTXT1D "\
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 4, no_o_id, new_order.rowid, o_c_id, \
orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 4 AND o_w_id = :w_id AND o_d_id = 4 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
"

#define SQLTXT1E "\
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 5, no_o_id, new_order.rowid, o_c_id, \
orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 5 AND o_w_id = :w_id AND o_d_id = 5 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
"

#define SQLTXT1F "\
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 6, no_o_id, new_order.rowid, o_c_id, \
orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 6 AND o_w_id = :w_id AND o_d_id = 6 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
"

#define SQLTXT1G "\
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 7, no_o_id, new_order.rowid, o_c_id, \
orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 7 AND o_w_id = :w_id AND o_d_id = 7 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
"
#endif
#endif

```

```
#define SQLTXT1H "\
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 8, no_o_id, new_order.rowid, o_c_id, \
orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 8 AND o_w_id = :w_id AND o_d_id = 8 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
"

#define SQLTXT1I "\
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 9, no_o_id, new_order.rowid, o_c_id, \
orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 9 AND o_w_id = :w_id AND o_d_id = 9 AND \
o_id = no_o_id AND rownum <= 1 UNION ALL \
"

#define SQLTXT1J "\
SELECT /*+ USE_NL(NEW_ORDER ORDERS) ORDERED */ 10, no_o_id, new_order.rowid, o_c_id, \
orders.rowid \
FROM new_order, orders \
WHERE no_w_id = :w_id AND no_d_id = 10 AND o_w_id = :w_id AND o_d_id = 10 AND \
o_id = no_o_id AND rownum <= 1"

#define SQLTXT2 "DELETE FROM new_order WHERE rowid = :no_rowid"
#endif

#ifdef DMLRETDEL
#define SQLTXT3 "UPDATE orders 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"
#else
#define SQLTXT3 "UPDATE orders SET o_carrier_id = :carrier_id \
WHERE rowid = :o_rowid"
#endif

#ifdef DMLRETDEL
#define SQLTXT4 "UPDATE /*+ buffer */ order_line SET ol_delivery_d = :cr_date \
WHERE ol_w_id = :w_id AND ol_d_id = :d_id AND ol_o_id = :o_id \
RETURNING ol_amount into :ol_amount"
#else
#define SQLTXT4 "UPDATE order_line SET ol_delivery_d = :cr_date \
WHERE ol_w_id = :w_id AND ol_d_id = :d_id AND ol_o_id = :o_id"

#define SQLTXT5A "\
SELECT :d_id1, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id1 AND ol_o_id = :o_id1 UNION ALL \
SELECT :d_id2, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id2 AND ol_o_id = :o_id2 UNION ALL \
"

#define SQLTXT5B "\
SELECT :d_id3, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id3 AND ol_o_id = :o_id3 UNION ALL \
SELECT :d_id4, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id4 AND ol_o_id = :o_id4 UNION ALL \
"

#define SQLTXT5C "\
SELECT :d_id5, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id5 AND ol_o_id = :o_id5 UNION ALL \
SELECT :d_id6, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id6 AND ol_o_id = :o_id6 UNION ALL \
"

#define SQLTXT5D "\
SELECT :d_id7, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id7 AND ol_o_id = :o_id7 UNION ALL \
SELECT :d_id8, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id8 AND ol_o_id = :o_id8 UNION ALL \
"

#define SQLTXT5E "\
SELECT :d_id9, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id9 AND ol_o_id = :o_id9 UNION ALL \
SELECT :d_id10, SUM(ol_amount) FROM order_line WHERE ol_w_id = :w_id AND \
ol_d_id = :d_id10 AND ol_o_id = :o_id10"

#endif

#ifdef TUX
#define SQLTXT6 "UPDATE customer 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 cons_ind[NDISTS];
    sb2 w_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];
    sb2 no_rowid_ind[NDISTS];
};

```



<pre> sb2 o_rowid_ind[NDISTS]; #if defined(ISO)    defined(ISO5)    defined(ISO6)    defined(ISO8) sb2 inum_ind; #endif  #ifdef DMLRETDL ub4 del_o_id_len[NDISTS]; ub4 c_id_len[NDISTS]; int oid_ctx; int cid_ctx; OCIBind *olamt_bp; #else ub2 del_o_id_len[NDISTS]; ub2 c_id_len[NDISTS]; #endif  ub2 cons_len[NDISTS]; 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 no_rowid_len[NDISTS]; ub2 no_rowid_ptr_len[NDISTS]; ub2 o_rowid_len[NDISTS]; ub2 o_rowid_ptr_len[NDISTS]; #if defined(ISO)    defined(ISO5)    defined(ISO6)    defined(ISO8) ub2 inum_len; #endif  ub2 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]; ub2 no_rowid_rcode[NDISTS]; ub2 o_rowid_rcode[NDISTS]; #if defined(ISO)    defined(ISO5)    defined(ISO6)    defined(ISO8) ub2 inum_rcode; #endif  int del_o_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]; #if defined(ISO)    defined(ISO5)    defined(ISO6)    defined(ISO8) char inum[10]; #endif  OCISmt *curd0; OCISmt *curd1; OCISmt *curd2; OCISmt *curd3; OCISmt *curd4; OCISmt *curd5; OCISmt *curd6; OCISmt *curdtest;  OCIBind *w_id_bp; OCIBind *w_id_bp3; OCIBind *w_id_bp4; OCIBind *w_id_bp5; OCIBind *w_id_bp6; OCIBind *d_id_bp; OCIBind *d_id_bp3; OCIBind *d_id_bp4; OCIBind *d_id_bp6; OCIBind *o_id_bp; OCIBind *cr_date_bp; OCIBind *c_id_bp; OCIBind *c_id_bp3; OCIBind *no_rowid_bp; OCIBind *carrier_id_bp; OCIBind *o_rowid_bp; OCIBind *del_o_id_bp; OCIBind *del_o_id_bp3; OCIBind *amt_bp; OCIBind *bstr1_bp[10]; OCIBind *bstr2_bp[10]; 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; </pre>	<pre> int norow; };  typedef struct delctx delctx;  /* delctx *dctx; */  #ifdef DMLRETDL struct amtctx { int ol_amt[NDISTS][NITEMS]; sb2 ol_amt_ind[NDISTS][NITEMS]; ub4 ol_amt_len[NDISTS][NITEMS]; ub2 ol_amt_rcode[NDISTS][NITEMS]; int ol_cnt[NDISTS]; }; typedef struct amtctx amtctx; /* amtctx *actx; */  #endif  #ifdef DMLRETDL extern sb4 no_data();  sb4 TPC_oid_data(dvoid *ctxp, OCIBind *bp, ub4 iter, ub4 index, dvoid **bufpp, ub4 **alenp, ub1 *piecep, dvoid **indpp, ub2 **rcodepp) { delctx *dctx = (delctx *)ctxp; *bufpp = &amp;dctx-&gt;del_o_id[iter]; *indpp = &amp;dctx-&gt;del_o_id_ind[iter]; dctx-&gt;del_o_id_len[iter] = sizeof(dctx-&gt;del_o_id[0]); *alenp = &amp;dctx-&gt;del_o_id_len[iter]; *rcodepp = &amp;dctx-&gt;del_o_id_rcode[iter]; *piecep = OCI_ONE_PIECE; return (OCI_CONTINUE); }  sb4 cid_data(dvoid *ctxp, OCIBind *bp, ub4 iter, ub4 index, dvoid **bufpp, ub4 **alenp, ub1 *piecep, dvoid **indpp, ub2 **rcodepp) { delctx *dctx = (delctx *)ctxp; *bufpp = &amp;dctx-&gt;c_id[iter]; *indpp = &amp;dctx-&gt;c_id_ind[iter]; dctx-&gt;c_id_len[iter] = sizeof(dctx-&gt;c_id[0]); *alenp = &amp;dctx-&gt;c_id_len[iter]; *rcodepp = &amp;dctx-&gt;c_id_rcode[iter]; *piecep = OCI_ONE_PIECE; return (OCI_CONTINUE); }  sb4 amt_data(dvoid *ctxp, OCIBind *bp, ub4 iter, ub4 index, dvoid **bufpp, ub4 **alenp, ub1 *piecep, dvoid **indpp, ub2 **rcodepp) { amtctx *actx; actx = (amtctx *)ctxp; actx-&gt;ol_cnt[iter] = actx-&gt;ol_cnt[iter] + 1; *bufpp = &amp;actx-&gt;ol_amt[iter][index]; *indpp = &amp;actx-&gt;ol_amt_ind[iter][index]; actx-&gt;ol_amt_len[iter][index] = sizeof(actx-&gt;ol_amt[0][0]); *alenp = &amp;actx-&gt;ol_amt_len[iter][index]; *rcodepp = &amp;actx-&gt;ol_amt_rcode[iter][index]; *piecep = OCI_ONE_PIECE; return (OCI_CONTINUE); }  #endif  tkvcdinit (ora_cn_data_t *ora_SlotDataP) { int i,j; char bstr1[10]; char bstr2[10]; text stmbuf[SQL_BUF_SIZE];  delctx *dctx; amtctx *actx; global_delivery_t *delP; OCISvcCtx *tpcenv = ora_SlotDataP-&gt;tpcenv; OCIServer *tpcsrv = ora_SlotDataP-&gt;tpcsrv; OCIError *errhp = ora_SlotDataP-&gt;errhp; OCISvcCtx *tpcsvc = ora_SlotDataP-&gt;tpcsvc; OCISession *tpcusr = ora_SlotDataP-&gt;tpcusr; OCISmt *curi = ora_SlotDataP-&gt;curi;  dctx = (delctx *) malloc (sizeof(delctx)); memset(dctx,(char)0,sizeof(delctx)); dctx-&gt;norow = 0;  ora_SlotDataP-&gt;dctx = (void *)dctx; delP = (global_delivery_t *) malloc(sizeof(global_delivery_t)); memset(delP, (char)0, sizeof(global_delivery_t)); ora_SlotDataP-&gt;delP = delP; </pre>
---	--

<pre> #ifdef DMLRETDDEL actx = (amtctx *) malloc (sizeof(amtctx)); memset(actx,(char)0,sizeof(amtctx)); ora_SlotDataP-&gt;actx = (void *)actx; #else for(i=0;i&lt;NDISTS;i++) { OCIERROR(errhp, OCIDescriptorAlloc(tpcenv,(dvoid**)&amp;detx-&gt;o_rowid_ptr[i], OCI_DTYPE_ROWID,0,(dvoid**)0)); OCIERROR(errhp, OCIDescriptorAlloc(tpcenv,(dvoid**)&amp;detx-&gt;o_rowid_ptr[i], OCI_DTYPE_ROWID,0,(dvoid**)0)); } #endif  #if defined(ISO)    defined(ISO5)    defined(ISO6)    defined(ISO8) OCIHandleAlloc(tpcenv, (dvoid **)&amp;detx-&gt;curd0, OCI_HTYPE_STMT, 0, (dvoid**)0); sprintf((char *) stmbuf, SQLTXTO); OCIStmtPrepare(detx-&gt;curd0, errhp, stmbuf, strlen((char *)stmbuf),OCI_NTV_SYNTAX, OCI_DEFAULT);  OCIDFNRA(detx-&gt;curd0, detx-&gt;inum_dp,errhp,1,detx-&gt;inum,SIZ(detx-&gt;inum),SQLT_STR, &amp;(detx-&gt;inum_ind),&amp;(detx-&gt;inum_len),&amp;(detx-&gt;inum_rcode)); #endif  /* If PLSQDEL and ISO? are both defined, then they both try to use curd0! This could cause a problem. Will try to fix later - VMM 12/30/97 */  #ifdef PLSQDEL OCIHandleAlloc(tpcenv, (dvoid **)&amp;detx-&gt;curd0, OCI_HTYPE_STMT, 0, (dvoid**)0); sprintf((char *) stmbuf, SQLTXTO); OCIStmtPrepare(detx-&gt;curd0, errhp, stmbuf, strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT); OCIBND(detx-&gt;curd0, detx-&gt;w_id_bp, errhp,"w_id",ADR(delP-&gt;w_id),SIZ(int), SQLT_INT); OCIBND(detx-&gt;curd0, detx-&gt;carrier_id_bp, errhp,"carrier_id", ADR(detx-&gt;carrier_id), SIZ(int), SQLT_INT);  OCIBNDRAA(detx-&gt;curd0, detx-&gt;o_id_bp, errhp, "order_id", detx-&gt;del_o_id,SIZ(int),SQLT_INT,detx-&gt;del_o_id_ind, detx-&gt;del_o_id_len,detx-&gt;del_o_id_rcode,NDISTS, &amp;detx-&gt;del_o_id_rent); OCIBND(detx-&gt;curd0, detx-&gt;retry_bp, errhp,"retry",ADR(detx-&gt;retry), SIZ(int),SQLT_INT); #else #ifdef DMLRETDDEL OCIHandleAlloc(tpcenv, (dvoid **)&amp;detx-&gt;curd1, OCI_HTYPE_STMT, 0, (dvoid**)0); sprintf((char *) stmbuf, "%s", SQLTXT1); OCIStmtPrepare(detx-&gt;curd1, errhp, stmbuf, strlen((char *)stmbuf),OCI_NTV_SYNTAX, OCI_DEFAULT);  OCIBND(detx-&gt;curd1, detx-&gt;w_id_bp,errhp,"w_id",detx-&gt;w_id,SIZ(int), SQLT_INT); OCIBNDRA(detx-&gt;curd1, detx-&gt;d_id_bp,errhp,"d_id",detx-&gt;d_id,SIZ(int), SQLT_INT,NULL,NULL,NULL);  OCIBNDRAD(detx-&gt;curd1, detx-&gt;del_o_id_bp, errhp, ":o_id", SIZ(int),SQLT_INT,NULL, (dvoid *)detx,no_data,TPC_oid_data); #else OCIHandleAlloc(tpcenv, (dvoid **)&amp;detx-&gt;curd1, OCI_HTYPE_STMT, 0, (dvoid**)0); sprintf((char *) stmbuf, "%s%s%s%s%s%s%s%s", SQLTXT1A, SQLTXT1B, SQLTXT1C, SQLTXT1D, SQLTXT1E, SQLTXT1F, SQLTXT1G, SQLTXT1H, SQLTXT1I, SQLTXT1J ); OCIStmtPrepare(detx-&gt;curd1, errhp, stmbuf, strlen((char *)stmbuf),OCI_NTV_SYNTAX, OCI_DEFAULT);  OCIERROR(errhp, OCIAttrSet(detx-&gt;curd1,OCI_HTYPE_STMT,(dvoid **)&amp;detx-&gt;norow,0, OCI_ATTR_PREFETCH_ROWS,errhp));  /* bind variables */ OCIBND(detx-&gt;curd1, detx-&gt;w_id_bp,errhp,"w_id",ADR(delP-&gt;w_id),SIZ(int),SQLT_INT);  OCIDFNRA(detx-&gt;curd1, detx-&gt;d_id_dp,errhp,1,detx-&gt;d_id,SIZ(int), SQLT_INT, detx-&gt;d_id_ind,detx-&gt;d_id_len,detx-&gt;d_id_rcode); OCIDFNRA(detx-&gt;curd1, detx-&gt;del_o_id_dp,errhp,2,detx-&gt;del_o_id, SIZ(int), SQLT_INT,detx-&gt;del_o_id_ind, detx-&gt;del_o_id_len, detx-&gt;del_o_id_rcode); </pre>	<pre> OCIDFNRA(detx-&gt;curd1, detx-&gt;no_rowid_dp,errhp,3,detx-&gt;no_rowid_ptr, SIZ(OCIRowid *), SQLT_RDD,detx-&gt;no_rowid_ind, detx-&gt;no_rowid_len, detx-&gt;no_rowid_rcode); OCIDFNRA(detx-&gt;curd1, detx-&gt;c_id_dp,errhp,4,detx-&gt;c_id,SIZ(detx-&gt;c_id[0]), SQLT_INT, detx-&gt;c_id_ind,detx-&gt;c_id_len,detx-&gt;c_id_rcode); OCIDFNRA(detx-&gt;curd1, detx-&gt;o_rowid_dp,errhp,5,detx-&gt;o_rowid_ptr, SIZ(OCIRowid *), SQLT_RDD,detx-&gt;o_rowid_ind, detx-&gt;o_rowid_len, detx-&gt;o_rowid_rcode);  /* open second cursor */  OCIHandleAlloc(tpcenv, (dvoid **)&amp;detx-&gt;curd2, OCI_HTYPE_STMT, 0, (dvoid**)0); sprintf((char *) stmbuf, SQLTXT2); OCIStmtPrepare(detx-&gt;curd2, errhp, stmbuf, strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT);  /* bind variables */ OCIBNDRA(detx-&gt;curd2, detx-&gt;no_rowid_bp,errhp,"no_rowid",&amp;(detx-&gt;no_rowid_ptr[0]), SIZ(detx-&gt;no_rowid_ptr[0]),SQLT_RDD,detx-&gt;no_rowid_ind, detx-&gt;no_rowid_len,detx-&gt;no_rowid_rcode);  #endif /*DMLRETDDEL*/  /* open third cursor */  OCIHandleAlloc(tpcenv, (dvoid **)&amp;detx-&gt;curd3, OCI_HTYPE_STMT, 0, (dvoid**)0); sprintf((char *) stmbuf, SQLTXT3); OCIStmtPrepare(detx-&gt;curd3, errhp, stmbuf, strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT);  /* bind variables */ OCIBNDRA(detx-&gt;curd3, detx-&gt;carrier_id_bp,errhp,"carrier_id",detx-&gt;carrier_id, SIZ(detx-&gt;carrier_id[0]),SQLT_INT,detx-&gt;carrier_id_ind, detx-&gt;carrier_id_len,detx-&gt;carrier_id_rcode);  #ifdef DMLRETDDEL OCIBNDRA(detx-&gt;curd3, detx-&gt;w_id_bp3, errhp, "w_id", detx-&gt;w_id,SIZ(int), SQLT_INT, NULL, NULL, NULL); OCIBNDRA(detx-&gt;curd3, detx-&gt;d_id_bp3, errhp, "d_id", detx-&gt;d_id,SIZ(int), SQLT_INT,NULL, NULL, NULL); OCIBNDRA(detx-&gt;curd3, detx-&gt;del_o_id_bp3, errhp, ":o_id", detx-&gt;del_o_id, SIZ(int), SQLT_INT,NULL,NULL,NULL); OCIBNDRAD(detx-&gt;curd3, detx-&gt;c_id_bp3, errhp, ":o_c_id", SIZ(int), SQLT_INT,NULL,(dvoid *)detx,no_data, cid_data); #else OCIBNDRA(detx-&gt;curd3, detx-&gt;o_rowid_bp,errhp,"o_rowid",&amp;(detx-&gt;o_rowid_ptr[0]), SIZ(detx-&gt;o_rowid_ptr[0]),SQLT_RDD,detx-&gt;o_rowid_ind, detx-&gt;o_rowid_ptr_len,detx-&gt;o_rowid_rcode);  #endif  /* open fourth cursor */  OCIHandleAlloc(tpcenv, (dvoid **)&amp;detx-&gt;curd4, OCI_HTYPE_STMT, 0, (dvoid**)0); sprintf((char *) stmbuf, SQLTXT4); OCIStmtPrepare(detx-&gt;curd4, errhp, stmbuf, strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT);  /* bind variables */ OCIBND(detx-&gt;curd4, detx-&gt;w_id_bp4,errhp,"w_id",detx-&gt;w_id, SIZ(int), SQLT_INT); OCIBND(detx-&gt;curd4, detx-&gt;d_id_bp4,errhp,"d_id",detx-&gt;d_id, SIZ(int), SQLT_INT); OCIBND(detx-&gt;curd4, detx-&gt;o_id_bp,errhp,"o_id",detx-&gt;del_o_id, SIZ(int),SQLT_INT); OCIBND(detx-&gt;curd4, detx-&gt;cr_date_bp,errhp,"cr_date", detx-&gt;del_date, SIZ(OCIDate), SQLT_ODT);  #ifdef DMLRETDDEL OCIBNDRAD(detx-&gt;curd4, detx-&gt;olam_bp, errhp, ":ol_amount", SIZ(int), SQLT_INT,NULL, actx,no_data,amt_data); #else  /* open fifth cursor */  OCIHandleAlloc(tpcenv, (dvoid **)&amp;detx-&gt;curd5, OCI_HTYPE_STMT, 0, (dvoid**)0); sprintf((char *) stmbuf, "%s%s%s%s%s", SQLTXT5A, SQLTXT5B, SQLTXT5C, SQLTXT5D, SQLTXT5E ); OCIStmtPrepare(detx-&gt;curd5, errhp, stmbuf, strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT);  OCIERROR(errhp, OCIAttrSet(detx-&gt;curd5,OCI_HTYPE_STMT,(dvoid **)&amp;detx-&gt;norow,0, OCI_ATTR_PREFETCH_ROWS,errhp));  /* bind variables */ OCIBND(detx-&gt;curd5,detx-&gt;w_id_bp,errhp,"w_id",ADR(delP-&gt;w_id),SIZ(delP-&gt;w_id),SQLT_INT); for (i = 0; i &lt; NDISTS; i++) { sprintf (bstr1, "d_id%d", i + 1); sprintf (bstr2, "o_id%d", i + 1); </pre>
--	---

<pre> OCIBNDRA(dctx-&gt;curd5,dctx-&gt;bstr1_bp[i],errhp,bstr1,ADR(dctx-&gt;d_id[i]), SIZ(dctx-&gt;d_id[0]),SQLT_INT,&amp;(dctx-&gt;d_id_ind[i]), &amp;(dctx-&gt;d_id_len[i]),&amp;(dctx-&gt;d_id_rcode[i])); OCIBNDRA(dctx-&gt;curd5,dctx-&gt;bstr2_bp[i],errhp,bstr2,ADR(dctx-&gt;del_o_id[i]), SIZ(dctx-&gt;del_o_id[0]),SQLT_INT,&amp;(dctx-&gt;del_o_id_ind[i]), &amp;(dctx-&gt;del_o_id_len[i]),&amp;(dctx-&gt;del_o_id_rcode[i])); }  OCIDFNRA(dctx-&gt;curd5,dctx-&gt;cons_dp,errhp,1,dctx-&gt;cons,SIZ(dctx-&gt;cons[0]),SQLT_INT, dctx-&gt;cons_ind,dctx-&gt;cons_len,dctx-&gt;cons_rcode); OCIDFNRA(dctx-&gt;curd5,dctx-&gt;amt_dp,errhp,2,dctx-&gt;amt,SIZ(dctx-&gt;amt[0]),SQLT_INT, dctx-&gt;amt_ind,dctx-&gt;amt_len,dctx-&gt;amt_rcode); #endif /* open sixth cursor */  OCIHandleAlloc(tpcenv, (dvoid **)&amp;dctx-&gt;curd6, OCI_HTYPE_STMT, 0, (dvoid**)0); sprintf((char *) stmbuf, SQLTXT6); OCIStmtPrepare(dctx-&gt;curd6, errhp, stmbuf, strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT);  /* bind variables */  OCIBND(dctx-&gt;curd6,dctx-&gt;amt_bp,errhp,"amt",dctx-&gt;amt,SIZ(int), SQLT_INT); OCIBND(dctx-&gt;curd6,dctx-&gt;w_id_bp,errhp,"w_id",dctx-&gt;w_id,SIZ(int), SQLT_INT); OCIBND(dctx-&gt;curd6,dctx-&gt;d_id_bp,errhp,"d_id",dctx-&gt;d_id,SIZ(int), SQLT_INT); OCIBND(dctx-&gt;curd6,dctx-&gt;c_id_bp,errhp,"c_id",dctx-&gt;c_id,SIZ(int), SQLT_INT); #endif return (0); }  void shiftdata(dctx *dctx,int from) { int i; for (i=from;i&lt;NDISTS-1;i++) { dctx-&gt;del_o_id_ind[i] = dctx-&gt;del_o_id_ind[i+1]; dctx-&gt;del_o_id[i] = dctx-&gt;del_o_id[i+1]; dctx-&gt;w_id[i] = dctx-&gt;w_id[i+1]; dctx-&gt;d_id[i] = dctx-&gt;d_id[i+1]; dctx-&gt;carrier_id[i] = dctx-&gt;carrier_id[i+1]; } }  kvcd(ora_cn_data_t *ora_SlotDataP) { int i,j,v; int rpe,rcount,count; int invalid; int tmp_id; int tmp_amt;  dctx *dctx = (dctx *)ora_SlotDataP-&gt;dctx; #ifdef DMLRETDDEL /* VMM 1/13/98 */ amtctx *actx = (amtctx *)ora_SlotDataP-&gt;actx; #endif /* DMLRETDDEL */ global_delivery_t *delP = ora_SlotDataP-&gt;delP; OCIEnv *tpcenv = ora_SlotDataP-&gt;tpcenv; OCIServer *tpcsrv = ora_SlotDataP-&gt;tpcsrv; OCIError *errhp = ora_SlotDataP-&gt;errhp; OCISvcCtx *tpscvc = ora_SlotDataP-&gt;tpscvc; OCISession *tpcusr = ora_SlotDataP-&gt;tpcusr; OCIStmt *curi = ora_SlotDataP-&gt;curi;  #if defined(ISO)    defined(ISO5)    defined(ISO6)    defined(ISO8) int hasno; int reread; char sdate[30];  OCIStmtExecute(tpscvc,dctx-&gt;curd0,errhp,1,0,0,OCI_DEFAULT); sysdate (sdate); printf ("Delivery started at %s on mode %s\n", sdate, dctx-&gt;inum); #endif #ifdef PLSQDEL for (i = 0; i &lt; NDISTS; i++) { dctx-&gt;del_o_id_ind[i] = TRUE; dctx-&gt;del_o_id_len[i] = sizeof(int); }  OCIERROR(errhp, OCIStmtExecute(tpscvc,dctx-&gt;curd0,errhp,1,0,0,OCI_DEFAULT));  for (i = 0; i &lt; NDISTS; i++) { delP-&gt;del_o_id[i] = 0; if (dctx-&gt;del_o_id_ind[i] == 0) { delP-&gt;del_o_id[i] = dctx-&gt;del_o_id[i]; } } </pre>	<pre> } #else retry:  #if defined(ISO)    defined(ISO5)    defined(ISO6)    defined(ISO8) reread = 1; #endif  iso:  invalid = 0;  /* initialization for array operations */  for (i = 0; i &lt; NDISTS; i++) { dctx-&gt;del_o_id_ind[i] = TRUE; dctx-&gt;cons_ind[i] = TRUE; dctx-&gt;w_id_ind[i] = TRUE; dctx-&gt;d_id_ind[i] = TRUE; dctx-&gt;c_id_ind[i] = TRUE; dctx-&gt;del_date_ind[i] = TRUE; dctx-&gt;carrier_id_ind[i] = TRUE; dctx-&gt;amt_ind[i] = TRUE; dctx-&gt;no_rowid_ind[i] = TRUE; dctx-&gt;o_rowid_ind[i] = TRUE;  dctx-&gt;del_o_id_len[i] = SIZ(dctx-&gt;del_o_id[0]); dctx-&gt;cons_len[i] = SIZ(dctx-&gt;cons[0]); dctx-&gt;w_id_len[i] = SIZ(dctx-&gt;w_id[0]); dctx-&gt;d_id_len[i] = SIZ(dctx-&gt;d_id[0]); dctx-&gt;c_id_len[i] = SIZ(dctx-&gt;c_id[0]); dctx-&gt;del_date_len[i] = DEL_DATE_LEN; dctx-&gt;carrier_id_len[i] = SIZ(dctx-&gt;carrier_id[0]); dctx-&gt;amt_len[i] = SIZ(dctx-&gt;amt[0]); dctx-&gt;no_rowid_len[i] = ROWIDLEN; dctx-&gt;o_rowid_len[i] = ROWIDLEN; dctx-&gt;o_rowid_ptr_len[i] = SIZ(dctx-&gt;o_rowid_ptr[0]); dctx-&gt;no_rowid_ptr_len[i] = SIZ(dctx-&gt;no_rowid_ptr[0]);  dctx-&gt;w_id[i] = delP-&gt;w_id; dctx-&gt;d_id[i] = i+1; dctx-&gt;carrier_id[i] = delP-&gt;o_carrier_id; memcpy(&amp;dctx-&gt;del_date[i],&amp;delP-&gt;cr_date,sizeof(OCIDate)); }  #ifdef DMLRETDDEL /* VMM 1/13/98 */ memset(actx,(char)0,sizeof(amtctx)); #endif /* DMLRETDDEL */ /* array select from new_order and orders tables */  delP-&gt;execstatus=OCIStmtExecute(tpscvc,dctx-&gt;curd1,errhp,NDISTS,0,0,OCI_DEFAULT); if((delP-&gt;execstatus != OCI_SUCCESS) &amp;&amp; (delP-&gt;execstatus != OCI_NO_DATA)) { OCITransRollback(tpscvc,errhp,OCI_DEFAULT); delP-&gt;errcode = OCIERROR(errhp,delP-&gt;execstatus); if(delP-&gt;errcode == NOT_SERIALIZABLE) { delP-&gt;retries++; goto retry; } else if (delP-&gt;errcode == RECOVER) { delP-&gt;retries++; goto retry; } else { return -1; } }  /* mark districts with no new order */ OCIAttrGet(dctx-&gt;curd1,OCI_HTYPE_STMT,&amp;rcount,0,OCI_ATTR_ROW_COUNT,errhp); rpe = rcount; #ifdef DMLRETDDEL /* we have to compress the array here */ if (rcount != NDISTS) { int j = 0; for (i=0;i &lt; NDISTS; i++) { if (dctx-&gt;del_o_id_ind[i] == 0) /* there is data here */ j++; else shiftdata(dctx, j); } } #else invalid = NDISTS - rcount; for (i = rpe; i &lt; NDISTS; i++) { dctx-&gt;del_o_id_ind[i] = NA; dctx-&gt;w_id_ind[i] = NA; dctx-&gt;d_id_ind[i] = NA; dctx-&gt;c_id_ind[i] = NA; dctx-&gt;carrier_id_ind[i] = NA; dctx-&gt;no_rowid_ind[i] = NA; dctx-&gt;o_rowid_ind[i] = NA; } #endif }  #if defined(ISO)    defined(ISO5)    defined(ISO6)    defined(ISO8) if (invalid) { sysdate (sdate); for (i = 1; i &lt;= NDISTS; i++) { hasno = 0; for (j = 0; j &lt; rpe; j++) { </pre>
--	--

```

        if (dctx->d_id[j] == i) {
            hasno = 1;
            break;
        }
    }
    if (!hasno)
        printf ("Delivery [dist %d] found no new order at %s\n", i, sdate);
    }
    if (reread) {
        sleep (60);
        sysdate (sdate);
        printf ("Delivery wake up at %s\n", sdate);
        reread = 0;
        goto iso;
    }
}
#endif

#ifndef DMLRETDEL
/* array delete of new_order table */
delP->execstatus=OCISmtExecute(tpsvc,dctx->curd2,errhp,rcp,0,0,OCI_DEFAULT);
if (delP->execstatus != OCI_SUCCESS) {
    OCITransRollback(tpsvc,errhp,OCI_DEFAULT);
    delP->errcode = OCIERROR(errhp,delP->execstatus);
    if (delP->errcode == NOT_SERIALIZABLE) {
        delP->retries++;
        goto retry;
    } else if (delP->errcode == RECOVERR) {
        delP->retries++;
        goto retry;
    } else {
        return -1;
    }
}

/* mark districts with no new order */
OCIAAttrGet(dctx->curd2,OCI_HTYPE_STMT,&rcount,0,OCI_ATTR_ROW_COUNT,errhp);

if (rcount != rpc) {
#ifdef TUX
    userlog ("Error in TPC-C server %d: %d rows selected, %d rows deleted\n",
            proc_no, rpc, dctx->curd2.rpc);
#else
    fprintf (stderr,
            "Error in TPC-C server %d: %d rows selected, %d rows deleted\n",
            proc_no, rpc, rcount);
#endif /* TUX */
    OCITransRollback(tpsvc,errhp,OCI_DEFAULT);
    return (DEL_ERROR);
}
#endif /* DMLRETDEL */

delP->execstatus=OCISmtExecute(tpsvc,dctx->curd3,errhp,rcp,0,0,OCI_DEFAULT);
if (delP->execstatus != OCI_SUCCESS) {
    OCITransRollback(tpsvc,errhp,OCI_DEFAULT);
    delP->errcode = OCIERROR(errhp,delP->execstatus);
    if (delP->errcode == NOT_SERIALIZABLE) {
        delP->retries++;
        goto retry;
    } else if (delP->errcode == RECOVERR) {
        delP->retries++;
        goto retry;
    } else {
        return -1;
    }
}

OCIAAttrGet(dctx->curd3,OCI_HTYPE_STMT,&rcount,0,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
    fprintf (stderr,
            "Error in TPC-C server %d: %d rows selected, %d ords updated\n",
            proc_no, rpc, rcount);
#endif
}
OCITransRollback(tpsvc,errhp,OCI_DEFAULT);
return (-1);
}

/* array update of order_line table */
delP->execstatus=OCISmtExecute(tpsvc,dctx->curd4,errhp,rcp,0,0,OCI_DEFAULT);
if (delP->execstatus != OCI_SUCCESS) {
    OCITransRollback(tpsvc,errhp,OCI_DEFAULT);
    delP->errcode = OCIERROR(errhp,delP->execstatus);
    if (delP->errcode == NOT_SERIALIZABLE) {
        delP->retries++;
        goto retry;
    } else if (delP->errcode == RECOVERR) {
        delP->retries++;
        goto retry;
    } else {
        return -1;
    }
}
}
#endif DMLRETDEL
OCIAAttrGet(dctx->curd4,OCI_HTYPE_STMT,&rcount,NULL,OCI_ATTR_ROW_COUNT,errhp);

```

```

/* add up amounts */
count=0;
for (i=0;i<rpc;i++)
{
    dctx->amt[i]=0;
    for (j=0;j<actx->ol_ent[i];j++)
        if (actx->ol_amt_rcode[i][j] == 0)
        {
            dctx->amt[i] = dctx->amt[i] + actx->ol_amt[i][j];
            count = count+1;
        }
}
if (rcount > rpc*NITEMS) {
    userlog ("Error in TPC-C server %d: %d ordnrs updated, %d ordl updated\n",
            proc_no, rpc, rcount);
}
}
#else
/* array select from order_line table */
delP->execstatus=OCISmtExecute(tpsvc,dctx->curd5,errhp,rcp,0,0,OCI_DEFAULT);
if ((delP->execstatus != OCI_SUCCESS) && (delP->execstatus != OCI_NO_DATA)) {
    OCITransRollback(tpsvc,errhp,OCI_DEFAULT);
    delP->errcode = OCIERROR(errhp,delP->execstatus);
    if (delP->errcode == NOT_SERIALIZABLE) {
        delP->retries++;
        goto retry;
    } else if (delP->errcode == RECOVERR) {
        delP->retries++;
        goto retry;
    } else {
        return -1;
    }
}

OCIAAttrGet(dctx->curd5,OCI_HTYPE_STMT,&rcount,0,OCI_ATTR_ROW_COUNT,errhp);
if (rcount != rpc) {
#ifdef TUX
    userlog ("Error in TPC-C server %d: %d rows selected, %d ordl selected\n",
            proc_no, rpc, rcount);
#else
    fprintf (stderr,
            "Error in TPC-C server %d: %d rows selected, %d ordl selected\n",
            proc_no, rpc, rcount);
#endif
}
OCITransRollback(tpsvc,errhp,OCI_DEFAULT);
return (-1);
}

/* reorder amount selected if necessary */
for (i = 0; i < rpc; i++) {
    if (dctx->cons[i] != dctx->d_id[i]) {
#ifdef TUX
        userlog ("TPC-C server %d: reordering amount\n", proc_no);
#else
        fprintf (stderr, "TPC-C server %d: reordering amount\n", proc_no);
#endif
}
for (j = i + 1; j < rpc; j++) {
    if (dctx->cons[j] == dctx->d_id[i]) {
        tmp_id = dctx->cons[j];
        dctx->cons[i] = dctx->cons[j];
        dctx->cons[j] = tmp_id;
        tmp_amt = dctx->amt[i];
        dctx->amt[i] = dctx->amt[j];
        dctx->amt[j] = tmp_amt;
        break;
    }
}
if (j >= rpc) {
#ifdef TUX
    userlog ("Error in TPC-C server %d: missing ordl?\n", proc_no);
#else
    fprintf (stderr,
            "Error in TPC-C server %d: missing ordl?\n", proc_no);
#endif
}
OCITransRollback(tpsvc,errhp,OCI_DEFAULT);
return (-1);
}
}
#endif
#ifdef ISO5 || defined(ISO6)
    printf ("d_id:amount\n");
    for (i = 0; i < rpc; i++)
        printf ("%d:%.2f ", dctx->d_id[i], (float)dctx->amt[i]/100);
    printf ("\n");
#endif

/* array update of customer table */
#ifdef ISO5 || defined(ISO6)
    execstatus=OCISmtExecute(tpsvc,dctx->curd6,errhp,rcp,0,0,OCI_DEFAULT);
#else
    delP->execstatus=OCISmtExecute(tpsvc,dctx->curd6,errhp,rcp,0,0,OCI_COMMIT_ON_SUCCESS | OCI_DEFAULT);
#endif

if (delP->execstatus != OCI_SUCCESS) {
    OCITransRollback(tpsvc,errhp,OCI_DEFAULT);
    delP->errcode = OCIERROR(errhp,delP->execstatus);
}

```

# plnew.c

```
if(delP->errcode == NOT_SERIALIZABLE) {
    delP->retries++;
    goto retry;
} else if (delP->errcode == RECOVER) {
    delP->retries++;
    goto retry;
} else {
    return -1;
}
}

OCIAttrGet(&delP->curd6,OCI_HTYPE_STMT,&rcount,0,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);
#endif
} else {
    fprintf (stderr,
            "Error in TPC-C server %d: %d rows selected, %d cust updated\n",
            proc_no, rpc, rcount);
}

#ifdef OCI_TRANS_ROLLBACK
    OCITransRollback(tpcvc, errhp, OCI_DEFAULT);
    return (-1);
}

#ifdef ISO5 || defined(ISO6)
    sysdate (sdate);
#ifdef ISO5
    printf ("Delivery sleep before commit at %s\n", sdate);
#else
    printf ("Delivery sleep before abort at %s\n", sdate);
#endif
    sleep (60);
    sysdate (sdate);
    printf ("Delivery wake up at %s\n", sdate);
#endif

#ifdef ISO6
    printf ("Delivery ISO6 Rolling back.\n");
    OCITransRollback(tpcvc, errhp, OCI_DEFAULT);
#endif

#ifdef ISO5
    OCITransCommit(tpcvc, errhp, OCI_DEFAULT);
#endif

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

/* return o_id's in district id order */
for (i = 0; i < NDISTS; i++)
    delP->del_o_id[i] = 0;
for (i = 0; i < rpc; i++)
    delP->del_o_id[delP->del_o_id[i] - 1] = delP->del_o_id[i];
}

return (0);
}

void tkvcdone (ora_cn_data_t *ora_SlotDataP)
{
    delP->del_o_id[delP->del_o_id[i] - 1] = delP->del_o_id[i];
    global_delivery_t *delP = ora_SlotDataP->delP;

    if (delP)
    {
#ifdef ISO || defined(ISO5) || defined(ISO6) || defined(ISO8)
        OCIHandleFree((dvoid *)delP->curd0,OCI_HTYPE_STMT);
#endif
#ifdef PLSQLEL
        OCIHandleFree((dvoid *)delP->curd0,OCI_HTYPE_STMT);
#else
        /* Again the above will cause a problem if both PLSQLEL and ISO are
           defined - VMM 12/30/97 */
        OCIHandleFree((dvoid *)delP->curd1,OCI_HTYPE_STMT);
        OCIHandleFree((dvoid *)delP->curd2,OCI_HTYPE_STMT);
        OCIHandleFree((dvoid *)delP->curd3,OCI_HTYPE_STMT);
        OCIHandleFree((dvoid *)delP->curd4,OCI_HTYPE_STMT);
        OCIHandleFree((dvoid *)delP->curd5,OCI_HTYPE_STMT);
        OCIHandleFree((dvoid *)delP->curd6,OCI_HTYPE_STMT);
#endif
    }

    free (delP);
    ora_SlotDataP->delP = NULL;
}

if (delP) {
    free(delP);
    ora_SlotDataP->delP = NULL;
}
}
```

```
#ifdef RCSID
static char *RCSid =
    "$Header: /afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/server/ora8.1_mt/RCS/plnew.c,v
    1.3 1999/05/26 16:29:56 wenjian 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"
#include "plora.h"
#ifdef TUX
#include <userlog.h>
#endif
#include "tpccflags.h"

extern void err_printf(char *format, ...);

#define PLSQLNO

#ifdef PLSQLNO
#define SQLTXT2 "BEGIN initnew.new_init(:idx1arr); END;"
#else
#define SQLTXT2 "UPDATE stock SET s_order_cnt = s_order_cnt + 1, \
    s_ytd = s_ytd + :ol_quantity, s_remote_cnt = s_remote_cnt + :s_remote, \
    s_quantity = :s_quantity \
    WHERE rowid = :s_rowid"

#define SQLTXT3 "\
SELECT 0,stock.rowid,i_price,i_name,i_data,s_dist %02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :10 AND s_w_id = :30 AND s_i_id = i_id UNION ALL \
SELECT 1,stock.rowid,i_price,i_name,i_data,s_dist %02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :11 AND s_w_id = :31 AND s_i_id = i_id UNION ALL \
SELECT 2,stock.rowid,i_price,i_name,i_data,s_dist %02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :12 AND s_w_id = :32 AND s_i_id = i_id UNION ALL \
SELECT 3,stock.rowid,i_price,i_name,i_data,s_dist %02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :13 AND s_w_id = :33 AND s_i_id = i_id UNION ALL \
SELECT 4,stock.rowid,i_price,i_name,i_data,s_dist %02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :14 AND s_w_id = :34 AND s_i_id = i_id UNION ALL \
SELECT 5,stock.rowid,i_price,i_name,i_data,s_dist %02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :15 AND s_w_id = :35 AND s_i_id = i_id UNION ALL \
SELECT 6,stock.rowid,i_price,i_name,i_data,s_dist %02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :16 AND s_w_id = :36 AND s_i_id = i_id UNION ALL \
SELECT 7,stock.rowid,i_price,i_name,i_data,s_dist %02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :17 AND s_w_id = :37 AND s_i_id = i_id UNION ALL \
SELECT 8,stock.rowid,i_price,i_name,i_data,s_dist %02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :18 AND s_w_id = :38 AND s_i_id = i_id UNION ALL \
SELECT 9,stock.rowid,i_price,i_name,i_data,s_dist %02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :19 AND s_w_id = :39 AND s_i_id = i_id UNION ALL \
SELECT 10,stock.rowid,i_price,i_name,i_data,s_dist %02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :20 AND s_w_id = :40 AND s_i_id = i_id UNION ALL \
SELECT 11,stock.rowid,i_price,i_name,i_data,s_dist %02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :21 AND s_w_id = :41 AND s_i_id = i_id UNION ALL \
SELECT 12,stock.rowid,i_price,i_name,i_data,s_dist %02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :22 AND s_w_id = :42 AND s_i_id = i_id UNION ALL \
SELECT 13,stock.rowid,i_price,i_name,i_data,s_dist %02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :23 AND s_w_id = :43 AND s_i_id = i_id UNION ALL \
SELECT 14,stock.rowid,i_price,i_name,i_data,s_dist %02d,s_data,s_quantity \
FROM item,stock WHERE i_id = :24 AND s_w_id = :44 AND s_i_id = i_id"

#define SQLTXT4 "INSERT INTO order_line \
    (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,:null_date,:ol_i_id,:ol_supply_w_id,:ol_quantity, \
    :ol_amount,:ol_dist_info)"
#endif /* PLSQLNO */

#define NITEMS 15
#define ROWIDLEN 20
#define OCIROWLEN 20

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);
}

struct newctx {
    sb2 nol_i_id_ind[NITEMS];
    sb2 nol_supply_w_id_ind[NITEMS];
    sb2 nol_quantity_ind[NITEMS];
};
```

<pre> sb2 nol_amount_ind[NITEMS]; sb2 i_name_ind[NITEMS]; sb2 s_quantity_ind[NITEMS]; sb2 i_price_ind[NITEMS]; sb2 ol_w_id_ind[NITEMS]; sb2 ol_d_id_ind[NITEMS]; sb2 ol_o_id_ind[NITEMS]; sb2 ol_number_ind[NITEMS]; sb2 cons_ind[NITEMS]; sb2 s_rowid_ind[NITEMS]; sb2 s_remote_ind[NITEMS]; sb2 s_quant_ind[NITEMS]; sb2 i_data_ind[NITEMS]; sb2 s_data_ind[NITEMS]; sb2 s_dist_info_ind[NITEMS]; sb2 ol_dist_info_ind[NITEMS]; sb2 null_date_ind[NITEMS]; #ifdef PLSQLNO sb2 s_bg_ind[NITEMS]; #endif #endif  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 i_data_len[NITEMS]; ub2 s_dist_info_len[NITEMS]; ub2 s_data_len[NITEMS]; ub2 ol_w_id_len[NITEMS]; ub2 ol_d_id_len[NITEMS]; ub2 ol_o_id_len[NITEMS]; ub2 ol_number_len[NITEMS]; ub2 cons_len[NITEMS]; ub2 s_rowid_len[NITEMS]; ub2 s_remote_len[NITEMS]; ub2 s_quant_len[NITEMS]; ub2 ol_dist_info_len[NITEMS]; ub2 null_date_len[NITEMS]; #ifdef PLSQLNO ub2 s_bg_len[NITEMS]; #endif #endif  ub2 nol_i_id_rcode[NITEMS]; ub2 nol_supply_w_id_rcode[NITEMS]; ub2 nol_quantity_rcode[NITEMS]; ub2 nol_amount_rcode[NITEMS]; ub2 i_name_rcode[NITEMS]; ub2 s_quantity_rcode[NITEMS]; ub2 i_price_rcode[NITEMS]; ub2 ol_w_id_rcode[NITEMS]; ub2 ol_d_id_rcode[NITEMS]; ub2 ol_o_id_rcode[NITEMS]; ub2 ol_number_rcode[NITEMS]; ub2 cons_rcode[NITEMS]; ub2 s_rowid_rcode[NITEMS]; ub2 s_remote_rcode[NITEMS]; ub2 s_quant_rcode[NITEMS]; ub2 i_data_rcode[NITEMS]; ub2 s_data_rcode[NITEMS]; ub2 s_dist_info_rcode[NITEMS]; ub2 ol_dist_info_rcode[NITEMS]; ub2 null_date_rcode[NITEMS]; #ifdef PLSQLNO ub2 s_bg_rcode[NITEMS]; #endif #endif  int ol_w_id[NITEMS]; int ol_d_id[NITEMS]; int ol_o_id[NITEMS]; int ol_number[NITEMS]; int cons[NITEMS];  OCIRowid *s_rowid_ptr[NITEMS];  int s_remote[NITEMS]; char i_data[NITEMS][51]; char s_data[NITEMS][51]; char s_dist_info[NITEMS][25]; OCIDate null_date[NITEMS]; /* base date for null date entry */ OCISmt *curn1; #ifdef PLSQLNO OCIBind *ol_i_id_bp; OCIBind *ol_supply_w_id_bp; OCIBind *i_price_bp; OCIBind *i_name_bp; OCIBind *s_bg_bp; OCIBind *s_data_bp; OCIBind *i_data_bp; ub4 nol_i_count; ub4 nol_s_count; ub4 nol_q_count; ub4 nol_item_count; ub4 nol_name_count; ub4 nol_qty_count; ub4 nol_bg_count; ub4 nol_am_count; </pre>	<pre> ub4 s_remote count; ub4 s_data count; ub4 i_data count; #endif OCISmt *curn2; OCISmt *curn3[10]; OCIBind *ol_i_id_bp4; OCIBind *ol_supply_w_id_bp4; OCIBind *ol_quantity_bp; OCIBind *ol_quantity_bp4; OCIBind *s_remote_bp; OCIBind *s_quantity_bp; OCISmt *curn4; 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 *s_rowid_bp; OCIBind *id_bp[10][15]; OCIBind *sd_bp[10][15]; OCIDefine *Dscons[10]; OCIDefine *Ds_rowid[10]; OCIDefine *Di_price[10]; OCIDefine *Di_data[10]; OCIDefine *Ds_dist_info[10]; OCIDefine *Ds_data[10]; OCIDefine *Ds_quantity[10]; OCIDefine *Di_name[10]; OCIBind *ol_o_id_bp; OCIBind *ol_d_id_bp; OCIBind *ol_w_id_bp; OCIBind *ol_number_bp; OCIBind *ol_amount_bp; OCIBind *ol_dist_info_bp; OCIBind *null_date_bp;  sb2 w_id_ind; ub2 w_id_len; ub2 w_id_rc;  sb2 d_id_ind; ub2 d_id_len; ub2 d_id_rc;  sb2 c_id_ind; ub2 c_id_len; ub2 c_id_rc;  sb2 o_all_local_ind; ub2 o_all_local_len; ub2 o_all_local_rc;  sb2 o_ol_cnt_ind; ub2 o_ol_cnt_len; ub2 o_ol_cnt_rc;  sb2 w_tax_ind; ub2 w_tax_len; ub2 w_tax_rc;  sb2 d_tax_ind; ub2 d_tax_len; ub2 d_tax_rc;  sb2 o_id_ind; ub2 o_id_len; ub2 o_id_rc;  sb2 c_discount_ind; ub2 c_discount_len; ub2 c_discount_rc;  sb2 c_credit_ind; ub2 c_credit_len; ub2 c_credit_rc;  sb2 c_last_ind; ub2 c_last_len; ub2 c_last_rc;  sb2 retries_ind; ub2 retries_len; ub2 retries_rc;  sb2 cr_date_ind; ub2 cr_date_len; ub2 cr_date_rc;  int cs; int norow; </pre>
---	--

```

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

typedef struct newctx newctx;

/* newctx *nctx; */

kvcninit (ora_cn_data_t *ora_SlotDataP)
{
    int i, j;
    text stmbuf[16*1024];
    char id[4];
    char sd[4];

    newctx *nctx;
    OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
    OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
    OCIErr *errhp = ora_SlotDataP->errhp;
    OCISvcCtx *tpscvc = ora_SlotDataP->tpscvc;
    OCISession *tpcsur = ora_SlotDataP->tpcsur;
    OCISmt *curi = ora_SlotDataP->curi;
    global_newOrder_t *newP;

    nctx = (newctx *) malloc (sizeof(newctx));
    memset(nctx, (char)0, sizeof(newctx));
    ora_SlotDataP->nctx = (void *)nctx;

    ora_SlotDataP->globals = (global_newOrder_t *) malloc (sizeof(global_newOrder_t));
    memset(ora_SlotDataP->globals, (char)0, sizeof(global_newOrder_t));
    newP = ora_SlotDataP->globals;

    nctx->cs = 1;
    nctx->norow = 0;
    for (i = 0; i < NITEMS; i++) {
        OCIErr *errhp, OCIDescriptorAlloc (tpcenv, (dvoid **)&nctx->s_rowid_ptr[i],
            OCI_DTYPE_ROWID, 0, (dvoid **)0);
    }
    nctx->w_id_ind = TRUE;
    nctx->w_id_len = sizeof(newP->w_id);
    nctx->d_id_ind = TRUE;
    nctx->d_id_len = sizeof(newP->d_id);
    nctx->c_id_ind = TRUE;
    nctx->c_id_len = sizeof(newP->c_id);
    nctx->o_all_local_ind = TRUE;
    nctx->o_all_local_len = sizeof(newP->o_all_local);
    nctx->o_of_cnt_ind = TRUE;
    nctx->o_of_cnt_len = sizeof(newP->o_of_cnt);
    nctx->w_tax_ind = TRUE;
    nctx->w_tax_len = 0;
    nctx->d_tax_ind = TRUE;
    nctx->d_tax_len = 0;
    nctx->o_id_ind = TRUE;
    nctx->o_id_len = sizeof(newP->o_id);
    nctx->c_discount_ind = TRUE;
    nctx->c_discount_len = 0;
    nctx->c_credit_ind = TRUE;
    nctx->c_credit_len = 0;
    nctx->c_last_ind = TRUE;
    nctx->c_last_len = 0;
    nctx->retries_ind = TRUE;
    nctx->retries_len = sizeof(newP->retries);
    nctx->cr_date_ind = TRUE;
    nctx->cr_date_len = sizeof(newP->cr_date);

    /* open first cursor */
    OCIErr *errhp, OCIHandleAlloc (tpcenv, (dvoid **)&nctx->cur1,
        OCI_HTYPE_STMT, 0, (dvoid **)0);
#ifdef PLSQLNO
    sqlfile ("tkvcnew.sql", stmbuf);
#else
    sqlfile ("tkvcnew.sql", stmbuf);
#endif
    OCIErr *errhp, OCISmtPrepare (nctx->cur1, errhp, stmbuf, strlen ((char *) stmbuf),
        OCI_NTV_SYNTAX, OCI_DEFAULT);

    /* bind variables */
    OCIBNDR (nctx->cur1, nctx->w_id_bp, errhp, "w_id", ADR (newP->w_id), SIZ (newP->w_id),
        SQLT_INT, &nctx->w_id_ind, &nctx->w_id_len, &nctx->w_id_rc);
    OCIBNDR (nctx->cur1, nctx->d_id_bp, errhp, "d_id", ADR (newP->d_id), SIZ (newP->d_id),
        SQLT_INT, &nctx->d_id_ind, &nctx->d_id_len, &nctx->d_id_rc);
    OCIBNDR (nctx->cur1, nctx->c_id_bp, errhp, "c_id", ADR (newP->c_id), SIZ (newP->c_id),
        SQLT_INT, &nctx->c_id_ind, &nctx->c_id_len, &nctx->c_id_rc);
    OCIBNDR (nctx->cur1, nctx->o_all_local_bp, errhp, "o_all_local",
        ADR (newP->o_all_local), SIZ (newP->o_all_local), SQLT_INT, &nctx->o_all_local_ind,
        &nctx->o_all_local_len, &nctx->o_all_local_rc);
    OCIBNDR (nctx->cur1, nctx->o_of_cnt_bp, errhp, "o_of_cnt", ADR (newP->o_of_cnt),
        SIZ (newP->o_of_cnt), SQLT_INT, &nctx->o_of_cnt_ind, &nctx->o_of_cnt_len,
        &nctx->o_of_cnt_rc);
    OCIBNDR (nctx->cur1, nctx->w_tax_bp, errhp, "w_tax", ADR (newP->w_tax), SIZ (newP->w_tax),
        SQLT_FLT, &nctx->w_tax_ind, &nctx->w_tax_len, &nctx->w_tax_rc);
    OCIBNDR (nctx->cur1, nctx->d_tax_bp, errhp, "d_tax", ADR (newP->d_tax), SIZ (newP->d_tax),
        SQLT_FLT, &nctx->d_tax_ind, &nctx->d_tax_len, &nctx->d_tax_rc);
    OCIBNDR (nctx->cur1, nctx->o_id_bp, errhp, "o_id", ADR (newP->o_id), SIZ (newP->o_id),
        SQLT_INT, &nctx->o_id_ind, &nctx->o_id_len, &nctx->o_id_rc);
    OCIBNDR (nctx->cur1, nctx->c_discount_bp, errhp, "c_discount",
        ADR (newP->c_discount), SIZ (newP->c_discount), SQLT_FLT,
        &nctx->c_discount_ind, &nctx->c_discount_len, &nctx->c_discount_rc);
    OCIBNDR (nctx->cur1, nctx->c_credit_bp, errhp, "c_credit", newP->c_credit,
        SIZ (newP->c_credit), SQLT_CHR,
        &nctx->c_credit_ind, &nctx->c_credit_len, &nctx->c_credit_rc);
    OCIBNDR (nctx->cur1, nctx->c_last_bp, errhp, "c_last", newP->c_last, SIZ (newP->c_last),
        SQLT_STR, &nctx->c_last_ind, &nctx->c_last_len, &nctx->c_last_rc);
    OCIBNDR (nctx->cur1, nctx->retries_bp, errhp, "retries", ADR (newP->retries),
        SIZ (newP->retries), SQLT_INT,
        &nctx->retries_ind, &nctx->retries_len, &nctx->retries_rc);
    OCIBNDR (nctx->cur1, nctx->cr_date_bp, errhp, "cr_date", &nctx->cr_date, SIZ (OCIDate),
        SQLT_ODT, &nctx->cr_date_ind, &nctx->cr_date_len, &nctx->cr_date_rc);

#ifdef PLSQLNO
    OCIBNDRAA (nctx->cur1, nctx->ol_id_bp, errhp, "ol_id", newP->ol_id,
        SIZ (int), SQLT_INT, nctx->ol_id_ind, nctx->ol_id_len,
        nctx->ol_id_rcode, NITEMS, &nctx->ol_id_count);
    OCIBNDRAA (nctx->cur1, nctx->ol_supply_w_id_bp, errhp, "ol_supply_w_id",
        newP->ol_supply_w_id, SIZ (int), SQLT_INT, nctx->ol_supply_w_id_ind,
        nctx->ol_supply_w_id_len, nctx->ol_supply_w_id_rcode,
        NITEMS, &nctx->ol_supply_w_id_count);
    OCIBNDRAA (nctx->cur1, nctx->ol_quantity_bp, errhp, "ol_quantity", newP->ol_quantity,
        SIZ (int), SQLT_INT, nctx->ol_quantity_ind, nctx->ol_quantity_len,
        nctx->ol_quantity_rcode, NITEMS, &nctx->ol_quantity_count);
    OCIBNDRAA (nctx->cur1, nctx->i_price_bp, errhp, "i_price", newP->i_price, SIZ (float),
        SQLT_FLT, nctx->i_price_ind, nctx->i_price_len, nctx->i_price_rcode,
        NITEMS, &nctx->ol_item_count);
    OCIBNDRAA (nctx->cur1, nctx->i_name_bp, errhp, "i_name", newP->i_name,
        SIZ (newP->i_name[0]), SQLT_STR, nctx->i_name_ind, nctx->i_name_len,
        nctx->i_name_rcode, NITEMS, &nctx->ol_name_count);
    OCIBNDRAA (nctx->cur1, nctx->s_quantity_bp, errhp, "s_quantity", newP->s_quantity,
        SIZ (int), SQLT_INT, nctx->s_quant_ind, nctx->s_quant_len,
        nctx->s_quant_rcode, NITEMS, &nctx->ol_qty_count);
    OCIBNDRAA (nctx->cur1, nctx->s_bg_bp, errhp, "brand_generic", newP->brand_generic,
        SIZ (char), SQLT_CHR, nctx->s_bg_ind, nctx->s_bg_len,
        nctx->s_bg_rcode, NITEMS, &nctx->ol_bg_count);
    OCIBNDRAA (nctx->cur1, nctx->ol_amount_bp, errhp, "ol_amount", newP->ol_amount,
        SIZ (int), SQLT_INT, nctx->ol_amount_ind, nctx->ol_amount_len,
        nctx->ol_amount_rcode, NITEMS, &nctx->ol_am_count);
    OCIBNDRAA (nctx->cur1, nctx->s_remote_bp, errhp, "s_remote", nctx->s_remote,
        SIZ (int), SQLT_INT, nctx->s_remote_ind, nctx->s_remote_len,
        nctx->s_remote_rcode, NITEMS, &nctx->s_remote_count);

    /* open second cursor */
    OCIErr *errhp, OCIHandleAlloc (tpcenv, (dvoid **)&nctx->cur2,
        OCI_HTYPE_STMT, 0, (dvoid **)0);

    sprintf ((char *) stmbuf, SQLTEXT2);
    OCIErr *errhp, OCISmtPrepare (nctx->cur2, errhp, stmbuf,
        strlen ((char *) stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT);

    /* execute second cursor to init newunit 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;
        newP->o_of_cnt = NITEMS;

        /* Bind array */
        OCIBNDRAA (nctx->cur2, idx1arr_bp, errhp, "idx1arr", idx1arr,
            SIZ (int), SQLT_INT, idx1arr_ind, idx1arr_len,
            idx1arr_rcode, NITEMS, &idx1arr_count);

        newP->execstatus = OCISmtExecute (tpscvc, nctx->cur2, errhp, 1, 0, 0, OCI_DEFAULT);
        if (newP->execstatus != OCI_SUCCESS) {
            OCITransRollback (tpscvc, errhp, OCI_DEFAULT);
            newP->errcode = OCIErr *errhp, newP->execstatus);
            return -1;
        }
    }
    #else
    /* open second cursor */
    OCIErr *errhp, OCIHandleAlloc (tpcenv, (dvoid **)&nctx->cur2, OCI_HTYPE_STMT,
        0, (dvoid **)0);

    sprintf ((char *) stmbuf, SQLTEXT2);
    OCIErr *errhp, OCISmtPrepare (nctx->cur2, errhp, stmbuf,
        strlen ((char *) stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT);

    /* bind variables */

```

```

OCIBNDRA(nctx->curm2, nctx->s_quantity_bp_errhp,":s_quantity",newP->s_quantity,
SIZ(int),SQLT_INT,nctx->s_quant_ind,nctx->s_quant_len,
nctx->s_quant_rcode);
OCIBNDRA(nctx->curm2, nctx->s_rowid_bp_errhp,":s_rowid",nctx->s_rowid_ptr,
sizeof(nctx->s_rowid_ptr[0]),SQLT_RDD,nctx->s_rowid_ind,
nctx->s_rowid_len,nctx->s_rowid_rcode);
OCIBNDRA(nctx->curm2, nctx->ol_quantity_bp_errhp,":ol_quantity",newP->ol_quantity,
SIZ(int),SQLT_INT,nctx->ol_quantity_ind,nctx->ol_quantity_len,
nctx->ol_quantity_rcode);
OCIBNDRA(nctx->curm2, nctx->s_remote_bp_errhp,":s_remote",nctx->s_remote,
SIZ(int),SQLT_INT,nctx->s_remote_ind,nctx->s_remote_len,
nctx->s_remote_rcode);

/* open third cursor and bind variables */
for (i = 0; i < 10; i++)
{
j = i + 1;
OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)&(nctx->curm3)[i]),
OCI_HTYPE_STMT, 0, (dvoid***)0);

sprintf((char *) stmbuf, SQLT3, j, j, j, j, j, j, j, j, j, j);

OCIERROR(errhp,OCIStmtPrepare((nctx->curm3)[i], errhp, stmbuf,
strlen((char *)stmbuf),OCI_NTV_SYNTAX,
OCI_DEFAULT));

OCIERROR(errhp,
OCIAttrSet(nctx->curm3[i],OCI_HTYPE_STMT,(dvoid **)&nctx->norow,0,
OCI_ATTR_PREFETCH_ROWS,errhp));
for (j = 0; j < NITEMS; j++)
{
sprintf(id, "%04d", j + 10);
sprintf(sd, "%04d", j + 30);
OCIBNDRA((nctx->curm3)[i],(nctx->id_bp)[i][j],errhp,id,ADR(newP->ol_id[j]),
SIZ(int),SQLT_INT,
&nctx->ol_id_ind[j],&nctx->ol_id_len[j],
&nctx->ol_id_rcode[j]);
OCIBNDRA((nctx->curm3)[i],(nctx->sd_bp)[i][j],errhp,sd,
ADR(nol_supply_w_id[j]),SIZ(int),SQLT_INT,
&nctx->nol_supply_w_id_ind[j],&nctx->nol_supply_w_id_len[j],
&nctx->nol_supply_w_id_rcode[j]);
nctx->nol_id_ind[j] = NA;
nctx->nol_supply_w_id_ind[j] = NA;
nctx->nol_id_len[j] = sizeof(int);
nctx->nol_supply_w_id_len[j] = sizeof(int);
}

OCIDEF((nctx->curm3)[i],(nctx->Dcons)[i],errhp,1,&(nctx->cons[0]),
SIZ(nctx->cons[0]),SQLT_INT);
OCIDEF((nctx->curm3)[i],(nctx->Ds_rowid)[i],errhp,2,
nctx->s_rowid_ptr, sizeof(nctx->s_rowid_ptr[0]), SQLT_RDD);
OCIDEF((nctx->curm3)[i],(nctx->Di_price)[i],errhp,3,newP->i_price,SIZ(int),
SQLT_INT);

OCIDFNRA((nctx->curm3)[i],(nctx->Di_name)[i],errhp,4,newP->i_name,
SIZ(i_name[0]),SQLT_STR, nctx->i_name_ind,nctx->i_name_len,
nctx->i_name_rcode);
OCIDFNRA((nctx->curm3)[i],(nctx->Di_data)[i],errhp,5,nctx->i_data,
SIZ(nctx->i_data[0]), SQLT_STR, NULL,nctx->i_data_len,NULL);
OCIDFNRA((nctx->curm3)[i],(nctx->Ds_dist_info)[i],errhp,6,
nctx->s_dist_info, SIZ(nctx->s_dist_info[0]),SQLT_STR,
NULL,nctx->s_dist_info_len, NULL);
OCIDFNRA((nctx->curm3)[i],(nctx->Ds_data)[i],errhp,7,nctx->s_data,
SIZ(nctx->s_data[0]),SQLT_STR,NULL,nctx->s_data_len,NULL);
OCIDEF((nctx->curm3)[i],(nctx->Ds_quantity)[i],errhp,8,newP->s_quantity,
SIZ(int),SQLT_INT);
}

/* open fourth cursor */
OCIHandleAlloc(tpcenv, (dvoid **)&(nctx->curm4), OCI_HTYPE_STMT, 0,
(dvoid***)0);
sprintf((char *) stmbuf, SQLT4);
OCIStmtPrepare(nctx->curm4, errhp, stmbuf, strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);

/* bind variables */
OCIBNDRA(nctx->curm4, nctx->ol_o_id_bp_errhp,":ol_o_id",nctx->ol_o_id,
SIZ(int),SQLT_INT, NULL,nctx->ol_o_id_len,
NULL);
OCIBNDRA(nctx->curm4, nctx->ol_d_id_bp_errhp,":ol_d_id",nctx->ol_d_id,
SIZ(int),SQLT_INT, NULL,nctx->ol_d_id_len,
NULL);
OCIBNDRA(nctx->curm4, nctx->ol_w_id_bp_errhp,":ol_w_id",nctx->ol_w_id,
SIZ(int),SQLT_INT, NULL,nctx->ol_w_id_len,
NULL);
OCIBNDRA(nctx->curm4, nctx->ol_number_bp_errhp,":ol_number",nctx->ol_number,
SIZ(int),SQLT_INT, NULL,nctx->ol_number_len,
NULL);
OCIBNDRA(nctx->curm4, nctx->ol_i_id_bp_errhp,":ol_i_id",newP->ol_i_id,SIZ(int),
SQLT_INT, NULL,nctx->ol_i_id_len, NULL);
OCIBNDRA(nctx->curm4, nctx->ol_supply_w_id_bp_errhp,":ol_supply_w_id",
newP->nol_supply_w_id,SIZ(int),SQLT_INT, NULL,
nctx->nol_supply_w_id_len, NULL);
OCIBNDRA(nctx->curm4, nctx->ol_quantity_bp_errhp,":ol_quantity",newP->ol_quantity,
SIZ(int),SQLT_INT, NULL,nctx->nol_quantity_len,
NULL);

OCIBNDRA(nctx->curm4, nctx->ol_amount_bp_errhp,":ol_amount",newP->ol_amount,
SIZ(int),SQLT_INT, NULL,nctx->nol_amount_len,
NULL);
OCIBNDRA(nctx->curm4, nctx->ol_dist_info_bp_errhp,":ol_dist_info",
nctx->s_dist_info, SIZ(nctx->s_dist_info[0]),SQLT_AFC,
NULL, nctx->ol_dist_info_len,
NULL);
OCIBNDRA(nctx->curm4, nctx->null_date_bp_errhp,":null_date",nctx->null_date,
SIZ(OCIDate), SQLT_ODT,NULL,
nctx->null_date_len, NULL);

/* set up the null date Null date is 15-sep-11 */
for (i=0;i<NITEMS;i++)
{
OCIDateSetDate(&nctx->null_date[i],(sb2)1811,(ub1)9,(ub1)15);
}
#endif

return (0);
}

tkvcn(ora_cn_data_t *ora_SlotDataP)
{
int i, j, k;
int rpc, rpc3, rowoff, iters,rcount;
ub4 flags;
int failed = 0;

OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
OCIError *errhp = ora_SlotDataP->errhp;
OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
OCISession *tpcsusr = ora_SlotDataP->tpcsusr;
OCIStmt *curi = ora_SlotDataP->curi;
global_newOrder_t *newP = ora_SlotDataP->globals;
newctx *nctx = (newctx *)ora_SlotDataP->nctx;

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

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

nctx->w_id_ind = TRUE;
nctx->w_id_len = sizeof(newP->w_id);
nctx->d_id_ind = TRUE;
nctx->d_id_len = sizeof(newP->d_id);
nctx->c_id_ind = TRUE;
nctx->c_id_len = sizeof(newP->c_id);
nctx->o_all_local_ind = TRUE;
nctx->o_all_local_len = sizeof(newP->o_all_local);
nctx->o_ol_cnt_ind = TRUE;
nctx->o_ol_cnt_len = sizeof(newP->o_ol_cnt);
nctx->w_tax_ind = TRUE;
nctx->w_tax_len = 0;
nctx->d_tax_ind = TRUE;
nctx->d_tax_len = 0;
nctx->o_id_ind = TRUE;
nctx->o_id_len = sizeof(newP->o_id);
nctx->c_discount_ind = TRUE;
nctx->c_discount_len = 0;
nctx->c_credit_ind = TRUE;
nctx->c_credit_len = 0;
nctx->c_last_ind = TRUE;
nctx->c_last_len = 0;
nctx->retries_ind = TRUE;
nctx->retries_len = sizeof(newP->retries);
nctx->cr_date_ind = TRUE;
nctx->cr_date_len = sizeof(newP->cr_date);
#endif PLSQLNO
/* this is the row count */
rcount = newP->o_ol_cnt;
nctx->nol_i_count = newP->o_ol_cnt;
nctx->nol_q_count = newP->o_ol_cnt;
nctx->nol_s_count = newP->o_ol_cnt;
nctx->s_remote_count = newP->o_ol_cnt;

nctx->nol_qty_count = 0;
nctx->nol_bg_count = 0;
nctx->nol_item_count = 0;
}

```



```

nctx->nol_name_count = 0;
nctx->nol_am_count = 0;
/* following not relevant */
nctx->s_data_count = newP->o_ol_cnt;
nctx->i_data_count = newP->o_ol_cnt;

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

for (i = newP->o_ol_cnt; i < NITEMS; i++) {
    nctx->nol_i_id_ind[i] = NA;
    nctx->nol_supply_w_id_ind[i] = NA;
    nctx->nol_quantity_ind[i] = NA;
    nctx->nol_amount_ind[i] = NA;
    nctx->ol_w_id_ind[i] = NA;
    nctx->ol_d_id_ind[i] = NA;
    nctx->ol_o_id_ind[i] = NA;
    nctx->ol_number_ind[i] = NA;
    nctx->ol_dist_info_ind[i] = NA;
    nctx->null_date_ind[i] = NA;
    nctx->s_remote_ind[i] = NA;
    nctx->s_data_ind[i] = NA;
    nctx->i_data_ind[i] = NA;
    nctx->s_quant_ind[i] = NA;
    nctx->s_bg_ind[i] = NA;
    nctx->cons_ind[i] = NA;
    nctx->s_rowid_ind[i] = NA;

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

newP->execstatus = OCISmtExecute(tpsvc,nctx->cum1,errhp,1,0,0,
                                OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);

#else
newP->execstatus = OCISmtExecute(tpsvc,nctx->cum1,errhp,1,0,0,OCI_DEFAULT);
#endif

if(newP->execstatus != OCI_SUCCESS) {
    OCITransRollback(tpsvc,errhp,OCI_DEFAULT);
    newP->errcode = OCIERROR(errhp,newP->execstatus);
    if(newP->errcode == NOT_SERIALIZABLE) {
        newP->retries++;
    }
}

```

```

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

#ifdef PLSQLNO
/* did the txn succeed ? */
if (rcount != newP->o_ol_cnt)
{
    newP->status = rcount - newP->o_ol_cnt;
    newP->o_ol_cnt = rcount;
}
#endif

#ifdef DEBUG
err_printf("tkvcn (NO): w_id=%d, d_id=%d, c_id=%d\n",w_id, d_id, c_id);
#endif

#ifdef PLSQLNO
/* initialization for array operations */

for (i = 0; i < o_ol_cnt; i++) {
    nctx->ol_w_id[i] = w_id;
    nctx->ol_d_id[i] = d_id;
    nctx->ol_number[i] = i + 1;
    nctx->null_date_ind[i] = TRUE;
    nctx->nol_i_id_ind[i] = TRUE;
    nctx->nol_supply_w_id_ind[i] = TRUE;
    nctx->nol_quantity_ind[i] = TRUE;
    nctx->nol_amount_ind[i] = TRUE;
    nctx->ol_w_id_ind[i] = TRUE;
    nctx->ol_d_id_ind[i] = TRUE;
    nctx->ol_o_id_ind[i] = TRUE;
    nctx->ol_number_ind[i] = TRUE;
    nctx->ol_dist_info_ind[i] = TRUE;
    nctx->s_remote_ind[i] = TRUE;
    nctx->s_quant_ind[i] = TRUE;
    nctx->cons_ind[i] = TRUE;
    nctx->s_rowid_ind[i] = TRUE;

    nctx->nol_i_id_len[i] = sizeof(int);
    nctx->nol_supply_w_id_len[i] = sizeof(int);
    nctx->nol_quantity_len[i] = sizeof(int);
    nctx->nol_amount_len[i] = sizeof(int);
    nctx->ol_w_id_len[i] = sizeof(int);
    nctx->ol_d_id_len[i] = sizeof(int);
    nctx->ol_o_id_len[i] = sizeof(int);
    nctx->ol_number_len[i] = sizeof(int);
    nctx->ol_dist_info_len[i] = nctx->s_dist_info_len[i];
    nctx->null_date_len[i] = sizeof(OCIDate);
    nctx->s_remote_len[i] = sizeof(int);
    nctx->s_quant_len[i] = sizeof(int);
    nctx->s_rowid_len[i] = sizeof(nctx->s_rowid_ptr[0]);
    nctx->cons_len[i] = sizeof(int);
}

for (i = o_ol_cnt; i < NITEMS; i++) {
    nctx->nol_i_id_ind[i] = NA;
    nctx->nol_supply_w_id_ind[i] = NA;
    nctx->nol_quantity_ind[i] = NA;
    nctx->nol_amount_ind[i] = NA;
    nctx->ol_w_id_ind[i] = NA;
    nctx->ol_d_id_ind[i] = NA;
    nctx->ol_o_id_ind[i] = NA;
    nctx->ol_number_ind[i] = NA;
    nctx->ol_dist_info_ind[i] = NA;
    nctx->null_date_ind[i] = NA;
    nctx->s_remote_ind[i] = NA;
    nctx->s_quant_ind[i] = NA;
    nctx->cons_ind[i] = NA;
    nctx->s_rowid_ind[i] = NA;

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

rpc3 = SellItemStk (nctx, newP, tpsvc, errhp);
if (rpc3 == -2)
    goto retry;
else if (rpc3 == -1)
    return (-1);
}

```

```

/* compute order line amounts, total amount and stock quantities */

total_amount = 0.0;
for (i = 0; i < newP->o_ol_cnt; i++)
{
    nctx->ol_o_id[i] = newP->o_id;
    if (nctx->nol_i_id_ind[i] != NA) {
        newP->s_quantity[i] = newP->nol_quantity[i];
        if (newP->s_quantity[i] < 10)
            newP->s_quantity[i] += 91;
        newP->nol_amount[i] = (newP->nol_quantity[i] * newP->i_price[i]);
        newP->total_amount += newP->nol_amount[i];
        if (strstr(nctx->i_data[i], "ORIGINAL") &&
            strstr(nctx->s_data[i], "ORIGINAL"))
            newP->brand_gen[i] = 'B';
        else
            newP->brand_gen[i] = 'G';
    }
}

total_amount *= ((float)(10000 - c_discount)/10000) * (1.0 + ((float)(d_tax)/10000) +
(float)(w_tax)/10000);
newP->total_amount = newP->total_amount/100;

rpc = UpdStk2 (nctx, newP, tpsvc, errhp);
if (rpc == -2)
    goto retry;
else if (rpc == -1)
    return (-1);

/* error processing - will keep it separated for readability */
/* number of items selected != number of stock updated */

if (rpc3 != rpc) {
    userlog ("Error in TPC-C server %d: %d rows of item read, ",
        newP->proc_no, rpc3);
    userlog ("          but %d rows of stock updated\n", rpc);
    /* rollback */
    OCITransRollback(tpsvc, errhp, OCI_DEFAULT);
    return (-1);
}

/* common code for insert into order_line */
for (i=0; i< newP->o_ol_cnt; i++) /* move district info in place */
{
    nctx->ol_dist_info_len[i]=nctx->s_dist_info_len[i];
}

/* array insert into order_line table */
flags= (newP->status ? OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
if ((newP->o_ol_cnt - newP->status) > 0)
{
    newP->execstatus = OCISmtExecute(tpsvc, nctx->cum4, errhp, newP->o_ol_cnt - newP->status,
        0, 0, 0, flags);
    if (newP->execstatus != OCI_SUCCESS) {
        OCITransRollback(tpsvc, errhp, OCI_DEFAULT);
        newP->errcode = OCIERROR(errhp, execstatus);
        if (newP->errcode == NOT_SERIALIZABLE) {
            newP->retries++;
            goto retry;
        } else if (newP->errcode == RECOVER) {
            newP->retries++;
            goto retry;
        } else {
            return -1;
        }
    }
    OCIAttrGet(nctx->cum4, OCI_HTYPE_STMT, &rcount, NULL,
        OCI_ATTR_ROW_COUNT, errhp);
    if (rcount != (newP->o_ol_cnt - newP->status))
    {
        userlog ("Error in TPC-C server %d: array insert failed\n",
            newP->proc_no);
        /* rollback */
        OCITransRollback(tpsvc, errhp, OCI_DEFAULT);
        return (-1);
    }
}

/* commit if no invalid item */

if (newP->status) {
    OCITransRollback(tpsvc, errhp, OCI_DEFAULT);
    fflush(stdout);
}

#else
newP->total_amount = 0.0;
for (i = 0; i < newP->o_ol_cnt; i++)
{
    if (nctx->nol_i_id_ind[i] != NA) {
        newP->total_amount += newP->nol_amount[i];
    }
}

newP->total_amount *= ((float)(10000 - newP->c_discount)/10000) * (1.0 + ((float)(newP->d_tax)/
10000) + ((float)(newP->w_tax)/10000));
newP->total_amount = newP->total_amount/100;
#endif

```

```

return (0);
}

void tkvendone (ora_cn_data_t *ora_SlotDataP)
{
    int i;
    newctx *nctx = (newctx *)ora_SlotDataP->nctx;
    global_newOrder_t *newP = ora_SlotDataP->globals;

    if (nctx)
    {
        OCIHandleFree((dvoid *)nctx->cum1, OCI_HTYPE_STMT);
        OCIHandleFree((dvoid *)nctx->cum2, OCI_HTYPE_STMT);
        for (i = 0; i < 10; i++)
            OCIHandleFree((dvoid *)nctx->cum3[i], OCI_HTYPE_STMT);
        OCIHandleFree((dvoid *)nctx->cum4, OCI_HTYPE_STMT);
        free (nctx);
    }
    if (newP) {
        err_printf("free_handles> newP: 0x%x\n", newP);
        free(newP);
        ora_SlotDataP->globals = NULL;
    }
}

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

shiftitemstock (i, j, nctx, newP)

int i, j;
newctx *nctx;
global_newOrder_t *newP;
{
    /* shift up the values for the stock table */
    nctx->s_remote[i] = nctx->s_remote[j];

    /* shift up the order_line values */

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

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

    nctx->nol_supply_w_id_ind[i] = nctx->nol_supply_w_id_ind[j];
    newP->nol_supply_w_id[i] = newP->nol_supply_w_id[j];
}

#if 0
/* TODO - this routine is not ever called. So, no changes for now */

swapitemstock (i, j)

int i, j;

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

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

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

    tempsb2 = nctx->i_price_ind[i];

```

<pre> nctx-&gt;i_price_ind[i] = nctx-&gt;i_price_ind[j]; nctx-&gt;i_price_ind[j] = tempb2; tempub2 = nctx-&gt;i_price_len[i]; nctx-&gt;i_price_len[i] = nctx-&gt;i_price_len[j]; nctx-&gt;i_price_len[j] = tempub2; tempub2 = nctx-&gt;i_price_rcode[i]; nctx-&gt;i_price_rcode[i] = nctx-&gt;i_price_rcode[j]; nctx-&gt;i_price_rcode[j] = tempub2; tempf = i_price[i]; i_price[i] = i_price[j]; i_price[j] = tempf;  tempb2 = nctx-&gt;i_name_ind[i]; nctx-&gt;i_name_ind[i] = nctx-&gt;i_name_ind[j]; nctx-&gt;i_name_ind[j] = tempb2; tempub2 = nctx-&gt;i_name_len[i]; nctx-&gt;i_name_len[i] = nctx-&gt;i_name_len[j]; nctx-&gt;i_name_len[j] = tempub2; tempub2 = nctx-&gt;i_name_rcode[i]; nctx-&gt;i_name_rcode[i] = nctx-&gt;i_name_rcode[j]; nctx-&gt;i_name_rcode[j] = tempub2; strncpy (tempstr, i_name[i], 25); strncpy (i_name[i], i_name[j], 25); strncpy (i_name[j], tempstr, 25);  tempb2 = nctx-&gt;i_data_ind[i]; nctx-&gt;i_data_ind[i] = nctx-&gt;i_data_ind[j]; nctx-&gt;i_data_ind[j] = tempb2; tempub2 = nctx-&gt;i_data_len[i]; nctx-&gt;i_data_len[i] = nctx-&gt;i_data_len[j]; nctx-&gt;i_data_len[j] = tempub2; tempub2 = nctx-&gt;i_data_rcode[i]; nctx-&gt;i_data_rcode[i] = nctx-&gt;i_data_rcode[j]; nctx-&gt;i_data_rcode[j] = tempub2; strncpy (tempstr, nctx-&gt;i_data[i], 51); strncpy (nctx-&gt;i_data[i], nctx-&gt;i_data[j], 51); strncpy (nctx-&gt;i_data[j], tempstr, 51);  tempb2 = nctx-&gt;s_quantity_ind[i]; nctx-&gt;s_quantity_ind[i] = nctx-&gt;s_quantity_ind[j]; nctx-&gt;s_quantity_ind[j] = tempb2; tempub2 = nctx-&gt;s_quantity_len[i]; nctx-&gt;s_quantity_len[i] = nctx-&gt;s_quantity_len[j]; nctx-&gt;s_quantity_len[j] = tempub2; tempub2 = nctx-&gt;s_quantity_rcode[i]; nctx-&gt;s_quantity_rcode[i] = nctx-&gt;s_quantity_rcode[j]; nctx-&gt;s_quantity_rcode[j] = tempub2; tempi = s_quantity[i]; s_quantity[i] = s_quantity[j]; s_quantity[j] = tempi;  tempb2 = nctx-&gt;s_dist_info_ind[i]; nctx-&gt;s_dist_info_ind[i] = nctx-&gt;s_dist_info_ind[j]; nctx-&gt;s_dist_info_ind[j] = tempb2; tempub2 = nctx-&gt;s_dist_info_len[i]; nctx-&gt;s_dist_info_len[i] = nctx-&gt;s_dist_info_len[j]; nctx-&gt;s_dist_info_len[j] = tempub2; tempub2 = nctx-&gt;s_dist_info_rcode[i]; nctx-&gt;s_dist_info_rcode[i] = nctx-&gt;s_dist_info_rcode[j]; nctx-&gt;s_dist_info_rcode[j] = tempub2; strncpy (tempstr, nctx-&gt;s_dist_info[i], 25); strncpy (nctx-&gt;s_dist_info[i], nctx-&gt;s_dist_info[j], 25); strncpy (nctx-&gt;s_dist_info[j], tempstr, 25);  tempb2 = nctx-&gt;s_data_ind[i]; nctx-&gt;s_data_ind[i] = nctx-&gt;s_data_ind[j]; nctx-&gt;s_data_ind[j] = tempb2; tempub2 = nctx-&gt;s_data_len[i]; nctx-&gt;s_data_len[i] = nctx-&gt;s_data_len[j]; nctx-&gt;s_data_len[j] = tempub2; tempub2 = nctx-&gt;s_data_rcode[i]; nctx-&gt;s_data_rcode[i] = nctx-&gt;s_data_rcode[j]; nctx-&gt;s_data_rcode[j] = tempub2; strncpy (tempstr, nctx-&gt;s_data[i], 51); strncpy (nctx-&gt;s_data[i], nctx-&gt;s_data[j], 51); strncpy (nctx-&gt;s_data[j], tempstr, 51); } #endif  SellItemStk (nctx, newP, tpsvc, errhp) newctx *nctx; global newOrder_t *newP; OCISvcCtx *tpsvc; OCIError *errhp; { int i, j, rpc3, rcount;  /* array select from item and stock tables */ newP-&gt;execstatus = OCISmtExecute(tpsvc, (nctx-&gt;cur3)[newP-&gt;d_id-1], errhp, newP-&gt;o_ol_cnt, 0, 0, OCI_DEFAULT); if (newP-&gt;execstatus != OCI_SUCCESS) { newP-&gt;errcode = OCIErrror(errhp, newP-&gt;execstatus); if (newP-&gt;errcode == NOT_SERIALIZABLE) { newP-&gt;retries++; OCITransRollback(tpsvc, errhp, OCI_DEFAULT); return (-2); } } } </pre>	<pre> } else if (newP-&gt;errcode == RECOVER) { /* In case of NO_DATA this should NOT return, but simply fall through */ OCITransRollback(tpsvc, errhp, OCI_DEFAULT); newP-&gt;retries++; return (-2); } else { OCITransRollback(tpsvc, errhp, OCI_DEFAULT); return (-1); } } /* mark invalid items */ OCIAttrGet((nctx-&gt;cur3)[newP-&gt;d_id-1], OCI_HTYPE_STMT, &amp;rcount, NULL, OCI_ATTR_ROW_COUNT, errhp); rpc3 = rcount;  /* the result is in order, so we have to shift up to fill */ /* the slot for the line with the invalid item. */ /* If more than one item is wrong, this is not an simulated */ /* error and we'll blow off */  if ((newP-&gt;status = newP-&gt;o_ol_cnt - rcount) &gt; 1) { userlog ("TPC-C server %d: more than 1 invalid item?"n", proc_no); return (rpc3); } if (newP-&gt;status == 0) return (rpc3);  /* find the invalid item, transfer the rowid information */  for (i = 0; i &lt; newP-&gt;o_ol_cnt; i++) { if (nctx-&gt;cons[i] != i) break; /* this item is invalid */ }  userlog ("TPC-C server %d: reordering items and stocks"n", proc_no);  /* not the last item - shift up */  for (j = i; j &lt; newP-&gt;o_ol_cnt-1; j++) { shifitemstock (j, j+1, nctx, newP); } /* zero the last item */ i = newP-&gt;o_ol_cnt-1; nctx-&gt;nol_id_ind[i] = NA; nctx-&gt;nol_supply_w_id_ind[i] = NA; nctx-&gt;nol_quantity_ind[i] = NA; nctx-&gt;nol_amount_ind[i] = NA; nctx-&gt;ol_w_id_ind[i] = NA; nctx-&gt;ol_d_id_ind[i] = NA; nctx-&gt;ol_o_id_ind[i] = NA; nctx-&gt;null_date_ind[i] = NA; nctx-&gt;ol_number_ind[i] = NA; nctx-&gt;ol_dist_info_ind[i] = NA; nctx-&gt;s_remote_ind[i] = NA; nctx-&gt;s_quant_ind[i] = NA;  nctx-&gt;nol_i_id_len[i] = 0; nctx-&gt;nol_supply_w_id_len[i] = 0; nctx-&gt;nol_quantity_len[i] = 0; nctx-&gt;nol_amount_len[i] = 0; nctx-&gt;ol_w_id_len[i] = 0; nctx-&gt;ol_d_id_len[i] = 0; nctx-&gt;ol_o_id_len[i] = 0; nctx-&gt;ol_number_len[i] = 0; nctx-&gt;ol_dist_info_len[i] = 0; nctx-&gt;null_date_ind[i] = 0; nctx-&gt;s_remote_len[i] = 0; nctx-&gt;s_quant_len[i] = 0;  return (rpc3); }  UpdStk2 (nctx, newP, tpsvc, errhp) newctx *nctx; global newOrder_t *newP; OCISvcCtx *tpsvc; OCIError *errhp; { int rpc, rowoff, iters, rcount;  /* array update of stock table */  newP-&gt;execstatus = OCISmtExecute(tpsvc, nctx-&gt;cur2, errhp, newP-&gt;o_ol_cnt-newP-&gt;status, 0, 0, OCI_DEFAULT); if (newP-&gt;execstatus != OCI_SUCCESS) { OCITransRollback(tpsvc, errhp, OCI_DEFAULT); newP-&gt;errcode = OCIErrror(errhp, newP-&gt;execstatus); if (newP-&gt;errcode == NOT_SERIALIZABLE) { newP-&gt;retries++; return (-2); } } else if (newP-&gt;errcode == RECOVER) { newP-&gt;retries++; return (-2); } } </pre>
--	--

```

} else {
return -1;
}
}
OCIAttrGet(nctx->cum2,OCI_HTYPE_STMT,&rcount,NULL, OCI_ATTR_ROW_COUNT, errhp);
rpc = rcount;

if (rpc != (newP->o_ol_cnt - newP->status)) {
userlog ("Error in TPC-C server %d: array update failed\n",
newP->proc_no);
OCITransRollback(tpcsvc,errhp,OCI_DEFAULT);
return (-1);
}

return (rpc);
}

```

## plora.h

```

#ifndef TPCC_PLORA_H
#define TPCC_PLORA_H

#include "tpcc.h"
#include "tpcc_info.h"

#define NEWO_TRANS (1)
#define PAYMENT_TRANS (2)
#define ORDER_STAT_TRANS (3)
#define DELIVERY_TRANS (4)
#define STOCK_TRANS (5)
#define MAX_TRAN_TYPE (5)

/* struct to copy-in/out neworder vars */
struct global_newOrder_t {
int w_id;
int d_id;
int c_id;
int nol_i_id[15];
int nol_supply_w_id[15];
int nol_quantity[15];
int retries;
ub4 datelen;
text o_entry_d[20];
int o_id;
int o_ol_cnt;
char c_last[17];
char c_credit[3];
float c_discount;
float w_tax;
float d_tax;
float total_amount;
char i_name[15][25];
int s_quantity[15];
char brand_gen[15];
float i_price[15];
int nol_amount[15];
int status;
int o_all_local;
int errcode;
int execstatus;
int proc_no;
char brand_generic[15][1];
int tracelevel;
OCIDate cr_date;
OCIDate c_since;
OCIDate o_entry_d_base;
OCIDate ol_d_base[15];
};

typedef struct global_newOrder_t global_newOrder_t;

struct global_payment_t {
int w_id;
int d_id;
int c_id;
char c_last[17];
char c_first[17];
char c_middle[3];
double c_balance;
int retries;
int bylastname;
OCIDate c_since;
int execstatus;
int errcode;
int c_w_id;
int c_d_id;
int h_amount;
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];
OCIDate cr_date;
};

typedef struct global_payment_t global_payment_t;

struct global_order_t {
OCIDate ol_d_base[15];
int w_id;
int d_id;
int c_id;
char c_last[17];
char c_first[17];
char c_middle[3];
double c_balance;
int o_id;
int o_carrier_id;
int o_ol_cnt;
int ol_supply_w_id[15];
int ol_i_id[15];
unsigned char o_entry_d_base[7];
int ol_quantity[15];
ub4 ol_del_len[15];
text ol_delivery_d[15][11];
int ol_amount[15];
int errcode;
int execstatus;
int retries;
int bylastname;
char o_entry_d[20];
};

typedef struct global_order_t global_order_t;

struct global_delivery_t {
int w_id;
int o_carrier_id;
int retries;
int del_o_id[10];
int errcode;
int execstatus;
int proc_no;
OCIDate cr_date;
};

typedef struct global_delivery_t global_delivery_t;

struct global_stock_t {
int w_id;
int d_id;
int threshold;
int retries;
int low_stock;
int errcode;
int execstatus;
};

typedef struct global_stock_t global_stock_t;

/* Oracle handles and rest of thread specific vars(thread slot data) */

struct ora_cn_data_t {
OCIEnv *tpcenv;
OCIServer *tpcsrv;
OCIError *errhp;
OCISvcCtx *tpcsvc;
OCISession *tpcusr;
OCISmt *curi;
dvoid *xmem;

global_newOrder_t *globals;
global_payment_t *payP;
global_order_t *ordP;
global_delivery_t *delP;
global_stock_t *stoP;
void *nctx;
void *pctx;
void *octx;
void *setx;
void *dctx;
void *actx; /* for #ifdef DMLRETDL */
void *cbctx; /* for orderstatus */
void *ctxp_octx; /* for orderstatus */
};

typedef struct ora_cn_data_t ora_cn_data_t;

```

```
#endif /* TPCC_PLORA_H */
```

## plord.c

```
#ifdef RCSID
```

```
static char *RCSid =
```

```
"$Header: /afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/server/ora8.1_mt/RCS/plord.c,v  
1.2 1999/04/15 12:16:51 oz Exp $ Copyr (c) 1994 Oracle";
```

```
#endif /* RCSID */
```

```
=====  
| Copyright (c) 1995 Oracle Corp. Redwood Shores, CA |  
| OPEN SYSTEMS PERFORMANCE GROUP |  
| All Rights Reserved |  
=====  
| FILENAME  
| plord.c  
| DESCRIPTION  
| OCI version (using PL/SQL anonymous block) of  
| ORDER STATUS transaction in TPC-C benchmark.  
=====*
```

```
#include "tpcc.h"
```

```
#include "plora.h" /* */
```

```
#include "tpccflags.h"
```

```
#ifdef PLSQLORD
```

```
#define SQLTXT "BEGIN orderstatus.getstatus (:w_id, :d_id, :c_id, :byln, \  
:c_last, :c_first, :c_middle, :c_balance, :o_id, :o_entry_d, :o_cr_id, \  
:o_ol_cnt, :ol_s_w_id, :ol_i_id, :ol_quantity, :ol_amount, :ol_d_id); END;"
```

```
#else
```

```
#define SQLCUR0 "SELECT rowid FROM customer \  
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 c_id, c_balance, c_first, c_middle, c_last, \  
o_id, o_entry_d, o_carrier_id, o_ol_cnt \  
FROM customer, orders \  
WHERE customer.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, o_d_id, o_w_id, o_id DESC"
```

```
#define SQLCUR2 "SELECT c_balance, c_first, c_middle, c_last, \  
o_id, o_entry_d, o_carrier_id, o_ol_cnt \  
FROM customer, orders \  
WHERE c_id = :c_id AND c_d_id = :d_id AND c_w_id = :w_id \  
AND o_d_id = c_d_id AND o_w_id = c_w_id AND o_c_id = c_id \  
ORDER BY o_c_id, o_d_id, o_w_id, o_id DESC"
```

```
#define SQLCUR3 "SELECT ol_i_id, ol_supply_w_id, ol_quantity, ol_amount, \  
ol_delivery_d \  
FROM order_line \  
WHERE ol_d_id = :d_id AND ol_w_id = :w_id AND ol_o_id = :o_id"
```

```
#define SQLCUR4 "SELECT count(c_last) FROM customer \  
WHERE c_d_id = :d_id AND c_w_id = :w_id AND c_last = :c_last"
```

```
#endif
```

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

```
ub4 c_rowid_len[100];  
ub2 ol_supply_w_id_len[NITEMS];  
ub2 ol_i_id_len[NITEMS];  
ub2 ol_quantity_len[NITEMS];  
ub2 ol_amount_len[NITEMS];  
ub2 ol_delivery_d_len[NITEMS];  
ub2 ol_w_id_len;  
ub2 ol_d_id_len;  
ub2 ol_o_id_len;
```

```
ub2 c_rowid_rcode[100];  
ub2 ol_supply_w_id_rcode[NITEMS];  
ub2 ol_i_id_rcode[NITEMS];
```

```
ub2 ol_quantity_rcode[NITEMS];  
ub2 ol_amount_rcode[NITEMS];  
ub2 ol_delivery_d_rcode[NITEMS];  
ub2 ol_w_id_rcode;  
ub2 ol_d_id_rcode;  
ub2 ol_o_id_rcode;
```

```
ub4 ol_supply_w_id_csize;  
ub4 ol_i_id_csize;  
ub4 ol_quantity_csize;  
ub4 ol_amount_csize;  
ub4 ol_delivery_d_csize;  
ub4 ol_w_id_csize;  
ub4 ol_d_id_csize;  
ub4 ol_o_id_csize;
```

```
OCIStmt *curo0;  
OCIBind *w_id_bp0;  
OCIBind *d_id_bp0;  
OCIBind *c_id_bp;  
OCIBind *c_last_bp;  
#ifdef PLSQLORD  
OCIBind *byln_bp;  
OCIBind *c_first_bp;  
OCIBind *c_middle_bp;  
OCIBind *c_balance_bp;  
OCIBind *o_entry_d_bp;  
OCIBind *o_cr_id_bp;  
OCIBind *o_ol_cnt_bp;  
OCIBind *ol_i_id_bp;  
OCIBind *ol_supply_w_id_bp;  
OCIBind *ol_quantity_bp;  
OCIBind *ol_amount_bp;  
OCIBind *ol_d_base_bp;  
ub4 ol_i_id_cnt;  
ub4 ol_sup_cnt;  
ub4 ol_qty_cnt;  
ub4 ol_amt_cnt;  
ub4 ol_del_d_cnt;
```

```
#else
```

```
OCIStmt *curo1;  
OCIStmt *curo2;  
OCIStmt *curo3;  
OCIStmt *curo4;  
OCIBind *w_id_bp2;  
OCIBind *w_id_bp3;  
OCIBind *w_id_bp4;  
OCIBind *d_id_bp2;  
OCIBind *d_id_bp3;  
OCIBind *d_id_bp4;  
OCIBind *c_last_bp4;  
OCIBind *o_id_bp;  
OCIBind *c_rowid_bp;  
OCIDefine *c_rowid_dp;  
OCIDefine *c_last_dp;  
OCIDefine *c_last_dp1;  
OCIDefine *c_id_dp;  
OCIDefine *c_first_dp1;  
OCIDefine *c_first_dp2;  
OCIDefine *c_middle_dp1;  
OCIDefine *c_middle_dp2;  
OCIDefine *c_balance_dp1;  
OCIDefine *c_balance_dp2;  
OCIDefine *o_id_dp1;  
OCIDefine *o_id_dp2;  
OCIDefine *o_entry_d_dp1;  
OCIDefine *o_entry_d_dp2;  
OCIDefine *o_cr_id_dp1;  
OCIDefine *o_cr_id_dp2;  
OCIDefine *o_ol_cnt_dp1;  
OCIDefine *o_ol_cnt_dp2;  
OCIDefine *ol_d_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];  
int cs;  
int cust_idx;  
int norow;  
int rcount;  
int somerows;
```

```
#endif  
};
```

```
typedef struct ordctx ordctx;
```

```
struct defctx  
{  
boolean reexec;  
ub4 count;  
};  
typedef struct defctx defctx;
```

```
struct defctx_ordctx {  
defctx *ctxp;  
ordctx *octx;
```

<pre> }; typedef struct defctx_ordctx defctx_ordctx;  /* ordctx *octx; */  /* defctx cbctx; */  #ifdef PLSQLORD sb4 rid_data(dvoid *ctxp_octx, OCIDefine *dp, ub4 iter,              dvoid **bufpp, ub4 **alenp, ub1 *piecep,              dvoid **indpp, ub2 **rcodepp) {     ub4 i;     defctx *ctxp = ((defctx_ordctx *)ctxp_octx)-&gt;ctxp;     ordctx *octx = ((defctx_ordctx *)ctxp_octx)-&gt;octx;     if (((defctx *)ctxp)-&gt;reexec)/* if this is the second execute - use entry 0 */     {         i = 0;         ((defctx *)ctxp)-&gt;count--; /* count down */     }     else         i = iter;     *bufpp = octx-&gt;c_rowid_ptr[i];     *indpp = &amp;octx-&gt;c_rowid_ind[i];     *alenp = &amp;octx-&gt;c_rowid_len[i];     *rcodepp = &amp;octx-&gt;c_rowid_rcode[i];     *piecep = OCI_ONE_PIECE;     return (OCI_CONTINUE); } #endif  kvcoinit (ora_cn_data_t *ora_SlotDataP) {     int i;     text stmbuf[SQL_BUF_SIZE];      ordctx *octx;     defctx *cbctx;     global_order_t *ordP;     OCIEnv *tpcenv = ora_SlotDataP-&gt;tpcenv;     OCIError *errhp = ora_SlotDataP-&gt;errhp;     OCISvcCtx *tpscvc = ora_SlotDataP-&gt;tpscvc;     OCISession *tpcsur = ora_SlotDataP-&gt;tpcsur;     OCIStmt *curi = ora_SlotDataP-&gt;curi;     defctx_ordctx *ctxp_octx;      octx = (ordctx *) malloc (sizeof(ordctx));     memset(octx,(char)0,sizeof(ordctx));     ora_SlotDataP-&gt;octx = (void *)octx; /* */      cbctx = (defctx *) malloc (sizeof(defctx));     memset(cbctx,(char)0,sizeof(defctx));     ora_SlotDataP-&gt;cbctx = (void *)cbctx; /* */      /* allocate the space */     ctxp_octx = (defctx_ordctx *)malloc(sizeof(defctx_ordctx));     ora_SlotDataP-&gt;ctxp_octx = (void *)ctxp_octx;      ora_SlotDataP-&gt;ordP = (global_order_t *)malloc(sizeof(global_order_t));     memset(ora_SlotDataP-&gt;ordP,(char)0,sizeof(global_order_t));     ordP = ora_SlotDataP-&gt;ordP;  #ifdef PLSQLORD     octx-&gt;cs = 1;     octx-&gt;norow = 0;     octx-&gt;somerows = 10;     /* get the rowid handles */     for(i=0;i&lt;100;i++) {         OCIERROR(errhp, OCIDescriptorAlloc(tpcenv,(dvoid*)&amp;octx-&gt;c_rowid_ptr[i],         OCI_DTYPE_ROWID,0,(dvoid**)0));     } #endif      OCIERROR(errhp,              OCIHandleAlloc(tpcenv,(dvoid*)&amp;octx-&gt;curo0,OCI_HTYPE_STMT,0,(dvoid**)0);     OCIERROR(errhp,              OCIHandleAlloc(tpcenv,(dvoid*)&amp;octx-&gt;curo0,OCI_HTYPE_STMT,0,(dvoid**)0); #ifdef PLSQLORD     OCIERROR(errhp,              OCIHandleAlloc(tpcenv,(dvoid*)&amp;octx-&gt;curo1,OCI_HTYPE_STMT,0,(dvoid**)0);     OCIERROR(errhp,              OCIHandleAlloc(tpcenv,(dvoid*)&amp;octx-&gt;curo2,OCI_HTYPE_STMT,0,(dvoid**)0);     OCIERROR(errhp,              OCIHandleAlloc(tpcenv,(dvoid*)&amp;octx-&gt;curo3,OCI_HTYPE_STMT,0,(dvoid**)0);     OCIERROR(errhp,              OCIHandleAlloc(tpcenv,(dvoid*)&amp;octx-&gt;curo4,OCI_HTYPE_STMT,0,(dvoid**)0); #endif  #ifdef PLSQLORD     sprintf((char *) stmbuf, SQLTXT);     OCIERROR(errhp,              OCIStmtPrepare(octx-&gt;curo0,errhp,stmbuf,strlen((char *)stmbuf),              OCI_NTV_SYNTAX,OCI_DEFAULT)); #endif  /* c_id = 0, use find customer by lastname. Get an array or rowid's back*/ </pre>	<pre> sprintf((char *) stmbuf, SQLCUR0); OCIERROR(errhp,          OCIStmtPrepare(octx-&gt;curo0,errhp,stmbuf,strlen((char *)stmbuf),          OCI_NTV_SYNTAX,OCI_DEFAULT));  OCIERROR(errhp,          OCIAttrSet(octx-&gt;curo0,OCI_HTYPE_STMT,(dvoid*)&amp;octx-&gt;norow,0,          OCI_ATTR_PREFETCH_ROWS,errhp)); /* get order/customer info back based on rowid */ sprintf((char *) stmbuf, SQLCUR1); OCIERROR(errhp,          OCIStmtPrepare(octx-&gt;curo1,errhp,stmbuf,strlen((char *)stmbuf),          OCI_NTV_SYNTAX,OCI_DEFAULT));  OCIERROR(errhp,          OCIAttrSet(octx-&gt;curo1,OCI_HTYPE_STMT,(dvoid*)&amp;octx-&gt;norow,0,          OCI_ATTR_PREFETCH_ROWS,errhp));  /* c_id == 0, use lastname to find customer */ sprintf((char *) stmbuf, SQLCUR2); OCIERROR(errhp,          OCIStmtPrepare(octx-&gt;curo2,errhp,stmbuf,strlen((char *)stmbuf),          OCI_NTV_SYNTAX,OCI_DEFAULT));  OCIERROR(errhp,          OCIAttrSet(octx-&gt;curo2,OCI_HTYPE_STMT,(dvoid*)&amp;octx-&gt;norow,0,          OCI_ATTR_PREFETCH_ROWS,errhp));  sprintf((char *) stmbuf, SQLCUR3); OCIERROR(errhp,          OCIStmtPrepare(octx-&gt;curo3,errhp,stmbuf,strlen((char *)stmbuf),          OCI_NTV_SYNTAX,OCI_DEFAULT));  OCIERROR(errhp,          OCIAttrSet(octx-&gt;curo3,OCI_HTYPE_STMT,(dvoid*)&amp;octx-&gt;norow,0,          OCI_ATTR_PREFETCH_ROWS,errhp));  sprintf((char *) stmbuf, SQLCUR4); OCIERROR(errhp,          OCIStmtPrepare(octx-&gt;curo4,errhp,stmbuf,strlen((char *)stmbuf),          OCI_NTV_SYNTAX,OCI_DEFAULT));  OCIERROR(errhp,          OCIAttrSet(octx-&gt;curo4,OCI_HTYPE_STMT,(dvoid*)&amp;octx-&gt;norow,0,          OCI_ATTR_PREFETCH_ROWS,errhp)); #endif  for (i = 0; i &lt; NITEMS; i++) {     octx-&gt;ol_supply_w_id_ind[i] = TRUE;     octx-&gt;ol_i_id_ind[i] = TRUE;     octx-&gt;ol_quantity_ind[i] = TRUE;     octx-&gt;ol_amount_ind[i] = TRUE;     octx-&gt;ol_delivery_d_ind[i] = TRUE;      octx-&gt;ol_supply_w_id_len[i] = sizeof(int);     octx-&gt;ol_i_id_len[i] = sizeof(int);     octx-&gt;ol_quantity_len[i] = sizeof(int);     octx-&gt;ol_amount_len[i] = sizeof(int);     octx-&gt;ol_delivery_d_len[i] = sizeof(ordP-&gt;ol_d_base[0]); }  octx-&gt;ol_supply_w_id_csize = NITEMS; octx-&gt;ol_i_id_csize = NITEMS; octx-&gt;ol_quantity_csize = NITEMS; octx-&gt;ol_amount_csize = NITEMS; octx-&gt;ol_delivery_d_csize = NITEMS; octx-&gt;ol_w_id_csize = NITEMS; octx-&gt;ol_o_id_csize = NITEMS; octx-&gt;ol_d_id_csize = NITEMS; octx-&gt;ol_w_id_ind = TRUE; octx-&gt;ol_d_id_ind = TRUE; octx-&gt;ol_o_id_ind = TRUE; octx-&gt;ol_w_id_len = sizeof(int); octx-&gt;ol_d_id_len = sizeof(int); octx-&gt;ol_o_id_len = sizeof(int);  /* bind variables */ #ifdef PLSQLORD     OCIBND(octx-&gt;curo0, octx-&gt;w_id_bp0, errhp,":w_id",ADR(ordP-&gt;w_id),     SIZ(int),SQLT_INT);     OCIBND(octx-&gt;curo0, octx-&gt;d_id_bp0, errhp,":d_id",ADR(ordP-&gt;d_id),     SIZ(int), SQLT_INT);     OCIBND(octx-&gt;curo0, octx-&gt;c_id_bp , errhp,":c_id",ADR(ordP-&gt;c_id),     SIZ(c_id),SQLT_INT);     OCIBND(octx-&gt;curo0, octx-&gt;byln_bp , errhp,":byln",ADR(ordP-&gt;bylastname),     SIZ(int),SQLT_INT);     OCIBND(octx-&gt;curo0, octx-&gt;c_last_bp , errhp,":c_last",ordP-&gt;c_last,     SIZ(ordP-&gt;c_last),SQLT_STR);     OCIBND(octx-&gt;curo0, octx-&gt;c_first_bp , errhp,":c_first",ordP-&gt;c_first,     SIZ(ordP-&gt;c_first),SQLT_STR);     OCIBND(octx-&gt;curo0, octx-&gt;c_middle_bp , errhp,":c_middle",ordP-&gt;c_middle,     SIZ(ordP-&gt;c_middle),SQLT_STR);     OCIBND(octx-&gt;curo0, octx-&gt;c_balance_bp , errhp,":c_balance",     ADR(ordP-&gt;c_balance),SIZ(float),SQLT_FLT);     OCIBND(octx-&gt;curo0, octx-&gt;c_id_bp , errhp,":o_id",ADR(ordP-&gt;o_id),     SIZ(int),SQLT_INT);     OCIBND(octx-&gt;curo0, octx-&gt;o_entry_d_bp , errhp,":o_entry_d",ordP-&gt;o_entry_d,     SIZ(ordP-&gt;o_entry_d),SQLT_STR);     OCIBND(octx-&gt;curo0, octx-&gt;o_cr_id_bp , errhp,":o_cr_id",ADR(ordP-&gt;o_carrier_id),     SIZ(int), SQLT_INT);     OCIBND(octx-&gt;curo0, octx-&gt;o_ol_cnt_bp , errhp,":o_ol_cnt",ADR(ordP-&gt;o_ol_cnt),     SIZ(int),SQLT_INT);      OCIBNDRAA(octx-&gt;curo0, octx-&gt;ol_i_id_bp, errhp, ":ol_i_id", </pre>
--	---

<pre> ordP-&gt;ol_i_id,SIZ(int),SQLT_INT, octx-&gt;ol_i_id_ind,octx-&gt;ol_i_id_len, octx-&gt;ol_i_id_rcode,NITEMS,&amp;octx-&gt;ol_i_id_cnt); OCIBNDRAA(octx-&gt;euro0,octx-&gt;ol_supply_w_id_bp,errhp,"ol_s_w_id", ordP-&gt;ol_supply_w_id,SIZ(int),SQLT_INT, octx-&gt;ol_supply_w_id_ind,octx-&gt;ol_supply_w_id_len, octx-&gt;ol_supply_w_id_rcode,NITEMS,&amp;octx-&gt;ol_sup_cnt); OCIBNDRAA(octx-&gt;euro0, octx-&gt;ol_quantity_bp,errhp,"ol_quantity", ordP-&gt;ol_quantity,SIZ(int),SQLT_INT, octx-&gt;ol_quantity_ind,octx-&gt;ol_quantity_len, octx-&gt;ol_quantity_rcode,NITEMS,&amp;octx-&gt;ol_qty_cnt); OCIBNDRAA(octx-&gt;euro0,octx-&gt;ol_amount_bp,errhp,"ol_amount",ordP-&gt;ol_amount, SIZ(float),SQLT_FLT,octx-&gt;ol_amount_ind, octx-&gt;ol_amount_len, octx-&gt;ol_amount_rcode,NITEMS, &amp;octx-&gt;ol_amt_cnt); OCIBNDRAA(octx-&gt;euro0,octx-&gt;ol_d_base_bp,errhp,"ol_d_d",ordP-&gt;ol_d_base, SIZ(OCIDate),SQLT_ODT,octx-&gt;ol_delivery_d_ind, octx-&gt;ol_delivery_d_len, octx-&gt;ol_delivery_d_rcode,NITEMS, &amp;octx-&gt;ol_del_d_cnt);  #else  /* c_id (customer id) is not known */ OCIBND(octx-&gt;euro0,octx-&gt;w_id_bp0,errhp,"w_id",ADR(ordP-&gt;w_id),SIZ(int),SQLT_INT); OCIBND(octx-&gt;euro0,octx-&gt;d_id_bp0,errhp,"d_id",ADR(ordP-&gt;d_id),SIZ(int),SQLT_INT); OCIBND(octx-&gt;euro0,octx-&gt;c_last_bp,errhp,"c_last",ordP-&gt;c_last,SIZ(ordP-&gt;c_last), SQLT_STR); ctxp_octx-&gt;ctxp = cbctx; ctxp_octx-&gt;octx = octx; OCIDFNDRYD(octx-&gt;euro0,octx-&gt;c_rowid_dp,errhp,1,octx-&gt;c_rowid_ptr, SIZ(OCIRowid*),SQLT_RDD, octx-&gt;c_rowid_ind, (dvoid *)ctxp_octx, rid_data);  OCIBND(octx-&gt;euro1,octx-&gt;c_rowid_bp,errhp,"cust_rowid", &amp;octx-&gt;c_rowid_ptr[octx-&gt;cust_idx], sizeof(octx-&gt;c_rowid_ptr[0]),SQLT_RDD); OCIDF(octx-&gt;euro1,octx-&gt;c_id_dp,errhp,1,ADR(ordP-&gt;c_id),SIZ(int),SQLT_INT); OCIDF(octx-&gt;euro1,octx-&gt;c_balance_dp1,errhp,2,ADR(ordP-&gt;c_balance), SIZ(double),SQLT_FLT); OCIDF(octx-&gt;euro1,octx-&gt;c_first_dp1,errhp,3,ordP-&gt;c_first,SIZ(ordP-&gt;c_first)-1, SQLT_CHR); OCIDF(octx-&gt;euro1,octx-&gt;c_middle_dp1,errhp,4,ordP-&gt;c_middle, SIZ(ordP-&gt;c_middle)-1,SQLT_AFC); OCIDF(octx-&gt;euro1,octx-&gt;c_last_dp1,errhp,5,ordP-&gt;c_last,SIZ(ordP-&gt;c_last)-1, SQLT_CHR); OCIDF(octx-&gt;euro1,octx-&gt;o_id_dp1,errhp,6,ADR(ordP-&gt;o_id),SIZ(int),SQLT_INT); OCIDF(octx-&gt;euro1,octx-&gt;o_entry_dp1,errhp,7, &amp;ordP-&gt;o_entry_d_base,SIZ(OCIDate),SQLT_ODT); OCIDF(octx-&gt;euro1,octx-&gt;o_cr_id_dp1,errhp,8,ADR(ordP-&gt;o_carrier_id), SIZ(int),SQLT_INT); OCIDF(octx-&gt;euro1,octx-&gt;o_ol_cnt_dp1,errhp,9,ADR(ordP-&gt;o_ol_cnt), SIZ(int),SQLT_INT);  /* Bind for third cursor , no-zero customer id */ OCIBND(octx-&gt;euro2,octx-&gt;w_id_bp2,errhp,"w_id",ADR(ordP-&gt;w_id),SIZ(int),SQLT_INT); OCIBND(octx-&gt;euro2,octx-&gt;d_id_bp2,errhp,"d_id",ADR(ordP-&gt;d_id),SIZ(int),SQLT_INT); OCIBND(octx-&gt;euro2,octx-&gt;c_id_bp,errhp,"c_id",ADR(ordP-&gt;c_id),SIZ(int),SQLT_INT); OCIDF(octx-&gt;euro2,octx-&gt;c_balance_dp2,errhp,1,ADR(ordP-&gt;c_balance), SIZ(double),SQLT_FLT); OCIDF(octx-&gt;euro2,octx-&gt;c_first_dp2,errhp,2,ordP-&gt;c_first,SIZ(ordP-&gt;c_first)-1, SQLT_CHR); OCIDF(octx-&gt;euro2,octx-&gt;c_middle_dp2,errhp,3,ordP-&gt;c_middle, SIZ(ordP-&gt;c_middle)-1,SQLT_AFC); OCIDF(octx-&gt;euro2,octx-&gt;c_last_dp2,errhp,4,ordP-&gt;c_last,SIZ(ordP-&gt;c_last)-1,SQLT_CHR); OCIDF(octx-&gt;euro2,octx-&gt;o_id_dp2,errhp,5,ADR(ordP-&gt;o_id),SIZ(int),SQLT_INT); OCIDF(octx-&gt;euro2,octx-&gt;o_entry_dp2,errhp,6, &amp;ordP-&gt;o_entry_d_base, SIZ(OCIDate),SQLT_ODT); OCIDF(octx-&gt;euro2, octx-&gt;o_cr_id_dp2,errhp,7,ADR(ordP-&gt;o_carrier_id), SIZ(int),SQLT_INT); OCIDF(octx-&gt;euro2,octx-&gt;o_ol_cnt_dp2,errhp,8,ADR(ordP-&gt;o_ol_cnt), SIZ(int),SQLT_INT);  /* Bind for last cursor */ OCIBND(octx-&gt;euro3,octx-&gt;w_id_bp3,errhp,"w_id",ADR(ordP-&gt;w_id),SIZ(int),SQLT_INT); OCIBND(octx-&gt;euro3,octx-&gt;d_id_bp3,errhp,"d_id",ADR(ordP-&gt;d_id),SIZ(int),SQLT_INT); OCIBND(octx-&gt;euro3,octx-&gt;o_id_bp,errhp,"o_id",ADR(ordP-&gt;o_id),SIZ(int),SQLT_INT);  OCIDFNRA(octx-&gt;euro3, octx-&gt;ol_i_id_dp, errhp, 1, ordP-&gt;ol_i_id,SIZ(int),SQLT_INT, octx-&gt;ol_i_id_ind,octx-&gt;ol_i_id_len, octx-&gt;ol_i_id_rcode); OCIDFNRA(octx-&gt;euro3,octx-&gt;ol_supply_w_id_dp,errhp,2, ordP-&gt;ol_supply_w_id, SIZ(int),SQLT_INT, octx-&gt;ol_supply_w_id_ind, octx-&gt;ol_supply_w_id_len, octx-&gt;ol_supply_w_id_rcode); OCIDFNRA(octx-&gt;euro3, octx-&gt;ol_quantity_dp,errhp,3, ordP-&gt;ol_quantity,SIZ(int), SQLT_INT, octx-&gt;ol_quantity_ind,octx-&gt;ol_quantity_len, octx-&gt;ol_quantity_rcode); OCIDFNRA(octx-&gt;euro3,octx-&gt;ol_amount_dp,errhp,4,ordP-&gt;ol_amount, SIZ(int), SQLT_INT,octx-&gt;ol_amount_ind, octx-&gt;ol_amount_len, octx-&gt;ol_amount_rcode); OCIDFNRA(octx-&gt;euro3,octx-&gt;ol_d_base_dp,errhp,5,ordP-&gt;ol_d_base,SIZ(OCIDate), SQLT_ODT, octx-&gt;ol_delivery_d_ind,octx-&gt;ol_delivery_d_len, octx-&gt;ol_delivery_d_rcode);  OCIBND(octx-&gt;euro4,octx-&gt;w_id_bp4,errhp,"w_id",ADR(ordP-&gt;w_id),SIZ(int),SQLT_INT); OCIBND(octx-&gt;euro4,octx-&gt;d_id_bp4,errhp,"d_id",ADR(ordP-&gt;d_id),SIZ(int),SQLT_INT); OCIBND(octx-&gt;euro4,octx-&gt;c_last_bp4,errhp,"c_last",ordP-&gt;c_last,SIZ(ordP-&gt;c_last), SQLT_STR); OCIDF(octx-&gt;euro4,octx-&gt;c_count_dp,errhp,1,ADR(octx-&gt;rcount),SIZ(int), SQLT_INT); </pre>	<pre> #endif return (0); }  tkvco (ora_cn_data_t *ora_SlotDataP) { int i; int rcount;  /* */ ordctx *octx = (ordctx *)ora_SlotDataP-&gt;octx; defctx *cbctx = (defctx *)ora_SlotDataP-&gt;cbctx; global_order_t *ordP = ora_SlotDataP-&gt;ordP; OCIEnv *tpcenv = ora_SlotDataP-&gt;tpcenv; OCIServer *tpcsrv = ora_SlotDataP-&gt;tpcsrv; OCLError *errhp = ora_SlotDataP-&gt;errhp; OCISvcCtx *tpcsvc = ora_SlotDataP-&gt;tpcsvc; OCISession *tpcusr = ora_SlotDataP-&gt;tpcusr; OCISmt *curi = ora_SlotDataP-&gt;curi;  for (i = 0; i &lt; NITEMS; i++) { octx-&gt;ol_supply_w_id_ind[i] = TRUE; octx-&gt;ol_i_id_ind[i] = TRUE; octx-&gt;ol_quantity_ind[i] = TRUE; octx-&gt;ol_amount_ind[i] = TRUE; octx-&gt;ol_delivery_d_ind[i] = TRUE; octx-&gt;ol_supply_w_id_len[i] = sizeof(int); octx-&gt;ol_i_id_len[i] = sizeof(int); octx-&gt;ol_quantity_len[i] = sizeof(int); octx-&gt;ol_amount_len[i] = sizeof(int); octx-&gt;ol_delivery_d_len[i] = sizeof(OCIDate); } octx-&gt;ol_supply_w_id_csize = NITEMS; octx-&gt;ol_i_id_csize = NITEMS; octx-&gt;ol_quantity_csize = NITEMS; octx-&gt;ol_amount_csize = NITEMS; octx-&gt;ol_delivery_d_csize = NITEMS; #endif PLSQLORD octx-&gt;ol_i_id_cnt = 0; octx-&gt;ol_sup_cnt = 0; octx-&gt;ol_qty_cnt = 0; octx-&gt;ol_amt_cnt = 0; octx-&gt;ol_del_d_cnt = 0; OCIERROR(errhp, OCISmtExecute(tpcsvc,octx-&gt;euro0,errhp,1,0,0,0,OCI_DEFAULT));  #else retry: if(ordP-&gt;bylastname) { cbctx-&gt;reexec = FALSE; ordP-&gt;execstatus=OCISmtExecute(tpcsvc,octx-&gt;euro0,errhp,100,0,0,0,OCI_DEFAULT); /* will get OCI_NO_DATA if &lt;100 found */ if ((ordP-&gt;execstatus != OCI_NO_DATA) &amp;&amp; (ordP-&gt;execstatus != OCI_SUCCESS)) { ordP-&gt;errcode=OCIERROR(errhp, ordP-&gt;execstatus); if((ordP-&gt;errcode == NOT_SERIALIZABLE)    (ordP-&gt;errcode == RECOVER)) { OCITransCommit(tpcsvc,errhp,OCI_DEFAULT); ordP-&gt;retries++; goto retry; } else { return -1; } } if (ordP-&gt;execstatus == OCI_NO_DATA) /* there are no more rows */ { /* get rowcount, find middle one */ OCIAttrGet(octx-&gt;euro0,OCI_HTYPE_STMT,&amp;rcount,NULL,OCI_ATTR_ROW_COUNT,errhp); if (rcount &lt; 1) { userlog("ORDERSTATUS rcount=%d\n",rcount); return (-1); } octx-&gt;cust_idx=(rcount+1)/2; } else { /* count the number of rows */ ordP-&gt;execstatus=OCISmtExecute(tpcsvc,octx-&gt;euro4,errhp,1,0,0,0,OCI_DEFAULT); if ((ordP-&gt;execstatus != OCI_NO_DATA) &amp;&amp; (ordP-&gt;execstatus != OCI_SUCCESS)) { if ((ordP-&gt;errcode == NOT_SERIALIZABLE)    (ordP-&gt;errcode == RECOVER)) { OCITransCommit(tpcsvc,errhp,OCI_DEFAULT); ordP-&gt;retries++; goto retry; } else { return -1; } } } if (octx-&gt;rcount+1 &lt; 2*10) octx-&gt;cust_idx=(octx-&gt;rcount+1)/2; else /* */ </pre>
--	---





<pre> ub2 h_amount_len; ub2 h_amount_rc;  OCIBind *c_last_bp; OCIBind *c_last_bp1; sb2 c_last_ind; ub2 c_last_len; ub2 c_last_rc;  OCIBind *w_street_1_bp; OCIBind *w_street_1_bp1; sb2 w_street_1_ind; ub2 w_street_1_len; ub2 w_street_1_rc;  OCIBind *w_street_2_bp; OCIBind *w_street_2_bp1; sb2 w_street_2_ind; ub2 w_street_2_len; ub2 w_street_2_rc;  OCIBind *w_city_bp; OCIBind *w_city_bp1; sb2 w_city_ind; ub2 w_city_len; ub2 w_city_rc;  OCIBind *w_state_bp; OCIBind *w_state_bp1; sb2 w_state_ind; ub2 w_state_len; ub2 w_state_rc;  OCIBind *w_zip_bp; OCIBind *w_zip_bp1; sb2 w_zip_ind; ub2 w_zip_len; ub2 w_zip_rc;  OCIBind *d_street_1_bp; OCIBind *d_street_1_bp1; sb2 d_street_1_ind; ub2 d_street_1_len; ub2 d_street_1_rc;  OCIBind *d_street_2_bp; OCIBind *d_street_2_bp1; sb2 d_street_2_ind; ub2 d_street_2_len; ub2 d_street_2_rc;  OCIBind *d_city_bp; OCIBind *d_city_bp1; sb2 d_city_ind; ub2 d_city_len; ub2 d_city_rc;  OCIBind *d_state_bp; OCIBind *d_state_bp1; sb2 d_state_ind; ub2 d_state_len; ub2 d_state_rc;  OCIBind *d_zip_bp; OCIBind *d_zip_bp1; sb2 d_zip_ind; ub2 d_zip_len; ub2 d_zip_rc;  OCIBind *c_first_bp; OCIBind *c_first_bp1; sb2 c_first_ind; ub2 c_first_len; ub2 c_first_rc;  OCIBind *c_middle_bp; OCIBind *c_middle_bp1; sb2 c_middle_ind; ub2 c_middle_len; ub2 c_middle_rc;  OCIBind *c_street_1_bp; OCIBind *c_street_1_bp1; sb2 c_street_1_ind; ub2 c_street_1_len; ub2 c_street_1_rc;  OCIBind *c_street_2_bp; OCIBind *c_street_2_bp1; sb2 c_street_2_ind; ub2 c_street_2_len; ub2 c_street_2_rc;  OCIBind *c_city_bp; OCIBind *c_city_bp1; sb2 c_city_ind; ub2 c_city_len; ub2 c_city_rc; </pre>	<pre> OCIBind *c_state_bp; OCIBind *c_state_bp1; sb2 c_state_ind; ub2 c_state_len; ub2 c_state_rc;  OCIBind *c_zip_bp; OCIBind *c_zip_bp1; sb2 c_zip_ind; ub2 c_zip_len; ub2 c_zip_rc;  OCIBind *c_phone_bp; OCIBind *c_phone_bp1; sb2 c_phone_ind; ub2 c_phone_len; ub2 c_phone_rc;  OCIBind *c_since_bp; OCIBind *c_since_bp1; sb2 c_since_ind; ub2 c_since_len; ub2 c_since_rc;  OCIBind *c_credit_bp; OCIBind *c_credit_bp1; sb2 c_credit_ind; ub2 c_credit_len; ub2 c_credit_rc;  OCIBind *c_credit_lim_bp; OCIBind *c_credit_lim_bp1; sb2 c_credit_lim_ind; ub2 c_credit_lim_len; ub2 c_credit_lim_rc;  OCIBind *c_discount_bp; OCIBind *c_discount_bp1; sb2 c_discount_ind; ub2 c_discount_len; ub2 c_discount_rc;  OCIBind *c_balance_bp; OCIBind *c_balance_bp1; sb2 c_balance_ind; ub2 c_balance_len; ub2 c_balance_rc;  OCIBind *c_data_bp; OCIBind *c_data_bp1; sb2 c_data_ind; ub2 c_data_len; ub2 c_data_rc;  OCIBind *h_date_bp; OCIBind *h_date_bp1; sb2 h_date_ind; ub2 h_date_len; ub2 h_date_rc;  OCIBind *retries_bp; OCIBind *retries_bp1; sb2 retries_ind; ub2 retries_len; ub2 retries_rc;  OCIBind *cr_date_bp; OCIBind *cr_date_bp1; sb2 cr_date_ind; ub2 cr_date_len; ub2 cr_date_rc;  OCIBind *byln_bp; sb2 byln_ind; ub2 byln_len; ub2 byln_rc; };  typedef struct payctx payctx;  /* payctx *pctx; */  tkvcpinit(ora_cn_data_t *ora_SlotDataP) { /* */ char *ora_home = getenv("ORACLE_HOME"); char sql_file_name[256]; payctx *pctx; global_payment_t *payP; OCIEnv *tpcenv = ora_SlotDataP-&gt;tpcenv; OCIServer *tpcsrv = ora_SlotDataP-&gt;tpcsrv; OCIError *errhp = ora_SlotDataP-&gt;errhp; OCISvcCtx *tpscvc = ora_SlotDataP-&gt;tpscvc; OCISession *tpcusr = ora_SlotDataP-&gt;tpcusr; OCIStmt *curi = ora_SlotDataP-&gt;curi; </pre>
--	--

<pre> text stmbuf[SQL_BUF_SIZE];  if (lora_home) {     err_printf("Cannot find env variable ORACLE_HOME\n");     exit(13); }  pctx = (payctx *)malloc(sizeof(payctx)); memset(pctx,(char)0,sizeof(payctx)); ora_SlotDataP-&gt;pctx = (void *)pctx;  ora_SlotDataP-&gt;payP = (global_payment_t *)malloc(sizeof(global_payment_t)); memset(ora_SlotDataP-&gt;payP,(char)0,sizeof(global_payment_t)); payP = ora_SlotDataP-&gt;payP;  /* cursor for init */ OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)&amp;(pctx-&gt;curpi)),     OCI_HTYPE_STMT,0,(dvoid**)0);  OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)&amp;(pctx-&gt;curp0)),     OCI_HTYPE_STMT,0,(dvoid**)0); OCIERROR(errhp,OCIHandleAlloc(tpcenv, (dvoid **)&amp;(pctx-&gt;curp1)),     OCI_HTYPE_STMT,0,(dvoid**)0);  /* build the init statement and execute it */  sprintf((char*)stmbuf,SQLTXT_INIT); OCIERROR(errhp,OCIStmtPrepare(pctx-&gt;curpi, errhp, stmbuf,     strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT)); OCIERROR(errhp,     OCIStmtExecute(tpcsvc,pctx-&gt;curpi,errhp,1,0,0,OCI_DEFAULT)); #ifdef PLSQLPAY /* prepare the stub for calling psql stored procedure */ sprintf((char*)stmbuf,SQLTXT_STP); OCIERROR(errhp,OCIStmtPrepare(pctx-&gt;curp0, errhp, stmbuf,     strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT)); #else  /* customer id != 0, go by last name */  sqlfile("paynz.sql",stmbuf); OCIERROR(errhp,OCIStmtPrepare(pctx-&gt;curp0, errhp, stmbuf,     strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT));  /* customer id == 0, go by last name */  sqlfile("payz.sql",stmbuf); /* sqlfile opens SO/bench/.../blocks/... */ OCIERROR(errhp,OCIStmtPrepare(pctx-&gt;curp1, errhp, stmbuf,     strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT)); #endif pctx-&gt;w_id_ind = TRUE; pctx-&gt;w_id_len = SIZ(payP-&gt;w_id); pctx-&gt;d_id_ind = TRUE; pctx-&gt;d_id_len = SIZ(payP-&gt;d_id); pctx-&gt;c_w_id_ind = TRUE; pctx-&gt;c_w_id_len = SIZ(payP-&gt;c_w_id); pctx-&gt;c_d_id_ind = TRUE; pctx-&gt;c_d_id_len = SIZ(payP-&gt;c_d_id); pctx-&gt;c_id_ind = TRUE; pctx-&gt;c_id_len = 0; pctx-&gt;h_amount_len = SIZ(payP-&gt;h_amount); pctx-&gt;h_amount_ind = TRUE; pctx-&gt;c_last_ind = TRUE; pctx-&gt;c_last_len = 0; pctx-&gt;w_street_1_ind = TRUE; pctx-&gt;w_street_1_len = 0; pctx-&gt;w_street_2_ind = TRUE; pctx-&gt;w_street_2_len = 0; pctx-&gt;w_city_ind = TRUE; pctx-&gt;w_city_len = 0; pctx-&gt;w_state_ind = TRUE; pctx-&gt;w_state_len = 0; pctx-&gt;w_zip_ind = TRUE; pctx-&gt;w_zip_len = 0; pctx-&gt;d_street_1_ind = TRUE; pctx-&gt;d_street_1_len = 0; pctx-&gt;d_street_2_ind = TRUE; pctx-&gt;d_street_2_len = 0; pctx-&gt;d_city_ind = TRUE; pctx-&gt;d_city_len = 0; pctx-&gt;d_state_ind = TRUE; pctx-&gt;d_state_len = 0; pctx-&gt;d_zip_ind = TRUE; pctx-&gt;d_zip_len = 0; pctx-&gt;c_first_ind = TRUE; pctx-&gt;c_first_len = 0; pctx-&gt;c_middle_ind = TRUE; pctx-&gt;c_middle_len = 0; pctx-&gt;c_street_1_ind = TRUE; pctx-&gt;c_street_1_len = 0; pctx-&gt;c_street_2_ind = TRUE; pctx-&gt;c_street_2_len = 0; pctx-&gt;c_city_ind = TRUE; pctx-&gt;c_city_len = 0; pctx-&gt;c_state_ind = TRUE; pctx-&gt;c_state_len = 0; pctx-&gt;c_zip_ind = TRUE; </pre>	<pre> pctx-&gt;c_zip_len = 0; pctx-&gt;c_phone_ind = TRUE; pctx-&gt;c_phone_len = 0; pctx-&gt;c_since_ind = TRUE; pctx-&gt;c_since_len = 0; pctx-&gt;c_credit_ind = TRUE; pctx-&gt;c_credit_len = 0; pctx-&gt;c_credit_lim_ind = TRUE; pctx-&gt;c_credit_lim_len = 0; pctx-&gt;c_discount_ind = TRUE; pctx-&gt;c_discount_len = 0; pctx-&gt;c_balance_ind = TRUE; pctx-&gt;c_balance_len = sizeof(double); pctx-&gt;c_data_ind = TRUE; pctx-&gt;c_data_len = 0; pctx-&gt;h_date_ind = TRUE; pctx-&gt;h_date_len = 0; pctx-&gt;retries_ind = TRUE; pctx-&gt;retries_len = 0; pctx-&gt;cr_date_ind = TRUE; pctx-&gt;cr_date_len = 7;  /* bind variables */  OCIBNDR(pctx-&gt;curp0, pctx-&gt;w_id_bp, errhp, "w_id",ADR(payP-&gt;w_id),SIZ(int),     SQLT_INT, &amp;pctx-&gt;w_id_ind, NULL, NULL); OCIBNDR(pctx-&gt;curp0, pctx-&gt;d_id_bp, errhp, "d_id",ADR(payP-&gt;d_id),SIZ(int),     SQLT_INT, &amp;pctx-&gt;d_id_ind, NULL, NULL); OCIBNDR(pctx-&gt;curp0, pctx-&gt;c_w_id_bp, errhp, "c_w_id",ADR(payP-&gt;c_w_id),SIZ(int),     SQLT_INT); OCIBNDR(pctx-&gt;curp0, pctx-&gt;c_d_id_bp, errhp, "c_d_id",ADR(payP-&gt;c_d_id),SIZ(int),     SQLT_INT); OCIBNDR(pctx-&gt;curp0, pctx-&gt;c_id_bp, errhp, "c_id",ADR(payP-&gt;c_id),SIZ(int),     SQLT_INT); #ifdef PLSQLPAY OCIBNDR(pctx-&gt;curp0, pctx-&gt;by_name_bp, errhp, "by_name",ADR(payP-&gt;bylastname),     SIZ(int), SQLT_INT); #endif #ifdef OCIBNDR(pctx-&gt;curp0, pctx-&gt;h_amount_bp, errhp, "h_amount",ADR(payP-&gt;h_amount),     SIZ(int),SQLT_INT, &amp;pctx-&gt;h_amount_ind, &amp;pctx-&gt;h_amount_len,     &amp;pctx-&gt;h_amount_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;c_last_bp, errhp, "c_last",payP-&gt;c_last,SIZ(payP-&gt;c_last),     SQLT_STR, &amp;pctx-&gt;c_last_ind, &amp;pctx-&gt;c_last_len, &amp;pctx-&gt;c_last_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;w_street_1_bp, errhp, "w_street_1",payP-&gt;w_street_1,     SIZ(payP-&gt;w_street_1),SQLT_STR, &amp;pctx-&gt;w_street_1_ind,     &amp;pctx-&gt;w_street_1_len, &amp;pctx-&gt;w_street_1_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;w_street_2_bp, errhp, "w_street_2",payP-&gt;w_street_2,     SIZ(payP-&gt;w_street_2),SQLT_STR, &amp;pctx-&gt;w_street_2_ind,     &amp;pctx-&gt;w_street_2_len, &amp;pctx-&gt;w_street_2_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;w_city_bp, errhp, "w_city",payP-&gt;w_city,SIZ(payP-&gt;w_city),     SQLT_STR, &amp;pctx-&gt;w_city_ind, &amp;pctx-&gt;w_city_len, &amp;pctx-&gt;w_city_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;w_state_bp, errhp, "w_state",payP-&gt;w_state,SIZ(payP-&gt;w_state),     SQLT_STR, &amp;pctx-&gt;w_state_ind, &amp;pctx-&gt;w_state_len, &amp;pctx-&gt;w_state_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;w_zip_bp, errhp, "w_zip",payP-&gt;w_zip,SIZ(payP-&gt;w_zip),     SQLT_STR, &amp;pctx-&gt;w_zip_ind, &amp;pctx-&gt;w_zip_len, &amp;pctx-&gt;w_zip_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;d_street_1_bp, errhp, "d_street_1",payP-&gt;d_street_1,     SIZ(payP-&gt;d_street_1),SQLT_STR, &amp;pctx-&gt;d_street_1_ind,     &amp;pctx-&gt;d_street_1_len, &amp;pctx-&gt;d_street_1_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;d_street_2_bp, errhp, "d_street_2",payP-&gt;d_street_2,     SIZ(payP-&gt;d_street_2),SQLT_STR, &amp;pctx-&gt;d_street_2_ind,     &amp;pctx-&gt;d_street_2_len, &amp;pctx-&gt;d_street_2_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;d_city_bp, errhp, "d_city",payP-&gt;d_city,SIZ(payP-&gt;d_city),     SQLT_STR, &amp;pctx-&gt;d_city_ind, &amp;pctx-&gt;d_city_len, &amp;pctx-&gt;d_city_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;d_state_bp, errhp, "d_state",payP-&gt;d_state,SIZ(payP-&gt;d_state),     SQLT_STR, &amp;pctx-&gt;d_state_ind, &amp;pctx-&gt;d_state_len, &amp;pctx-&gt;d_state_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;d_zip_bp, errhp, "d_zip",payP-&gt;d_zip,SIZ(payP-&gt;d_zip),     SQLT_STR, &amp;pctx-&gt;d_zip_ind, &amp;pctx-&gt;d_zip_len, &amp;pctx-&gt;d_zip_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;c_first_bp, errhp, "c_first",payP-&gt;c_first,SIZ(payP-&gt;c_first),     SQLT_STR, &amp;pctx-&gt;c_first_ind, &amp;pctx-&gt;c_first_len, &amp;pctx-&gt;c_first_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;c_middle_bp, errhp, "c_middle",payP-&gt;c_middle,2,     SQLT_AFC, &amp;pctx-&gt;c_middle_ind, &amp;pctx-&gt;c_middle_len,     &amp;pctx-&gt;c_middle_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;c_street_1_bp, errhp, "c_street_1",payP-&gt;c_street_1,     SIZ(payP-&gt;c_street_1),SQLT_STR, &amp;pctx-&gt;c_street_1_ind,     &amp;pctx-&gt;c_street_1_len, &amp;pctx-&gt;c_street_1_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;c_street_2_bp, errhp, "c_street_2",payP-&gt;c_street_2,     SIZ(payP-&gt;c_street_2),SQLT_STR, &amp;pctx-&gt;c_street_2_ind,     &amp;pctx-&gt;c_street_2_len, &amp;pctx-&gt;c_street_2_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;c_city_bp, errhp, "c_city",payP-&gt;c_city,SIZ(payP-&gt;c_city),     SQLT_STR, &amp;pctx-&gt;c_city_ind, &amp;pctx-&gt;c_city_len, &amp;pctx-&gt;c_city_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;c_state_bp, errhp, "c_state",payP-&gt;c_state,SIZ(payP-&gt;c_state),     SQLT_STR, &amp;pctx-&gt;c_state_ind, &amp;pctx-&gt;c_state_len, &amp;pctx-&gt;c_state_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;c_zip_bp, errhp, "c_zip",payP-&gt;c_zip,SIZ(payP-&gt;c_zip),     SQLT_STR, &amp;pctx-&gt;c_zip_ind, &amp;pctx-&gt;c_zip_len, &amp;pctx-&gt;c_zip_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;c_phone_bp, errhp, "c_phone",payP-&gt;c_phone,SIZ(payP-&gt;c_phone),     SQLT_STR, &amp;pctx-&gt;c_phone_ind, &amp;pctx-&gt;c_phone_len, &amp;pctx-&gt;c_phone_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;c_since_bp, errhp, "c_since",&amp;payP-&gt;c_since,     SIZ(OCIDate), SQLT_ODT, &amp;pctx-&gt;c_since_ind, &amp;pctx-&gt;c_since_len,     &amp;pctx-&gt;c_since_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;c_credit_bp, errhp, "c_credit",payP-&gt;c_credit,     SIZ(payP-&gt;c_credit),SQLT_CHR, &amp;pctx-&gt;c_credit_ind, &amp;pctx-&gt;c_credit_len,     &amp;pctx-&gt;c_credit_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;c_credit_lim_bp, errhp, "c_credit_lim",     ADR(payP-&gt;c_credit_lim),SIZ(int), SQLT_INT, &amp;pctx-&gt;c_credit_lim_ind,     &amp;pctx-&gt;c_credit_lim_len, &amp;pctx-&gt;c_credit_lim_rc); OCIBNDR(pctx-&gt;curp0, pctx-&gt;c_discount_bp, errhp, "c_discount", </pre>
--	--

<pre> ADR(payP-&gt;c_discount,SIZ(float),SQLT_FLT,&amp;ptcx-&gt;c_discount_ind, &amp;ptcx-&gt;c_discount_len,&amp;ptcx-&gt;c_discount_rc); OCIBNDR(pctx-&gt;curp0,ptcx-&gt;c_balance_bp,errhp,":c_balance",ADR(payP-&gt;c_balance), SIZ(double),SQLT_FLT,&amp;ptcx-&gt;c_balance_ind,&amp;ptcx-&gt;c_balance_len, &amp;ptcx-&gt;c_balance_rc); OCIBNDR(pctx-&gt;curp0,ptcx-&gt;c_data_bp,errhp,":c_data",payP-&gt;c_data,SIZ(payP-&gt;c_data), SQLT_STR,&amp;ptcx-&gt;c_data_ind,&amp;ptcx-&gt;c_data_len,&amp;ptcx-&gt;c_data_rc); */ OCIBNDR(pctx-&gt;curp0,ptcx-&gt;h_date_bp,errhp,":h_date",payP-&gt;h_date,SIZ(payP-&gt;h_date), SQLT_STR,&amp;ptcx-&gt;h_date_ind,&amp;ptcx-&gt;h_date_len,&amp;ptcx-&gt;h_date_rc); */ OCIBNDR(pctx-&gt;curp0,ptcx-&gt;retries_bp,errhp,":retries",ADR(payP-&gt;retries),SIZ(int), SQLT_INT,&amp;ptcx-&gt;retries_ind,&amp;ptcx-&gt;retries_len,&amp;ptcx-&gt;retries_rc); OCIBNDR(pctx-&gt;curp0,ptcx-&gt;cr_date_bp,errhp,":cr_date",ADR(payP-&gt;cr_date), SIZ(OCIDate),SQLT_ODT,&amp;ptcx-&gt;cr_date_ind,&amp;ptcx-&gt;cr_date_len, &amp;ptcx-&gt;cr_date_rc); #endifdef PLSQLPAY */ ---- Binds for the second cursor */ OCIBNDR(pctx-&gt;curp1,ptcx-&gt;w_id_bp1,errhp,":w_id",ADR(payP-&gt;w_id),SIZ(int), SQLT_INT,&amp;ptcx-&gt;w_id_ind,&amp;ptcx-&gt;w_id_len,&amp;ptcx-&gt;w_id_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;d_id_bp1,errhp,":d_id",ADR(payP-&gt;d_id),SIZ(int), SQLT_INT,&amp;ptcx-&gt;d_id_ind,&amp;ptcx-&gt;d_id_len,&amp;ptcx-&gt;d_id_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;e_w_id_bp1,errhp,":e_w_id",ADR(payP-&gt;e_w_id),SIZ(int), SQLT_INT); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;c_d_id_bp1,errhp,":c_d_id",ADR(payP-&gt;c_d_id),SIZ(int), SQLT_INT); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;c_id_bp1,errhp,":c_id",ADR(payP-&gt;c_id),SIZ(int), SQLT_INT,&amp;ptcx-&gt;c_id_ind,&amp;ptcx-&gt;c_id_len,&amp;ptcx-&gt;c_id_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;h_amount_bp1,errhp,":h_amount",ADR(payP-&gt;h_amount), SIZ(int),SQLT_INT,&amp;ptcx-&gt;h_amount_ind,&amp;ptcx-&gt;h_amount_len, &amp;ptcx-&gt;h_amount_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;c_last_bp1,errhp,":c_last",payP-&gt;c_last,SIZ(payP-&gt;c_last), SQLT_STR); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;w_street_1_bp1,errhp,":w_street_1",payP-&gt;w_street_1, SIZ(payP-&gt;w_street_1),SQLT_STR,&amp;ptcx-&gt;w_street_1_ind, &amp;ptcx-&gt;w_street_1_len,&amp;ptcx-&gt;w_street_1_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;w_street_2_bp1,errhp,":w_street_2",payP-&gt;w_street_2, SIZ(payP-&gt;w_street_2),SQLT_STR,&amp;ptcx-&gt;w_street_2_ind, &amp;ptcx-&gt;w_street_2_len,&amp;ptcx-&gt;w_street_2_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;w_city_bp1,errhp,":w_city",payP-&gt;w_city,SIZ(payP-&gt;w_city), SQLT_STR,&amp;ptcx-&gt;w_city_ind,&amp;ptcx-&gt;w_city_len,&amp;ptcx-&gt;w_city_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;w_state_bp1,errhp,":w_state",payP-&gt;w_state,SIZ(payP-&gt;w_state), SQLT_STR,&amp;ptcx-&gt;w_state_ind,&amp;ptcx-&gt;w_state_len,&amp;ptcx-&gt;w_state_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;w_zip_bp1,errhp,":w_zip",payP-&gt;w_zip,SIZ(payP-&gt;w_zip), SQLT_STR,&amp;ptcx-&gt;w_zip_ind,&amp;ptcx-&gt;w_zip_len,&amp;ptcx-&gt;w_zip_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;d_street_1_bp1,errhp,":d_street_1",payP-&gt;d_street_1, SIZ(payP-&gt;d_street_1),SQLT_STR,&amp;ptcx-&gt;d_street_1_ind, &amp;ptcx-&gt;d_street_1_len,&amp;ptcx-&gt;d_street_1_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;d_street_2_bp1,errhp,":d_street_2",payP-&gt;d_street_2, SIZ(payP-&gt;d_street_2),SQLT_STR,&amp;ptcx-&gt;d_street_2_ind, &amp;ptcx-&gt;d_street_2_len,&amp;ptcx-&gt;d_street_2_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;d_city_bp1,errhp,":d_city",payP-&gt;d_city,SIZ(payP-&gt;d_city), SQLT_STR,&amp;ptcx-&gt;d_city_ind,&amp;ptcx-&gt;d_city_len,&amp;ptcx-&gt;d_city_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;d_state_bp1,errhp,":d_state",payP-&gt;d_state, SIZ(payP-&gt;d_state),SQLT_STR,&amp;ptcx-&gt;d_state_ind,&amp;ptcx-&gt;d_state_len, &amp;ptcx-&gt;d_state_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;d_zip_bp1,errhp,":d_zip",payP-&gt;d_zip,SIZ(payP-&gt;d_zip), SQLT_STR,&amp;ptcx-&gt;d_zip_ind,&amp;ptcx-&gt;d_zip_len,&amp;ptcx-&gt;d_zip_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;c_first_bp1,errhp,":c_first",payP-&gt;c_first, SIZ(payP-&gt;c_first),SQLT_STR,&amp;ptcx-&gt;c_first_ind,&amp;ptcx-&gt;c_first_len, &amp;ptcx-&gt;c_first_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;c_middle_bp1,errhp,":c_middle",payP-&gt;c_middle,2, SQLT_AFC,&amp;ptcx-&gt;c_middle_ind,&amp;ptcx-&gt;c_middle_len, &amp;ptcx-&gt;c_middle_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;c_street_1_bp1,errhp,":c_street_1",payP-&gt;c_street_1, SIZ(payP-&gt;c_street_1),SQLT_STR,&amp;ptcx-&gt;c_street_1_ind, &amp;ptcx-&gt;c_street_1_len,&amp;ptcx-&gt;c_street_1_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;c_street_2_bp1,errhp,":c_street_2",payP-&gt;c_street_2, SIZ(payP-&gt;c_street_2),SQLT_STR,&amp;ptcx-&gt;c_street_2_ind, &amp;ptcx-&gt;c_street_2_len,&amp;ptcx-&gt;c_street_2_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;c_city_bp1,errhp,":c_city",payP-&gt;c_city,SIZ(payP-&gt;c_city),SQLT_STR, &amp;ptcx-&gt;c_city_ind,&amp;ptcx-&gt;c_city_len,&amp;ptcx-&gt;c_city_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;c_state_bp1,errhp,":c_state",payP-&gt;c_state,SIZ(payP-&gt;c_state), SQLT_STR,&amp;ptcx-&gt;c_state_ind,&amp;ptcx-&gt;c_state_len,&amp;ptcx-&gt;c_state_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;c_zip_bp1,errhp,":c_zip",payP-&gt;c_zip,SIZ(payP-&gt;c_zip), SQLT_STR,&amp;ptcx-&gt;c_zip_ind,&amp;ptcx-&gt;c_zip_len,&amp;ptcx-&gt;c_zip_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;c_phone_bp1,errhp,":c_phone",payP-&gt;c_phone,SIZ(payP-&gt;c_phone), SQLT_STR,&amp;ptcx-&gt;c_phone_ind,&amp;ptcx-&gt;c_phone_len,&amp;ptcx-&gt;c_phone_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;c_since_bp1,errhp,":c_since",&amp;payP-&gt;c_since, SIZ(OCIDate),SQLT_ODT,&amp;ptcx-&gt;c_since_ind,&amp;ptcx-&gt;c_since_len, &amp;ptcx-&gt;c_since_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;c_credit_bp1,errhp,":c_credit",payP-&gt;c_credit, SIZ(payP-&gt;c_credit),SQLT_CHR,&amp;ptcx-&gt;c_credit_ind,&amp;ptcx-&gt;c_credit_len, &amp;ptcx-&gt;c_credit_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;c_credit_lim_bp1,errhp,":c_credit_lim", ADR(payP-&gt;c_credit_lim),SIZ(int),SQLT_INT,&amp;ptcx-&gt;c_credit_lim_ind, &amp;ptcx-&gt;c_credit_lim_len,&amp;ptcx-&gt;c_credit_lim_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;c_discount_bp1,errhp,":c_discount", ADR(payP-&gt;c_discount),SIZ(int),SQLT_FLT,&amp;ptcx-&gt;c_discount_ind, &amp;ptcx-&gt;c_discount_len,&amp;ptcx-&gt;c_discount_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;c_balance_bp1,errhp,":c_balance",ADR(payP-&gt;c_balance), SIZ(double),SQLT_FLT,&amp;ptcx-&gt;c_balance_ind,&amp;ptcx-&gt;c_balance_len, &amp;ptcx-&gt;c_balance_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;c_data_bp1,errhp,":c_data",payP-&gt;c_data,SIZ(payP-&gt;c_data), SQLT_STR,&amp;ptcx-&gt;c_data_ind,&amp;ptcx-&gt;c_data_len,&amp;ptcx-&gt;c_data_rc); </pre>	<pre> /* OCIBNDR(pctx-&gt;curp1,ptcx-&gt;h_date_bp1,errhp,":h_date",payP-&gt;h_date,SIZ(payP-&gt;h_date), SQLT_STR,&amp;ptcx-&gt;h_date_ind,&amp;ptcx-&gt;h_date_len,&amp;ptcx-&gt;h_date_rc); */ OCIBNDR(pctx-&gt;curp1,ptcx-&gt;retries_bp1,errhp,":retries",ADR(payP-&gt;retries),SIZ(int), SQLT_INT,&amp;ptcx-&gt;retries_ind,&amp;ptcx-&gt;retries_len,&amp;ptcx-&gt;retries_rc); OCIBNDR(pctx-&gt;curp1,ptcx-&gt;cr_date_bp1,errhp,":cr_date",ADR(payP-&gt;cr_date), SIZ(OCIDate),SQLT_ODT,&amp;ptcx-&gt;cr_date_ind,&amp;ptcx-&gt;cr_date_len, &amp;ptcx-&gt;cr_date_rc); #endif return (0); }  tkvcp (ora_cn_data_t *ora_SlotDataP) { /* */ payctx *pctx = ora_SlotDataP-&gt;pctx; global_payment_t *payP = ora_SlotDataP-&gt;payP; OCIEnv *tpcenv = ora_SlotDataP-&gt;tpcenv; OCIServer *tpcsrv = ora_SlotDataP-&gt;tpcsrv; OCIError *errhp = ora_SlotDataP-&gt;errhp; OCISvcCtx *tpscvc = ora_SlotDataP-&gt;tpscvc; OCISession *tpcusr = ora_SlotDataP-&gt;tpcusr; OCIStmt *curi = ora_SlotDataP-&gt;curi;  retry: ptcx-&gt;w_id_ind = TRUE; ptcx-&gt;w_id_len = SIZ(payP-&gt;w_id); ptcx-&gt;d_id_ind = TRUE; ptcx-&gt;d_id_len = SIZ(payP-&gt;d_id); ptcx-&gt;e_w_id_ind = TRUE; ptcx-&gt;c_d_id_ind = TRUE; ptcx-&gt;c_id_ind = TRUE; ptcx-&gt;h_amount_len = SIZ(payP-&gt;h_amount); ptcx-&gt;h_amount_ind = TRUE; ptcx-&gt;c_last_ind = TRUE; ptcx-&gt;c_last_len = SIZ(payP-&gt;c_last); ptcx-&gt;w_street_1_ind = TRUE; ptcx-&gt;w_street_1_len = 0; ptcx-&gt;w_street_2_ind = TRUE; ptcx-&gt;w_street_2_len = 0; ptcx-&gt;w_city_ind = TRUE; ptcx-&gt;w_city_len = 0; ptcx-&gt;w_state_ind = TRUE; ptcx-&gt;w_state_len = 0; ptcx-&gt;w_zip_ind = TRUE; ptcx-&gt;w_zip_len = 0; ptcx-&gt;d_street_1_ind = TRUE; ptcx-&gt;d_street_1_len = 0; ptcx-&gt;d_street_2_ind = TRUE; ptcx-&gt;d_street_2_len = 0; ptcx-&gt;d_city_ind = TRUE; ptcx-&gt;d_city_len = 0; ptcx-&gt;d_state_ind = TRUE; ptcx-&gt;d_state_len = 0; ptcx-&gt;d_zip_ind = TRUE; ptcx-&gt;d_zip_len = 0; ptcx-&gt;c_first_ind = TRUE; ptcx-&gt;c_first_len = 0; ptcx-&gt;c_middle_ind = TRUE; ptcx-&gt;c_middle_len = 0; ptcx-&gt;c_street_1_ind = TRUE; ptcx-&gt;c_street_1_len = 0; ptcx-&gt;c_street_2_ind = TRUE; ptcx-&gt;c_street_2_len = 0; ptcx-&gt;c_city_ind = TRUE; ptcx-&gt;c_city_len = 0; ptcx-&gt;c_state_ind = TRUE; ptcx-&gt;c_state_len = 0; ptcx-&gt;c_zip_ind = TRUE; ptcx-&gt;c_zip_len = 0; ptcx-&gt;c_phone_ind = TRUE; ptcx-&gt;c_phone_len = 0; ptcx-&gt;c_since_ind = TRUE; ptcx-&gt;c_since_len = 0; ptcx-&gt;c_credit_ind = TRUE; ptcx-&gt;c_credit_len = 0; ptcx-&gt;c_credit_lim_ind = TRUE; ptcx-&gt;c_credit_lim_len = 0; ptcx-&gt;c_discount_ind = TRUE; ptcx-&gt;c_discount_len = 0; ptcx-&gt;c_balance_ind = TRUE; ptcx-&gt;c_balance_len = sizeof(double); ptcx-&gt;c_data_ind = TRUE; ptcx-&gt;c_data_len = 0; ptcx-&gt;h_date_ind = TRUE; ptcx-&gt;h_date_len = 0; ptcx-&gt;retries_ind = TRUE; ptcx-&gt;retries_len = 0; ptcx-&gt;cr_date_ind = TRUE; </pre>
---	---

```

pctx->cr_date_len = 7;

#ifdef PLSQLPAY
payP->execstatus=OCIStmtExecute(tpcsvc,pctx->curp0,errhp,1,0,0,OCI_DEFAULT|OCI_COMMIT_ON_SUCCESS);
#else
if(payP->bylastname) {
payP->execstatus=OCIStmtExecute(tpcsvc,pctx->curp1,errhp,1,0,0,OCI_DEFAULT|OCI_COMMIT_ON_SUCCESS);
} else {
payP->execstatus=OCIStmtExecute(tpcsvc,pctx->curp0,errhp,1,0,0,OCI_DEFAULT|OCI_COMMIT_ON_SUCCESS);
}
#endif

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

void tkvcpdone (ora_cn_data_t *ora_SlotDataP)
{
/* TODO: Should we free the cursor handles?? */
if(ora_SlotDataP->pctx) {
free(ora_SlotDataP->pctx);
ora_SlotDataP->pctx = NULL;
}
if(ora_SlotDataP->payP) {
free(ora_SlotDataP->payP);
ora_SlotDataP->payP = NULL;
}
}

plsto.c

#ifdef RCSID
static char *RCSid =
"$Header: /afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/server/ora8.1_mt/RCS/plsto.c,v 1.2 1999/04/15 12:16:52 oz Exp $ 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"
#include "plora.h" /* */
#include "tpccflags.h"

#ifdef PLSQLSTO
#define SQLTXT "BEGIN stocklevel.getstocklevel (:w_id,:d_id,:threshold,\
:low_stock); END;"
#else
#define SQLTXT "SELECT /*+ nocache(stock) */ \
count (DISTINCT s_i_id) \
FROM order_line, stock, district \
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)"
/* query using functional index */
#define SQLTXT "SELECT count (DISTINCT s_i_id) \
FROM order_line, stock, district \
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_o_id BETWEEN (d_next_o_id - 20) AND (d_next_o_id - 1) AND \
decode(SIGN(s_quantity - 21), -1, s_w_id * 100000 + s_i_id, NULL) \
= ol_w_id * 100000 + ol_i_id AND \
s_quantity < :threshold;"
*/

```

```

#endif

struct stoctx {
OCIStmt *curs;
OCIBind *w_id_bp;
OCIBind *d_id_bp;
OCIBind *threshold_bp;
#ifdef PLSQLSTO
OCIBind *low_stock_bp;
#endif
OCIDefine *low_stock_bp;
#endif
int norow;
};

typedef struct stoctx stoctx;

/* stoctx *sctx; */

tkvcscinit (ora_cn_data_t *ora_SlotDataP)
{
/* */
stoctx *sctx;
global_stock_t *stoP;
OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
OCIError *errhp = ora_SlotDataP->errhp;
OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
OCISession *tpcusr = ora_SlotDataP->tpcusr;
OCIStmt *curi = ora_SlotDataP->curi;

text stmbuf[SQL_BUF_SIZE];

sctx = (stoctx *) malloc(sizeof(stoctx));
memset(sctx,(char)0,sizeof(stoctx));
ora_SlotDataP->sctx = (void *) sctx;

ora_SlotDataP->stoP = (global_stock_t *) malloc(sizeof(global_stock_t));
memset(ora_SlotDataP->stoP,(char)0,sizeof(global_stock_t));
stoP = ora_SlotDataP->stoP;

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));

#ifdef PLSQLSTO
OCIERROR(errhp,
OCIAttrSet(sctx->curs,OCI_HTYPE_STMT,(dvoid *) &sctx->norow,0,
OCI_ATTR_PREFETCH_ROWS,errhp));
#endif

/* bind variables */

OCIBND(sctx->curs,sctx->w_id_bp,errhp, "w_id", ADR(stoP->w_id),sizeof(int),
SQLT_INT);
OCIBND(sctx->curs,sctx->d_id_bp,errhp, "d_id", ADR(stoP->d_id),sizeof(int),
SQLT_INT);
OCIBND(sctx->curs,sctx->threshold_bp,errhp, "threshold", ADR(stoP->threshold),
sizeof(int),SQLT_INT);
#ifdef PLSQLSTO
OCIBND(sctx->curs,sctx->low_stock_bp,errhp,"low_stock", ADR(stoP->low_stock),
sizeof(int),SQLT_INT);
#endif
#ifndef PLSQLSTO
OCIDEFINE(sctx->curs,sctx->low_stock_bp,errhp, 1, ADR(stoP->low_stock),
sizeof(int),SQLT_INT);
#endif

return (0);
}

tkvcs (ora_cn_data_t *ora_SlotDataP)
{
stoctx *sctx = (stoctx *) ora_SlotDataP->sctx;
global_stock_t *stoP = ora_SlotDataP->stoP;
OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
OCIError *errhp = ora_SlotDataP->errhp;
OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
OCISession *tpcusr = ora_SlotDataP->tpcusr;
OCIStmt *curi = ora_SlotDataP->curi;

retry:
stoP->execstatus= OCIStmtExecute(tpcsvc,sctx->curs,errhp,1,0,0,
OCI_COMMIT_ON_SUCCESS | OCI_DEFAULT);
if (stoP->execstatus != OCI_SUCCESS)
{
stoP->errcode=OCIERROR(errhp,stoP->execstatus);
OCITransCommit(tpcsvc,errhp,OCI_DEFAULT);
}
}

```

```

if((stoP->errcode == NOT_SERIALIZABLE) || (stoP->errcode == RECOVER))
{
stoP->retries++;
goto retry;
} else {
return -1;
}
}
return (0);
}

void tkvcdone (ora_cn_data_t *ora_SlotDataP)
{
/* */
stoctx *sctx = (stoctx *)ora_SlotDataP->sctx;
if (sctx) {
free(sctx);
ora_SlotDataP->sctx = NULL;
}
if (ora_SlotDataP->stoP) {
free(ora_SlotDataP->stoP);
ora_SlotDataP->stoP = NULL;
}
}

```

### tkvcin.sql

```

-- The initnew package for storing variables used in the
-- New Order anonymous block

```

```

CREATE OR REPLACE PACKAGE initnew
AS
TYPE intarray IS TABLE OF INTEGER index by binary_integer;
TYPE distarray IS TABLE OF VARCHAR(24) index by binary_integer;
nulldate DATE;
s_dist distarray;
idxlarr intarray;
s_remote intarray;
PROCEDURE new_init(idxarr intarray);
END initnew;
/
show errors;

CREATE OR REPLACE PACKAGE BODY initnew AS
PROCEDURE new_init (idxarr intarray)
IS
BEGIN
-- initialize null date
nulldate := TO_DATE('01-01-1811', 'MM-DD-YYYY');
idxlarr := idxarr;
END new_init;
END initnew;
/
show errors
exit

```

### tkvcnew.sql

```

-- New Order Anonymous block

```

```

DECLARE
idx BINARY_INTEGER;
dummy_local BINARY_INTEGER;
not_serializable EXCEPTION;
PRAGMA EXCEPTION_INIT(not_serializable,-8177);
deadlock EXCEPTION;
PRAGMA EXCEPTION_INIT(deadlock,-60);
snapshot_too_old EXCEPTION;
PRAGMA EXCEPTION_INIT(snapshot_too_old,-1555);
PROCEDURE u1 IS
BEGIN
FORALL idx IN 1 .. :o_ol_cnt
UPDATE stock_item
SET s_order_cnt = s_order_cnt + 1,
s_ytd = s_ytd + :ol_quantity(idx),
s_remote_cnt = s_remote_cnt + :s_remote(idx),
s_quantity = s_quantity - :ol_quantity(idx) +
DECODE(sign(s_quantity - :ol_quantity(idx)) -
10),-1,91,0)
WHERE i_id = :ol_i_id(idx)
AND s_w_id = :ol_supply_w_id(idx)
RETURNING i_price, i_name, s_quantity, s_dist_01,
DECODE (instr(i_data,'original'), 0, 'G',
DECODE(instr(s_data,'original'), 0, 'G', 'B'))
BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist,
brand_generic;

END u1;

PROCEDURE u2 IS
BEGIN
FORALL idx IN 1 .. :o_ol_cnt

```

```

UPDATE stock_item
SET s_order_cnt = s_order_cnt + 1,
s_ytd = s_ytd + :ol_quantity(idx),
s_remote_cnt = s_remote_cnt + :s_remote(idx),
s_quantity = s_quantity - :ol_quantity(idx) +
DECODE(sign(s_quantity - :ol_quantity(idx)) -
10),-1,91,0)
WHERE i_id = :ol_i_id(idx)
AND s_w_id = :ol_supply_w_id(idx)
RETURNING i_price, i_name, s_quantity, s_dist_02,
DECODE (instr(i_data,'original'), 0, 'G',
DECODE(instr(s_data,'original'), 0, 'G', 'B'))
BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist,
brand_generic;

END u2;

PROCEDURE u3 IS
BEGIN
FORALL idx IN 1 .. :o_ol_cnt
UPDATE stock_item
SET s_order_cnt = s_order_cnt + 1,
s_ytd = s_ytd + :ol_quantity(idx),
s_remote_cnt = s_remote_cnt + :s_remote(idx),
s_quantity = s_quantity - :ol_quantity(idx) +
DECODE(sign(s_quantity - :ol_quantity(idx)) -
10),-1,91,0)
WHERE i_id = :ol_i_id(idx)
AND s_w_id = :ol_supply_w_id(idx)
RETURNING i_price, i_name, s_quantity, s_dist_03,
DECODE (instr(i_data,'original'), 0, 'G',
DECODE(instr(s_data,'original'), 0, 'G', 'B'))
BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist,
brand_generic;

END u3;

PROCEDURE u4 IS
BEGIN
FORALL idx IN 1 .. :o_ol_cnt
UPDATE stock_item
SET s_order_cnt = s_order_cnt + 1,
s_ytd = s_ytd + :ol_quantity(idx),
s_remote_cnt = s_remote_cnt + :s_remote(idx),
s_quantity = s_quantity - :ol_quantity(idx) +
DECODE(sign(s_quantity - :ol_quantity(idx)) -
10),-1,91,0)
WHERE i_id = :ol_i_id(idx)
AND s_w_id = :ol_supply_w_id(idx)
RETURNING i_price, i_name, s_quantity, s_dist_04,
DECODE (instr(i_data,'original'), 0, 'G',
DECODE(instr(s_data,'original'), 0, 'G', 'B'))
BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist,
brand_generic;

END u4;

PROCEDURE u5 IS
BEGIN
FORALL idx IN 1 .. :o_ol_cnt
UPDATE stock_item
SET s_order_cnt = s_order_cnt + 1,
s_ytd = s_ytd + :ol_quantity(idx),
s_remote_cnt = s_remote_cnt + :s_remote(idx),
s_quantity = s_quantity - :ol_quantity(idx) +
DECODE(sign(s_quantity - :ol_quantity(idx)) -
10),-1,91,0)
WHERE i_id = :ol_i_id(idx)
AND s_w_id = :ol_supply_w_id(idx)
RETURNING i_price, i_name, s_quantity, s_dist_05,
DECODE (instr(i_data,'original'), 0, 'G',
DECODE(instr(s_data,'original'), 0, 'G', 'B'))
BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist,
brand_generic;

END u5;

PROCEDURE u6 IS
BEGIN
FORALL idx IN 1 .. :o_ol_cnt
UPDATE stock_item
SET s_order_cnt = s_order_cnt + 1,
s_ytd = s_ytd + :ol_quantity(idx),
s_remote_cnt = s_remote_cnt + :s_remote(idx),
s_quantity = s_quantity - :ol_quantity(idx) +
DECODE(sign(s_quantity - :ol_quantity(idx)) -
10),-1,91,0)
WHERE i_id = :ol_i_id(idx)
AND s_w_id = :ol_supply_w_id(idx)
RETURNING i_price, i_name, s_quantity, s_dist_06,
DECODE (instr(i_data,'original'), 0, 'G',
DECODE(instr(s_data,'original'), 0, 'G', 'B'))
BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist,
brand_generic;

END u6;

PROCEDURE u7 IS
BEGIN
FORALL idx IN 1 .. :o_ol_cnt
UPDATE stock_item
SET s_order_cnt = s_order_cnt + 1,
s_ytd = s_ytd + :ol_quantity(idx),
s_remote_cnt = s_remote_cnt + :s_remote(idx),

```

<pre> s_quantity = s_quantity - :ol_quantity(idx) + DECODE(sign(s_quantity - :ol_quantity(idx) - 10),-1,91.0) WHERE i_id = :ol_i_id(idx) AND s_w_id = :ol_supply_w_id(idx) RETURNING i_price, i_name, s_quantity, s_dist_07, DECODE(instr(i_data,'original'), 0, 'G', DECODE(instr(s_data,'original'), 0, 'G', 'B')) BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist, :brand_generic; END u7;  PROCEDURE u8 IS BEGIN FORALL idx IN 1 .. :o_ol_cnt UPDATE stock_item SET s_order_cnt = s_order_cnt + 1, s_ytd = s_ytd + :ol_quantity(idx), s_remote_cnt = s_remote_cnt + :s_remote(idx), s_quantity = s_quantity - :ol_quantity(idx) + 10),-1,91.0) WHERE i_id = :ol_i_id(idx) AND s_w_id = :ol_supply_w_id(idx) RETURNING i_price, i_name, s_quantity, s_dist_08, DECODE(instr(i_data,'original'), 0, 'G', DECODE(instr(s_data,'original'), 0, 'G', 'B')) BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist, :brand_generic; END u8;  PROCEDURE u9 IS BEGIN FORALL idx IN 1 .. :o_ol_cnt UPDATE stock_item SET s_order_cnt = s_order_cnt + 1, s_ytd = s_ytd + :ol_quantity(idx), s_remote_cnt = s_remote_cnt + :s_remote(idx), s_quantity = s_quantity - :ol_quantity(idx) + 10),-1,91.0) WHERE i_id = :ol_i_id(idx) AND s_w_id = :ol_supply_w_id(idx) RETURNING i_price, i_name, s_quantity, s_dist_09, DECODE(instr(i_data,'original'), 0, 'G', DECODE(instr(s_data,'original'), 0, 'G', 'B')) BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist, :brand_generic; END u9;  PROCEDURE u10 IS BEGIN FORALL idx IN 1 .. :o_ol_cnt UPDATE stock_item SET s_order_cnt = s_order_cnt + 1, s_ytd = s_ytd + :ol_quantity(idx), s_remote_cnt = s_remote_cnt + :s_remote(idx), s_quantity = s_quantity - :ol_quantity(idx) + 10),-1,91.0) WHERE i_id = :ol_i_id(idx) AND s_w_id = :ol_supply_w_id(idx) RETURNING i_price, i_name, s_quantity, s_dist_10, DECODE(instr(i_data,'original'), 0, 'G', DECODE(instr(s_data,'original'), 0, 'G', 'B')) BULK COLLECT INTO :i_price, :i_name, :s_quantity, initnew.s_dist, :brand_generic; END u10;  PROCEDURE fix_items IS rows_lost BINARY_INTEGER; max_index BINARY_INTEGER; temp_index BINARY_INTEGER; BEGIN -- gotta shift price, name, s_quantity, brand_generic, s_dist, ol_amount idx := 1; -- found 0 bad rows rows_lost := 0; -- so many rows in out array to begin with max_index := sql%rowcount;  WHILE (max_index != :o_ol_cnt) LOOP -- find item where item ids dont match WHILE (idx &lt;= sql%rowcount AND sql%bulk_rowcount(idx + rows_lost) = 1) LOOP idx := idx + 1; END LOOP; -- shift the items please temp_index := max_index; WHILE (temp_index &gt;= idx + rows_lost) LOOP :i_price(temp_index + 1) := :i_price(temp_index); :i_name(temp_index + 1) := :i_name(temp_index); :s_quantity(temp_index + 1) := :s_quantity(temp_index); initnew.s_dist(temp_index + 1) := initnew.s_dist(temp_index); :brand_generic(temp_index + 1) := :brand_generic(temp_index); temp_index := temp_index - 1; </pre>	<pre> END LOOP; -- values for the non-existent items if not at end IF (idx + rows_lost &lt;= :o_ol_cnt) THEN :i_price(idx + rows_lost) := 0; :i_name(idx + rows_lost) := NULL; :s_quantity(idx + rows_lost) := 0; initnew.s_dist(idx + rows_lost) := NULL; :brand_generic(idx + rows_lost) := NULL;  -- one more bad row rows_lost := rows_lost + 1; max_index := max_index + 1; END IF;  END LOOP; END fix_items;  BEGIN LOOP BEGIN UPDATE district 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 customer, warehouse WHERE c_id = :c_id AND c_d_id = :d_id AND c_w_id = :w_id AND w_id = :w_id;  INSERT INTO new_order (no_o_id, no_d_id, no_w_id) VALUES (:o_id, :d_id, :w_id); INSERT INTO orders (o_id, o_d_id, o_w_id, o_c_id, o_entry_d, o_carrier_id, o_ol_cnt, o_all_local) VALUES (:o_id, :d_id, :w_id, :c_id, :cr_date, 11, :o_ol_cnt, :o_all_local);  -- copying :d_id in local variable . dummy_local := :d_id;  IF (dummy_local = 1) THEN u1; END IF; IF (dummy_local = 2) THEN u2; END IF; IF (dummy_local = 3) THEN u3; END IF; IF (dummy_local = 4) THEN u4; END IF; IF (dummy_local = 5) THEN u5; END IF; IF (dummy_local = 6) THEN u6; END IF; IF (dummy_local = 7) THEN u7; END IF; IF (dummy_local = 8) THEN u8; END IF; IF (dummy_local = 9) THEN u9; END IF; IF (dummy_local = 10) THEN u10; END IF;  -- cache the no of rows processed dummy_local := sql%rowcount;  -- fix the rows if necessary IF (dummy_local != :o_ol_cnt) THEN fix_items; END IF;  -- calculate ol_amount  FOR idx IN 1 ..:o_ol_cnt LOOP :ol_amount(idx) := :ol_quantity(idx) * :i_price(idx); END LOOP;  FORALL idx IN 1..:o_ol_cnt -- doesnt hurt if we insert entries for invalid item too INSERT INTO order_line (ol_o_id, ol_d_id, ol_w_id, ol_number, ol_delivery_d, ol_i_id, ol_supply_w_id, ol_quantity, ol_amount, ol_dist_info) VALUES (:o_id, :d_id, :w_id, initnew.idx_larr(idx), initnew.nulldate, :ol_i_id(idx), :ol_supply_w_id(idx), :ol_quantity(idx), :ol_amount(idx), initnew.s_dist(idx));  --If there are no errors, then just return without COMMITTING --The COMMIT is done on the driver side by OCI -- If there are errors, then rollback and set o_ol_cnt to the processed value IF (dummy_local != :o_ol_cnt) THEN :o_ol_cnt := dummy_local; ROLLBACK; END IF;  EXIT;  EXCEPTION WHEN not_serializable OR deadlock OR snapshot_too_old THEN ROLLBACK; :retry := :retry + 1; END; END LOOP; </pre>
---	---

<pre> END;                                  <b>tpcc.h</b>  /*  * \$Header: /afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/server/ora8.1_mt/RCS/tpcc.h,v 1.1 1999/04/14 19:03:06 wenjian Exp S 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 &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;ctype.h&gt; #include &lt;string.h&gt;  #include &lt;oratypes.h&gt; #include &lt;oci.h&gt; #include &lt;ocidfn.h&gt; /* #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 int TPCexit (); extern int TPCdumpinit (); extern int TPCdumpnew (); extern int TPCdumpppay (); extern int TPCdumpord (); extern int TPCdumpdel (); extern int TPCdumpsto (); extern int TPCdumpexit ();  /* Error codes */ #define RECOVERR -10 #define IRRECERR -20 #define NOERR 111 #define DEL_ERROR -666 #define DEL_DATE_LEN 7 #define NDISTS 10 #define NITEMS 15 #define SQL_BUF_SIZE 8192  #define FULLDATE "dd-mon-yy.hh:mi:ss" #define SHORTDATE "dd-mm-yyyy"  #define DELRT 80.0  extern int tkvcninit (); extern int tkvcpnit (); extern int tkvcoint (); extern int tkvcdinit (); extern int tkvcsinit ();  extern int tkvcn (); extern int tkvcp (); extern int tkvco (); extern int tkvcd (); extern int tkvcs (); </pre>	<pre> 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 multitrans; extern int ord_init;  extern errprt (); 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[];  #ifdef 0 extern int execstatus; extern int errcode;  extern OCIEnv *tpcenv; extern OCIServer *tpcsrv; extern OCIError *errhp; extern OCISvcCtx *tpcsvc; extern OCISession *tpcsur; extern OCISmt *curmtst; /* The bind and define handles for each transaction are    included in their respective header files. */  /* for stock-level transaction */  extern int w_id; extern int d_id; extern int c_id; extern int threshold; extern int low_stock;  /* for delivery transaction */  extern int del_o_id[10]; extern int carrier_id; extern int retries;  /* for order-status transaction */  extern int bylastname; extern char c_last[17]; extern char c_first[17]; extern char c_middle[3]; extern double c_balance; extern int o_id; extern text o_entry_d[20]; extern int o_carrier_id; extern int o_ol_cnt; extern int ol_supply_w_id[15]; extern int ol_i_id[15]; extern int ol_quantity[15]; extern int ol_amount[15]; extern ub4 ol_del_len[15]; extern text ol_delivery_d[15][11];  /* for payment transaction */  extern int c_w_id; extern int c_d_id; extern int h_amount; extern char w_street_1[21]; extern char w_street_2[21]; extern char w_city[21]; extern char w_state[3]; extern char w_zip[10]; extern char d_street_1[21]; extern char d_street_2[21]; extern char d_city[21]; extern char d_state[3]; extern char d_zip[10]; extern char c_street_1[21]; extern char c_street_2[21]; extern char c_city[21]; extern char c_state[3]; extern char c_zip[10]; extern char c_phone[17]; extern text c_since_d[11]; extern char c_credit[3]; extern int c_credit_lim; extern float c_discount; extern char c_data[201]; extern text h_date[20];  /* for new order transaction */  extern int nol_i_id[15]; extern int nol_supply_w_id[15]; </pre>
---	---

```

extern int nol_quantity[15];
extern int nol_quantio[15];
extern int nol_quant91[15];
extern int nol_ytdqty[15];
extern int nol_amount[15];
extern int o_all_local;
extern float w_tax;
extern float d_tax;
extern float total_amount;
extern char i_name[15][25];
extern int i_name_strlen[15];
extern ub2 i_name_strlen_len[15];
extern ub2 i_name_strlen_rocode[15];
extern ub4 i_name_strlen_cszie;
extern int s_quantity[15];
extern char brand_gen[15];
extern ub2 brand_gen_len[15];
extern ub2 brand_gen_rocode[15];
extern ub4 brand_gen_cszie;
extern int i_price[15];
extern char brand_generic[15][1];
extern int status;
extern int tracelevel;

/* Miscellaneous */
extern OCIDate cr_date;
extern OCIDate c_since;
extern OCIDate o_entry_d_base;
extern OCIDate ol_d_base[15];
#endif

#ifndef DISCARD
#define DISCARD (void)
#endif

#ifndef sword
#define sword int
#endif

#define VER7 2

#define NA -1 /* ANSI SQL NULL */
#define NLT 1 /* length for string null terminator */
#define DEADLOCK 60 /* ORA-00060: deadlock */
#define NO_DATA_FOUND 1403 /* ORA-01403: no data found */
#define NOT_SERIALIZABLE 8177 /* ORA-08177: transaction not serializable */
#define SNAPSHOT_TOO_OLD 1555 /* ORA-01555: snapshot too old */

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

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

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

#define min(x,y) (((x) < (y)) ? (x) : (y))

#define OCIERROR(errp,function) \
ocierror(_FILE_, _LINE_, (errp), (function));

#define OCIBND(stmp, bndp, errp, sqlvar, progvl, ftype) \
ocierror(_FILE_, _LINE_, (errp), \
OCIHandleAlloc((stmp), (dvoid**) &(bndp), OCI_HTYPE_BIND, 0, (dvoid**) 0); \
ocierror(_FILE_, _LINE_, (errp), \
OCIBindByName((stmp), &(bndp), (errp), \
(text *) (sqlvar), strlen((sqlvar)), \
(progvl), (progvl), (ftype), 0, 0, 0, 0, OCI_DEFAULT));

#define OCIBNDRA(stmp, bndp, errp, sqlvar, progvl, ftype, indp, alen, arcode) \
ocierror(_FILE_, _LINE_, (errp), \
OCIHandleAlloc((stmp), (dvoid**) &(bndp), OCI_HTYPE_BIND, 0, (dvoid**) 0); \
ocierror(_FILE_, _LINE_, (errp), \
OCIBindByName((stmp), &(bndp), (errp), (text *) (sqlvar), strlen((sqlvar)), \
(progvl), (progvl), (ftype), (indp), (alen), (rcode), 0, 0, OCI_DEFAULT));

#define OCIBNDRAD(stmp, bndp, errp, sqlvar, progvl, ftype, indp, ctxp, cbf_nodata, cbf_data) \
ocierror(_FILE_, _LINE_, (errp), \
OCIHandleAlloc((stmp), (dvoid**) &(bndp), OCI_HTYPE_BIND, 0, (dvoid**) 0); \
ocierror(_FILE_, _LINE_, (errp), \
OCIBindByName((stmp), &(bndp), (errp), (text *) (sqlvar), \
strlen((sqlvar)), 0, (progvl), (ftype), \
indp, 0, 0, 0, OCI_DATA_AT_EXEC); \
ocierror(_FILE_, _LINE_, (errp), \
OCIBindDynamic((bndp), (errp), (ctxp), (cbf_nodata), (cbf_data)));

#define OCIBNDR(stmp, bndp, errp, sqlvar, progvl, ftype, indp, alen, arcode) \
ocierror(_FILE_, _LINE_, (errp), \
OCIHandleAlloc((stmp), (dvoid**) &(bndp), OCI_HTYPE_BIND, 0, (dvoid**) 0); \
ocierror(_FILE_, _LINE_, (errp), \
OCIBindByName((stmp), &(bndp), (errp), (text *) (sqlvar), strlen((sqlvar)), \
(progvl), (progvl), (ftype), (indp), (alen), (rcode), 0, 0, OCI_DEFAULT));

#define OCIBNDRAA(stmp, bndp, errp, sqlvar, 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)), \
(progvl), (progvl), (ftype), (indp), (alen), (rcode), (ms), (cu), OCI_DEFAULT));

#define OCIDEFINE(stmp, dfnp, errp, pos, progvl, ftype) \
OCIDefineByPos((stmp), &(dfnp), (errp), (pos), (progvl), (progvl), \
0, 0, 0, OCI_DEFAULT);

#define OCIDDEF(stmp, dfnp, errp, pos, progvl, ftype) \
OCIHandleAlloc((stmp), (dvoid**) &(dfnp), OCI_HTYPE_DEFINE, 0, \
(dvoid**) 0); \
OCIDefineByPos((stmp), &(dfnp), (errp), (pos), (progvl), (progvl), \
(ftype), NULL, NULL, NULL, OCI_DEFAULT);

#define OCIDFNRA(stmp, dfnp, errp, pos, progvl, ftype, indp, alen, arcode) \
OCIHandleAlloc((stmp), (dvoid**) &(dfnp), OCI_HTYPE_DEFINE, 0, \
(dvoid**) 0); \
OCIDefineByPos((stmp), &(dfnp), (errp), (pos), (progvl), \
(progvl), (ftype), (indp), (alen), \
(rcode), OCI_DEFAULT);

#define OCIDFNDYN(stmp, dfnp, errp, pos, 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), (progvl), (progvl), (ftype), \
(indp), NULL, NULL, OCI_DYNAMIC_FETCH)); \
ocierror(_FILE_, _LINE_, (errp), \
OCIDefineDynamic((dfnp), (errp), (ctxp), (cbf_data)));

#ifndef 0
/* New order */
struct newinstruct {
int w_id;
int d_id;
int c_id;
int ol_i_id[15];
int ol_supply_w_id[15];
int ol_quantity[15];
};

struct newoutstruct {
int terror;
int o_id;
int o_ol_cnt;
char c_last[17];
char c_credit[3];
float c_discount;
float w_tax;
float d_tax;
char o_entry_d[20];
float total_amount;
char i_name[15][25];
int s_quantity[15];
char brand_generic[15];
float i_price[15];
float ol_amount[15];
char status[26];
int retry;
};

struct newstruct {
struct newinstruct newin;
struct newoutstruct newout;
};

/* Payment */
struct payinstruct {
int w_id;
int d_id;
int c_w_id;
int c_d_id;
int c_id;
int bylastname;
int h_amount;
char c_last[17];
};

struct payoutstruct {
int terror;
char w_street_1[21];
char w_street_2[21];
char w_city[21];
char w_state[3];
char w_zip[10];
char d_street_1[21];
char d_street_2[21];
};

```



```

char d_city[21];
char d_state[3];
char d_zip[10];
int c_id;
char c_first[17];
char c_middle[3];
char c_last[17];
char c_street_1[21];
char c_street_2[21];
char c_city[21];
char c_state[3];
char c_zip[10];
char c_phone[17];
char c_since[11];
char c_credit[3];
double c_credit_lim;
float c_discount;
double c_balance;
char c_data[201];
char h_date[20];
int retry;
};

struct paystruct {
    struct payinstruct payin;
    struct payoutstruct payout;
};

/* Order status */

struct ordinstruct {
    int w_id;
    int d_id;
    int c_id;
    int bylastname;
    char c_last[17];
};

struct ordoutstruct {
    int terror;
    int c_id;
    char c_last[17];
    char c_first[17];
    char c_middle[3];
    double c_balance;
    int o_id;
    char o_entry_d[20];
    int o_carrier_id;
    int o_ol_cnt;
    int ol_supply_w_id[15];
    int ol_i_id[15];
    int ol_quantity[15];
    float ol_amount[15];
    char ol_delivery_d[15][11];
    int retry;
};

struct ordstruct {
    struct ordinstruct ordin;
    struct ordoutstruct ordout;
};

/* Delivery */

struct delinstruct {
    int w_id;
    int o_carrier_id;
    double qtime;
    int in_timing_int;
};

struct deloutstruct {
    int terror;
    int retry;
};

struct delstruct {
    struct delinstruct delin;
    struct deloutstruct delout;
};

/* Stock level */

struct stoinstruct {
    int w_id;
    int d_id;
    int threshold;
};

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

struct stostruct {

```

```

    struct stoinstruct stoin;
    struct stooutstruct stoout;
};

#endif

#endif

                                tpcc_info.h

/*
 * $Header:
 */
/afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/server/ora8.1_mt/RCS/tpcc_info.h,v 1.1
1999/04/14 19:03:06 wenjian Exp $ 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_INFO_H
#define TPCC_INFO_H

/* New order */

struct newinstruct {
    int w_id;
    int d_id;
    int c_id;
    int ol_i_id[15];
    int ol_supply_w_id[15];
    int ol_quantity[15];
};

struct newoutstruct {
    int terror;
    int o_id;
    int o_ol_cnt;
    char c_last[17];
    char c_credit[3];
    float c_discount;
    float w_tax;
    float d_tax;
    char o_entry_d[20];
    float total_amount;
    char i_name[15][25];
    int s_quantity[15];
    char brand_generic[15];
    float i_price[15];
    float ol_amount[15];
    char status[26];
    int retry;
};

struct newstruct {
    struct newinstruct newin;
    struct newoutstruct newout;
};

/* Payment */

struct payinstruct {
    int w_id;
    int d_id;
    int c_w_id;
    int c_d_id;
    int c_id;
    int bylastname;
    int h_amount;
    char c_last[17];
};

struct payoutstruct {
    int terror;
    char w_street_1[21];
    char w_street_2[21];
    char w_city[21];
    char w_state[3];
    char w_zip[10];
    char d_street_1[21];
    char d_street_2[21];
    char d_city[21];
    char d_state[3];
    char d_zip[10];
    int c_id;
    char c_first[17];
    char c_middle[3];
    char c_last[17];
    char c_street_1[21];
    char c_street_2[21];

```

```

char c_city[21];
char c_state[3];
char c_zip[10];
char c_phone[17];
char c_since[11];
char c_credit[3];
double c_credit_lim;
float c_discount;
double c_balance;
char c_data[201];
char h_date[20];
int retry;
};

struct paystruct {
    struct payinstruct payin;
    struct payoutstruct payout;
};

/* Order status */

struct ordinstruct {
    int w_id;
    int d_id;
    int c_id;
    int bylastname;
    char c_last[17];
};

struct ordoutstruct {
    int terror;
    int c_id;
    char c_last[17];
    char c_first[17];
    char c_middle[3];
    double c_balance;
    int o_id;
    char o_entry_d[20];
    int o_carrier_id;
    int o ol_cnt;
    int ol_supply_w_id[15];
    int ol_i_id[15];
    int ol_quantity[15];
    float ol_amount[15];
    char ol_delivery_d[15][11];
    int retry;
};

struct ordstruct {
    struct ordinstruct ordin;
    struct ordoutstruct ordout;
};

/* Delivery */

struct delinstruct {
    int w_id;
    int o_carrier_id;
    double qtime;
    int in_timing_int;
};

struct deloutstruct {
    int terror;
    int retry;
};

struct delstruct {
    struct delinstruct delin;
    struct deloutstruct delout;
};

/* Stock level */

struct stoinstruct {
    int w_id;
    int d_id;
    int threshold;
};

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

struct stostruct {
    struct stoinstruct stoin;
    struct stoutstruct stoout;
};

#endif

```

## tpccflags.h

```

/* #define DMLRETNO */
#define PLSOLNO
#define DMLRETDEL
/* #define PLSQLORD */

                                tpccpl.c

#ifdef RCSID
static char *RCSid =
    "$Header: /afs/transarc.com/project/encina/rcs/test/src/benchmarks/tpcc/server/ora8.1_mt/RCS/tpccpl.c,v
    1.7 1999/05/26 16:29:59 wenjian Exp $ 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 "tpcc.h"
#ifdef TUX
#include <userlog.h>
#else
#include <stdarg.h>
#endif
#ifdef MULTI_THREADED
#include <dce/pthread.h>
#endif
#include "plora.h"

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

int proc_no;
static char *db_uid;
static char *db_pwd;

/** Delivery file infomation: Global.
 * One output file for deliveries for the server
 */
static char delivery_file_name[100];

#ifdef MULTI_THREADED
pthread_mutex_t dvry_log_lock;
#define DVRY_LOCK pthread_mutex_lock(&dvry_log_lock);
#define DVRY_UNLOCK pthread_mutex_unlock(&dvry_log_lock);
#define DVRY_LOCK_INIT pthread_mutex_init(&dvry_log_lock, pthread_mutexattr_default);
#else
#define DVRY_LOCK
#define DVRY_UNLOCK
#define DVRY_LOCK_INIT
#endif
FILE *lfp;
FILE *fopen ();

#ifdef ORA_NT
extern double dpbtimef();
#define gettime dpbtimef
#else
double gettime ();
#endif

/** Initialization of one connection */
static void initOCIhandles(ora_cn_data_t *cn_dataP, char* uid, char *pwd);

static int init_cn_data(ora_cn_data_t *dataP);

extern char oracle_home[256];

/* NewOrder Binding stuff */

#ifdef TUX
void userlog (char* fmt, ...)
{
    va_list va;
    va_start(va,fmt);
    vfprintf(stderr,fmt,va);
    va_end(va);
}
#endif

/* vmm313 void ocierror(fname, lineno, errhp, status) */
int ocierror(fname, lineno, errhp, status)
char *fname;
int lineno;
OCIError *errhp;
sword status;
{

```

```

text errbuf[512];
ub4 buflen;
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);
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);
/* vmm313 TPCexit(1); */
/* vmm313 exit(1); */
case OCI_INVALID_HANDLE:
    fprintf(stderr,"Module %s Line %d\n", fname, lineno);
    fprintf(stderr,"Error - OCI_INVALID_HANDLE\n");
    TPCexit(1);
    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);
}

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

#ifdef DEBUG
    fprintf(stderr, "tkvopen() 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);
    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:%02d0",
        day,month,year,hour,min,sec);

return;
#endif

TPCexit ()
{
if (lfp) {
fclose (lfp);
lfp = NULL;
}
}

/* clean_connection
* Called to clean a connection.
* When using pthread this is registered during pthread_create
* and called automatically by pthread when the thread exits.
*/
void clean_connection(void *ptr)
{
/* free trans specific cursor handles first and later the ora handles */
ora_cn_data_t *cn_dataP = (ora_cn_data_t *)ptr;

if (cn_dataP != NULL) {
OCI_Srvr *tpcsrv;
OCI_Sess *tpcsur;
OCI_Env *tpcenv;
OCI_Err *errhp;
OCI_SvcCtx *tpcsvc;

fprintf(stderr, "clean_connection, Freeing OCI handles\n");
tkvcndone(cn_dataP);
tkvcpdone(cn_dataP);
tkvcodone(cn_dataP);
tkvcddone(cn_dataP);
tkvcsdone(cn_dataP);

/* free OCI handles */
if (tpcsur = cn_dataP->tpcsur) {
fprintf(stderr, "free_handles> OCIHandleFree tpcsur\n");
OCIHandleFree((dvoid *)tpcsur, OCI_HTYPE_SESSION);
}
if (tpcsvc = cn_dataP->tpcsvc) {
fprintf(stderr, "free_handles> OCIHandleFree tpcsvc\n");
OCIHandleFree((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX);
}
if (errhp = cn_dataP->errhp) {
fprintf(stderr, "free_handles> OCIHandleFree errhp\n");
OCIHandleFree((dvoid *)errhp, OCI_HTYPE_ERROR);
}
if (tpcsrv = cn_dataP->tpcsrv) {
fprintf(stderr, "free_handles> OCIHandleFree tpcsrv\n");
OCIHandleFree((dvoid *)tpcsrv, OCI_HTYPE_SERVER);
}
if (tpcenv = cn_dataP->tpcenv) {
fprintf(stderr, "free_handles> OCIHandleFree tpcenv\n");
OCIHandleFree((dvoid *)tpcenv, OCI_HTYPE_ENV);
}

fprintf(stderr, "free_handles> free cn_dataP\n");
}
}

TPCinit (id, uid, pwd, dvryFileName)
int id;
char *uid;
char *pwd;
char *dvryFileName;
{
int i;
text stmbuf[100];

fprintf(stderr, "TPCinit id %d, uid %s pwd %s\n", id, uid, pwd);

DVRY_LOCK_INIT;

proc_no = id;
db_uid = (char *)calloc(strlen(uid) + 1, sizeof(char));
strcpy(db_uid, uid);
db_pwd = (char *)calloc(strlen(pwd) + 1, sizeof(char));
strcpy(db_pwd, pwd);

err_printf("dvryFileName is %s\n", dvryFileName);
sprintf(delivery_file_name, "%s.%d", dvryFileName, proc_no);

err_printf("delivery_file_name is %s\n", delivery_file_name);
#ifdef USE_ORACLE_WAY
if ((lfp = fopen (delivery_file_name, "w")) == NULL) {
#ifdef TUX
userlog ("Error in TPC-C server %d: Failed to open %s\n",
proc_no, delivery_file_name);
#else
fprintf (stderr, "Error in TPC-C server %d: Failed to open %s\n",
proc_no, delivery_file_name);
#endif
}
return (-1);
}
#endif

OCIInitialize(OCI_DEFAULT|OCI_OBJECT,(dvoid *)0,0,0); /* check tpcpl.c */

return (0);
}

static void initOCIhandles(ora_cn_data_t *cn_dataP, char *uid, char *pwd)
{
int tracelevel = 0; /* new define */
OCIDate cr_date;
text stmbuf[100];
OCI_Env *tpcenv;
OCI_Srvr *tpcsrv;
OCI_Err *errhp;
OCI_SvcCtx *tpcsvc;
OCI_Sess *tpcsur;
OCIStm *curi;

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);
OCI_SrvrAttach(tpcsrv, errhp, (text *)0, OCI_DEFAULT);
OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX, (dvoid *)tpcsrv, (ub4)0, OCI_ATTR_SERVER,
errhp);
OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsur, OCI_HTYPE_SESSION, 0, (dvoid **)0);
OCIAttrSet((dvoid *)tpcsur, OCI_HTYPE_SESSION, (dvoid *)uid,
(ub4)strlen(uid), OCI_ATTR_USERNAME, errhp);
OCIAttrSet((dvoid *)tpcsur, OCI_HTYPE_SESSION, (dvoid *)pwd, (ub4)strlen(pwd),
OCI_ATTR_PASSWORD, errhp);
OCIERROR(errhp, OCI_SessBegin(tpcsvc, errhp, tpcsur, OCI_CRED_RDBMS, OCI_DEFAULT));

OCIAttrSet(tpcsvc, OCI_HTYPE_SVCCTX, tpcsur, 0, OCI_ATTR_SESSION, errhp);

/* run all transaction in serializable mode */

OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0, (dvoid **)0);
sprintf((char *)stmbuf, SQLTXTR);
OCIStmPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT);
OCIERROR(errhp, OCIStmExecute(tpcsvc, curi, errhp, 1, 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, SQLTXTR);
OCIStmPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);
OCIERROR(errhp, OCIStmExecute(tpcsvc, curi, errhp, 1, 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, SQLTXTR);
OCIStmPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);
OCIERROR(errhp, OCIStmExecute(tpcsvc, curi, errhp, 1, 0, 0, OCI_DEFAULT));
OCIHandleFree((dvoid *)curi, OCI_HTYPE_STMT);
}

OCIERROR(errhp, OCIDateSysDate(errhp, &cr_date));

/* Store the handles just initialized in the thread slot
*/
cn_dataP->tpcenv = tpcenv;
cn_dataP->tpcsrv = tpcsrv;
cn_dataP->errhp = errhp;
cn_dataP->tpcsvc = tpcsvc;
cn_dataP->tpcsur = tpcsur;
cn_dataP->curi = curi;
}

/*
* init_cn_data
* Initializes all the transactions for a single connection
*/
static int init_cn_data(ora_cn_data_t *cnP)
{
int status;
}

```

<pre> initOCIhandles(cnP, db_uid, db_pwd);  if (status = tkvcninit (cnP)) { /* new order */     fprintf(stderr, "tkvcninit failed: %d\n", status);     TPCexit ();     return (status); }  if (status = tkvcpinit (cnP)) { /* payment */     fprintf(stderr, "tkvcpinit failed: %d\n", status);     TPCexit ();     return (status); }  if (status = tkvcoinit (cnP)) { /* order status */     fprintf(stderr, "tkvcoinit failed: %d\n", status);     TPCexit ();     return (status); }  if (status = tkvcdinit (cnP)) { /* delivery */     fprintf(stderr, "tkvcdinit failed: %d\n", status);     TPCexit ();     return (status); }  if (status = tkvesinit (cnP)) { /* stock level */     fprintf(stderr, "tkvesinit failed: %d\n", status);     TPCexit ();     return (status); } return 0; }  void *create_ora_connection() {     ora_cn_data_t *cnP = (ora_cn_data_t *)malloc(sizeof(ora_cn_data_t));     init_cn_data(cnP);     return (void *)cnP; }  ***** The Transaction Code *****  TPCnew (cnP, str) void *cnP; struct newstruct *str; {     int i;     ora_cn_data_t *cn_dataP = (ora_cn_data_t *)cnP;     global_newOrder_t *newP = cn_dataP-&gt;globals;     OCIErr *errhp = cn_dataP-&gt;errhp;      newP-&gt;w_id = str-&gt;newin.w_id;     newP-&gt;d_id = str-&gt;newin.d_id;     newP-&gt;c_id = str-&gt;newin.c_id;     for (i = 0; i &lt; 15; i++) {         newP-&gt;no_l_id[i] = str-&gt;newin.ol_i_id[i];         newP-&gt;no_l_supply_w_id[i] = str-&gt;newin.ol_supply_w_id[i];         newP-&gt;no_l_quantity[i] = str-&gt;newin.ol_quantity[i];     }     newP-&gt;retries = 0; }  vgetdate(newP-&gt;cr_date); /* OCIErr *errhp, OCIDateSysDate(errhp, &amp;newP-&gt;cr_date));  if (str-&gt;newout.terror = tkvcn (cn_dataP)) {     if (str-&gt;newout.terror != RECOVERR)         str-&gt;newout.terror = IRRECERR;     str-&gt;newout.retry = newP-&gt;retries;     return (-1); }  /* fill in date for o_entry_d from time in beginning of txn*/ cvtdmyhms(newP-&gt;cr_date, newP-&gt;o_entry_d); newP-&gt;datelen = sizeof(newP-&gt;o_entry_d); OCIErr *errhp, OCIDateToText(errhp, &amp;newP-&gt;cr_date, (text *)FULLDATE, SIZ(FULLDATE), (text *)0, 0, &amp;newP-&gt;datelen, newP-&gt;o_entry_d);  str-&gt;newout.terror = NOERR; str-&gt;newout.o_id = newP-&gt;o_id; str-&gt;newout.o_ol_cnt = newP-&gt;o_ol_cnt; strncpy (str-&gt;newout.c_last, newP-&gt;c_last, 17); strncpy (str-&gt;newout.c_credit, newP-&gt;c_credit, 3); str-&gt;newout.c_discount = newP-&gt;c_discount; str-&gt;newout.w_tax = newP-&gt;&gt;w_tax; str-&gt;newout.d_tax = newP-&gt;d_tax; strncpy (str-&gt;newout.o_entry_d, (char *)newP-&gt;o_entry_d, 20); str-&gt;newout.total_amount = newP-&gt;total_amount; for (i = 0; i &lt; newP-&gt;o_ol_cnt; i++) {     strncpy (str-&gt;newout.i_name[i], newP-&gt;i_name[i], 25);     str-&gt;newout.s_quantity[i] = newP-&gt;s_quantity[i];     str-&gt;newout.brand_generic[i] = newP-&gt;brand_gen[i];     str-&gt;newout.i_price[i] = (float)(newP-&gt;i_price[i])/100;     str-&gt;newout.ol_amount[i] = (float)(newP-&gt;no_l_amount[i])/100; } </pre>	<pre> if (newP-&gt;status)     strcpy (str-&gt;newout.status, "Item number is not valid"); else     str-&gt;newout.status[0] = '\0'; str-&gt;newout.retry = newP-&gt;retries;  return (0); }  TPCpay (cnP, str) void *cnP; struct paystruct *str; {     ora_cn_data_t *cn_dataP = (ora_cn_data_t *)cnP;     global_payment_t *payP = cn_dataP-&gt;payP;     OCIErr *errhp = cn_dataP-&gt;errhp;      payP-&gt;w_id = str-&gt;payin.w_id;     payP-&gt;d_id = str-&gt;payin.d_id;     payP-&gt;c_w_id = str-&gt;payin.c_w_id;     payP-&gt;c_d_id = str-&gt;payin.c_d_id;     payP-&gt;h_amount = str-&gt;payin.h_amount;     payP-&gt;bylastname = str-&gt;payin.bylastname;      /*     vgetdate(payP-&gt;cr_date); /*     OCIErr *errhp, OCIDateSysDate(errhp, &amp;payP-&gt;cr_date));      if (payP-&gt;bylastname) {         payP-&gt;c_id = 0;         strcpy (payP-&gt;c_last, str-&gt;payin.c_last, 17);     }     else {         payP-&gt;c_id = str-&gt;payin.c_id;         strcpy (payP-&gt;c_last, " ");     }     payP-&gt;retries = 0;      if (str-&gt;payout.terror = tkvcp (cn_dataP)) {         if (str-&gt;payout.terror != RECOVERR)             str-&gt;payout.terror = IRRECERR;         return (-1);     }      /*     cvtdmyhms(cr_date, h_date);     payP-&gt;hlen = SIZ(payP-&gt;h_date);     OCIErr *errhp, OCIDateToText(errhp, &amp;payP-&gt;cr_date,     (text *)FULLDATE, strlen(FULLDATE), (text *)0, 0, &amp;payP-&gt;hlen, payP-&gt;h_date));      /*     cvtdmy(c_since, e_since_d);     payP-&gt;syncelen = SIZ(payP-&gt;c_since_d);     OCIErr *errhp, OCIDateToText(errhp, &amp;payP-&gt;c_since,     (text *)SHORTDATE, strlen(SHORTDATE), (text *)0, 0, &amp;payP-&gt;syncelen, payP-&gt;c_since_d);      str-&gt;payout.terror = NOERR;     strncpy (str-&gt;payout.w_street_1, payP-&gt;w_street_1, 21);     strncpy (str-&gt;payout.w_street_2, payP-&gt;w_street_2, 21);     strncpy (str-&gt;payout.w_city, payP-&gt;w_city, 21);     strncpy (str-&gt;payout.w_state, payP-&gt;w_state, 3);     strncpy (str-&gt;payout.w_zip, payP-&gt;w_zip, 10);     strncpy (str-&gt;payout.d_street_1, payP-&gt;d_street_1, 21);     strncpy (str-&gt;payout.d_street_2, payP-&gt;d_street_2, 21);     strncpy (str-&gt;payout.d_city, payP-&gt;d_city, 21);     strncpy (str-&gt;payout.d_state, payP-&gt;d_state, 3);     strncpy (str-&gt;payout.d_zip, payP-&gt;d_zip, 10);     str-&gt;payout.c_id = payP-&gt;c_id;     strncpy (str-&gt;payout.c_first, payP-&gt;c_first, 17);     strncpy (str-&gt;payout.c_middle, payP-&gt;c_middle, 3);     strncpy (str-&gt;payout.c_last, payP-&gt;c_last, 17);     strncpy (str-&gt;payout.c_street_1, payP-&gt;c_street_1, 21);     strncpy (str-&gt;payout.c_street_2, payP-&gt;c_street_2, 21);     strncpy (str-&gt;payout.c_city, payP-&gt;c_city, 21);     strncpy (str-&gt;payout.c_state, payP-&gt;c_state, 3);     strncpy (str-&gt;payout.c_zip, payP-&gt;c_zip, 10);     strncpy (str-&gt;payout.c_phone, payP-&gt;c_phone, 17);     strncpy (str-&gt;payout.c_since, (char *)payP-&gt;c_since_d, 11);     strncpy (str-&gt;payout.c_credit, payP-&gt;c_credit, 3);     str-&gt;payout.c_credit_lim = (float)(payP-&gt;c_credit_lim)/100;     str-&gt;payout.c_discount = payP-&gt;c_discount;     str-&gt;payout.c_balance = (float)(payP-&gt;c_balance)/100;     strncpy (str-&gt;payout.c_data, payP-&gt;c_data, 201);     strncpy (str-&gt;payout.h_date, (char *)payP-&gt;h_date, 20);     str-&gt;payout.retry = payP-&gt;retries;      return (0); } </pre>
---	---

```

TPCord (cnP, str)
void *cnP;
struct ordstruct *str;

{
  ora_cn_data_t *cn_dataP = (ora_cn_data_t *)cnP;
  global_order_t *ordP = cn_dataP->ordP;
  int i;

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

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

  str->ordout.terror = NOERR;
  str->ordout.c_id = ordP->c_id;
  strncpy (str->ordout.c_last, ordP->c_last, 17);
  strncpy (str->ordout.c_first, ordP->c_first, 17);
  strncpy (str->ordout.c_middle, ordP->c_middle, 3);
  str->ordout.c_balance = ordP->c_balance/100;
  str->ordout.o_id = ordP->o_id;
  strncpy (str->ordout.o_entry_d, (char*)ordP->o_entry_d, 20);
  if (ordP->o_carrier_id == 11)
    str->ordout.o_carrier_id = 0;
  else
    str->ordout.o_carrier_id = ordP->o_carrier_id;
  str->ordout.o_ol_cnt = ordP->o_ol_cnt;
  for (i = 0; i < ordP->o_ol_cnt; i++) {
    ordP->ol_delivery_d[i][10] = '0';
    if (!strcmp((char*)ordP->ol_delivery_d[i], "01-01-1811"))
      strncpy((char*)ordP->ol_delivery_d[i], "NOT DELIVR", 10);
    str->ordout.ol_supply_w_id[i] = ordP->ol_supply_w_id[i];
    str->ordout.ol_i_id[i] = ordP->ol_i_id[i];
    str->ordout.ol_quantity[i] = ordP->ol_quantity[i];
    str->ordout.ol_amount[i] = (float)(ordP->ol_amount[i])/100;
    strncpy (str->ordout.ol_delivery_d[i], (char*)ordP->ol_delivery_d[i], 11);
  }
  str->ordout.retry = ordP->retries;

  return (0);
}

```

```

TPCdel (cnP, str)
void *cnP;
struct delstruct *str;

{
  ora_cn_data_t *cn_dataP = (ora_cn_data_t *)cnP;
  global_delivery_t *delP = cn_dataP->delP;
  OCIError *errhp = cn_dataP->errhp;
  double tr_end, tr_begin;
  int i, skipped;
  struct timeval cur_time;
  static int tran_cnt=0;
  int pos, len;
  int queue_time, start_time, end_time;
  char stdout_buf[1024];

  /* Open the delivery log file if needed */
  if (lfp == NULL) {
    DVRY_LOCK;
    if (lfp == NULL) {
      err_printf("TPCdel: delivery_file_name is %s\n", delivery_file_name);
      if ((lfp = fopen (delivery_file_name, "w")) == NULL) {
        fprintf (stderr, "Error in TPC-C server: Failed to open %s\n",
          delivery_file_name);
        DVRY_UNLOCK;
        return(-1);
      }
      err_printf("Opened delivery file %s\n", delivery_file_name);
    }
    DVRY_UNLOCK;
  }

  #ifndef USE_ORACLE_DVRY_FORMAT
  gettimeofday(&cur_time, NULL);
  tr_begin = (double)cur_time.tv_sec + 1.0e-6 * (double)cur_time.tv_usec;
  start_time = cur_time.tv_sec;
  #endif

  delP->w_id = str->delin.w_id;
  delP->o_carrier_id = str->delin.o_carrier_id;

```

```

delP->retries = 0;
/*
vgetdate(cr_date); */
OCIERROR(errhp, OCIDateSysDate(errhp, &delP->cr_date));

if (str->delout.terror = tkved (cn_dataP)) {
  if (str->delout.terror == DEL_ERROR)
    return DEL_ERROR;
  if (str->delout.terror != RECOVERR)
    str->delout.terror = IRRECERR;
  return (-1);
}

#ifdef USE_ORACLE_DVRY_FORMAT
tr_end = gettimeofday();
DVRY_LOCK;
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, delP->w_id, delP->o_carrier_id);
for (i = 0; i < 10; i++) {
  fprintf (lfp, " %d %d", i + 1, delP->del_o_id[i]);
  if (delP->del_o_id[i] <= 0) {
#ifdef TUX
    userlog ("DELIVERY: no new order for w_id: %d, d_id %d\n",
      delP->w_id, i + 1);
    delP->w_id, i + 1);
#else
    fprintf (stderr, "DELIVERY: no new order for w_id: %d, d_id %d\n",
      delP->w_id, i + 1);
#endif
  }
}
fprintf (lfp, " %d\n", delP->retries);

#else /* not USE_ORACLE_DVRY_FORMAT */
gettimeofday(&cur_time, NULL);
tr_end = (double)cur_time.tv_sec + 1.0e-6 * (double)cur_time.tv_usec;
end_time = cur_time.tv_sec;

queue_time = str->delin.qtime;
pos = 0;

DVRY_LOCK;
++tran_cnt;

pos += sprintf(&stdout_buf[pos], "--Tran %d Queue %.3f Start %.3f\n",
  tran_cnt, str->delin.qtime, tr_begin);
pos += sprintf(&stdout_buf[pos], "W_ID: %d, CARRIER_ID: %d",
  str->delin.w_id, str->delin.o_carrier_id);

if (str->delout.terror == DEL_ERROR) {
  pos += sprintf(&stdout_buf[pos],
    "\nDelivery transaction failed (DEL_ERROR)\n");
} else if (str->delout.terror != 0) {
  pos += sprintf(&stdout_buf[pos], "Delivery transaction failed (%d)",
    str->delout.terror);
} else {
  int skipped[10];
  int num_skipped = 0;
  for (i = 0; i < 10; i++) {
    if (delP->del_o_id[i] <= 0) {
      skipped[i] = 1;
      num_skipped++;
    } else {
      skipped[i] = 0;
    }
  }
  pos += sprintf(&stdout_buf[pos], "%d", delP->del_o_id[i]);
}
pos += sprintf(&stdout_buf[pos], "\n");
if (num_skipped > 0) {
  for (i=0; i<10; i++) {
    if (skipped[i] == 1) {
      pos += sprintf(&stdout_buf[pos],
        "D_ID %d has no new orders.\n", i+1);
    }
  }
}
}

fprintf(lfp, "%s\n", stdout_buf, tr_end);
fflush (lfp);
#endif /* USE_ORACLE_DVRY_FORMAT */

DVRY_UNLOCK;
str->delout.terror = NOERR;
str->delout.retry = delP->retries;

return (0);
}

TPCsto (cnP, str)
void *cnP;
struct stostruct *str;

{
  ora_cn_data_t *cn_dataP = (ora_cn_data_t *)cnP;
  global_stock_t *stoP = cn_dataP->stoP;

```

```

stoP->w_id = str->stoin.w_id;
stoP->d_id = str->stoin.d_id;
stoP->threshold = str->stoin.threshold;
stoP->retries = 0;

if (str->stoout.error = tkves (cn_dataP)) {
  if (str->stoout.error != RECOVERR)
    str->stoout.error = IRRECERR;
  return (-1);
}

str->stoout.error = NOERR;
str->stoout.low_stock = stoP->low_stock;
str->stoout.retry = stoP->retries;

return (0);
}

```

### views.sql

```

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 customer c, warehouse 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 district d, warehouse w
  where w.w_id = d.d_w_id
/

create or replace view stock_item
(i_id, s_w_id, i_price, i_name, i_data, s_data, s_quantity,
 s_order_cnt, s_ytd, s_remote_cnt,
 s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
 s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10)
as
select i.i_id, s_w_id, i.i_price, i.i_name, i.i_data, s_data, s_quantity,
       s_order_cnt, s_ytd, s_remote_cnt,
       s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
       s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10
   from stock s, item i
  where i.i_id = s.s_i_id
/
exit

```

# APPENDIX B: Tunable Parameters

## B.1 Database Parameters

```
# SHeader: p_run.ora 7030100.1 95/07/14 18:49:15 plai Generic<base> $ Copyr (c) 1993 Oracle
#
=====+
# Copyright (c) 1995 Oracle Corp. Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
=====+
# FILENAME
# p_run.ora
# DESCRIPTION
# Oracle parameter file for running TPC-C.
#
=====+
control_files = /dev/rvtppc_cntl1, /dev/rvtppc_cntl2, /dev/rvtppc_cntl3
disk_asynch_io = TRUE
#_huge_sga_base_address = 0x400000000
db_writer_processes = 8 # original 3
recovery_parallelism = 48
parallel_max_servers = 200
enqueue_resources = 30000
compatible = 8.1.5.0.0
db_name = tpcc
db_files = 800
db_block_size = 4096
# This will trace ORA00064 error
# event = "64 trace name errorstack, level 2";
dml_locks = 500
hash_join_enabled = FALSE
log_archive_start = FALSE
log_checkpoint_interval = 0
log_checkpoints_to_alert = TRUE
gc_releasable_locks = 0
max_rollback_segments = 800
max_dump_file_size = 3000
open_cursors = 1000
sessions = 1500
transactions = 3000
distributed_transactions = 0
transactions_per_rollback_segment = 1
rollback_segments = (t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14,t15,\
t16,t17,t18,t19,t20,t21,t22,t23,t24,t25,t26,t27,t28,t29,t30,\
t31,t32,t33,t34,t35,t36,t37,t38,t39,t40,t41,t42,t43,t44,t45,\
t46,t47,t48,t49,t50,t51,t52,t53,t54,t55,t56,t57,t58,t59,t60,\
t61,t62,t63,t64,t65,t66,t67,t68,t69,t70,t71,t72,t73,t74,t75,\
t76,t77,t78,t79,t80,t81,t82,t83,t84,t85,t86,t87,t88,t89,t90,\
t91,t92,t93,t94,t95,t96,t97,t98,t99,t100,t101,t102,t103,t104,\
t105,t106,t107,t108,t109,t110,t111,t112,t113,t114,t115,t116,\
t117,t118,t119,t120,t121,t122,t123,t124,t125,t126,t127,t128,\
t129,t130,t131,t132,t133,t134,t135,t136,t137,t138,t139,t140,\
t141,t142,t143,t144,t145,t146,t147,t148,t149,t150,t151,t152,\
t153,t154,t155,t156,t157,t158,t159,t160,t161,t162,t163,t164,t165,t166,\
t167,t168,t169,t170,t171,t172,t173,t174,t175,t176,t177,t178,t179,t180,\
t181,t182,t183,t184,t185,t186,t187,t188,t189,t190,t191,t192,t193,t194,\
t195,t196,t197,t198,t199,t200,t201,t202,t203,t204,t205,t206,t207,t208,\
t209,t210,t211,t212,t213,t214,t215,t216,t217,t218,t219,t220,t221,t222,\
t223,t224,t225,t226,t227,t228,t229,t230,t231,t232,t233,t234,t235,t236,\
t237,t238,t239,t240,t241,t242,t243,t244,t245,t246,t247,t248,t249,t250,\
t251,t252,t253,t254,t255,t256,t257,t258,t259,t260,t261,t262,t263,t264,\
t265,t266,t267,t268,t269,t270,t271,t272,t273,t274,t275,t276,t277,t278,\
t279,t280,t281,t282,t283,t284,t285,t286,t287,t288,t289,t290,t291,t292,\
t293,t294,t295,t296,t297,t298,t299,t300,t301,t302,t303,t304,t305,t306,\
t307,t308,t309,t310,t311,t312,t313,t314,t315,t316,t317,t318,t319,t320,\
t321,t322,t323,t324,t325,t326,t327,t328,t329,t330,t331,t332,t333,t334,\
t335,t336,t337,t338,t339,t340,t341,t342,t343,t344,t345,t346,t347,t348,\
t349,t350,t351,t352,t353,t354,t355,t356,t357,t358,t359,t360,t361,t362,\
t363,t364,t365,t366,t367,t368,t369,t370,t371,t372,t373,t374,t375,t376,\
t377,t378,t379,t380,t381,t382,t383,t384,t385,t386,t387,t388,t389,t390,\
t391,t392,t393,t394,t395,t396,t397,t398,t399,t400,t401,t402,t403,t404,t405,\
t406,t407,t408,t409,t410,t411,t412,t413,t414,t415,t416,t417,t418,t419,t420,\
t421,t422,t423,t424,t425,t426,t427,t428,t429,t430,t431,t432,t433,t434,\
t435,t436,t437,t438,t439,t440,t441,t442,t443,t444,t445,t446,t447,t448,\
t449,t450,t451,t452,t453,t454,t455,t456,t457,t458,t459,t460,t461,t462,\
t463,t464,t465,t466,t467,t468,t469,t470,t471,t472,t473,t474,t475,t476,\
t477,t478,t479,t480,t481,t482,t483,t484,t485,t486,t487,t488,t489,t490,\
t491,t492,t493,t494,t495,t496,t497,t498,t499,t500,t501,t502,t503,t504,t505,\
t506,t507,t508,t509,t510,t511,t512,t513,t514,t515,t516,t517,t518,t519,t520,\
t521,t522,t523,t524,t525,t526,t527,t528,t529,t530,t531,t532,t533,t534,\
t535,t536,t537,t538,t539,t540,t541,t542,t543,t544,t545,t546,t547,t548,\
t549,t550,t551,t552,t553,t554,t555,t556,t557,t558,t559,t560,t561,t562,\
t563,t564,t565,t566,t567,t568,t569,t570,t571,t572,t573,t574,t575,t576,\
t577,t578,t579,t580,t581,t582,t583,t584,t585,t586,t587,t588,t589,t590,\
t591,t592,t593,t594,t595,t596,t597,t598,t599,t600,t601,t602,t603,t604,t605,\
t606,t607,t608,t609,t610,t611,t612,t613,t614,t615,t616,t617,t618,t619,t620,\
```

```
t621,t622,t623,t624,t625,t626,t627,t628,t629,t630,t631,t632,t633,t634,\
t635,t636,t637,t638,t639,t640,t641,t642,t643,t644,t645,t646,t647,t648,\
t649,t650,t651,t652,t653,t654,t655,t656,t657,t658,t659,t660,t661,t662,\
t663,t664,t665,t666,t667,t668,t669,t670,t671,t672,t673,t674,t675,t676,\
t677,t678,t679,t680,t681,t682,t683,t684,t685,t686,t687,t688,t689,t690,\
t691,t692,t693,t694,t695,t696,t697,t698,t699,t700)
```

```
cursor_space_for_time = TRUE
replication_dependency_tracking = FALSE
processes = 1000
shared_pool_size = 100000000
shared_pool_reserved_size = 20000000
db_block_lru_latches = 120
buffer_pool_keep = (buffers:10990952,lru_latches:40) # 76% of db_block_buffers
buffer_pool_recycle = (buffers:34782,lru_latches:40)
log_buffer = 25165824 # was 20971520
db_block_buffers = 14417920 #55GB
_enable_list_io = FALSE
_spin_count = 20000
_disable_incremental_checkpoints = TRUE
```

```
lock_sga = TRUE
# DB Writer important parameters
fast_start_io_target = 0
_db_aging_stay_count = 1
db_block_max_dirty_target = 7208960 # db_block_buffers/2
_db_block_hash_buckets = 28835840 # db_block_buffers * 2
_db_percent_hot_keep = 50
_db_percent_hot_recycle = 50
parallel_automatic_tuning = TRUE
#_dump_MTTTR_to_trace = TRUE
transaction_auditing=false
db_file_multiblock_read_count=1
_db_file_noncontig_mblock_read_count=1
_log_simultaneous_copies=72
```

## B.2 Transaction Monitor Parameters

### tpccrc

```
CellLogVolume ecmlog
CellDataVolume ecmdata
NodeLogVolume enmlog
TpccApplicationDirectory /home/encina
TpccDbServer oratpcc
StatsFrequency 10
Version 1.0
Servers:delivery PAS 1 Threads 3 Name del IFS ---D- Dvry 2
Servers:online PAS 30 Threads 1 Name onl IFS NPO-S Dvry 0
```

## B.3 AIX Parameters

### RISC SYSTEM/6000 MODEL S80

#### OS PARAMETERS

keylock	normal	State of system keylock at boot time	False
maxbuf	20	Maximum number of pages in block I/O BUFFER CACHE	True
maxmbuf	0	Maximum Kbytes of real memory allowed for MBUFS	True
maxuproc	81920	Maximum number of PROCESSES allowed per user	True
autorestart	false	Automatically REBOOT system after a crash	True
iostat	false	Continuously maintain DISK I/O history	True
realmem	67108864	Amount of usable physical memory in Kbytes	False
consonlog	enable	System Console Login	False
fwversion	IBM,19990621 (B)	Firmware version and revision levels	False
minpout	0	HIGH water mark for pending write I/Os per file	True
minpout	0	LOW water mark for pending write I/Os per file	True
fullcore	false	Enable full CORE dump	True
pre430core	false	Use pre-430 style CORE dump	True
rtasversion	1	Open Firmware RTAS version	False
modelname	IBM,vpd property exists but TM field does not exist	Machine name	False
systemid	IBM,vpd property exists but SE field does not exist	Hardware system identifier	False
boottype	disk		N/A
SW_dist_intr	false	Enable SW distribution of interrupts	True



## Appendix C: Database Setup Code

### C.1 Database Creation Scripts

#### addfile.sh

```
#!/bin/ksh
#
# $Header: addfile.sh 7030100.1 96/05/02 10:30:04 plai Generic<base> $ Copyr (c) 1995 Oracle
#
#====#+
# Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#====#+
# FILENAME
# addfile.sh
# DESCRIPTION
# Add datafile to a tablespace.
# USAGE
# addfile.sh <tablespace> <data file> <size>
#====#+
FILE='basename $2'

if [ -d ./outdir ]
then
echo `date` > ./outdir/${FILE}.addf
fi

svrmgrl <<!
connect internal
set echo on
alter tablespace $1 add datafile '$2' size $3 reuse;
exit;
!

if [ -d ./outdir ]
then
echo `date` >> ./outdir/${FILE}.addf
fi
```

#### addfs.sh

```
#!/bin/ksh
#
# Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#====#+
# FILENAME
# addfs.sh
# DESCRIPTION
# Add tablespace to database.
# USAGE
# addfs.sh <tablespace> <data file> <size> <# of files>
#====#+

setenv

tablespace=$1
datafile=$2
size=$3
nfiles=$4
let total=$nfiles+1

i=2
while [ $i -le $total ]
do
j=1
while [ $j -le 20 ]
do
if [ $i -le $total ]
then
echo "addfile.sh $tablespace ${datafile}$i $size"
addfile.sh $tablespace ${datafile}$i $size &
let i=$i+1
let j=$j+1
else
j=21
fi
done
wait
done
```

#### addroll.sh

```
#!/bin/ksh
#====#+
# Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#====#+
# FILENAME
# addts.sh
# DESCRIPTION
# Add tablespace to database.
# USAGE
# addts.sh <tablespace> <data file> <size>
#====#+
echo 'ORACLE_HOME=' $ORACLE_HOME
echo 'ORACLE_SID=' $ORACLE_SID

FILE='basename $1'

if [ -d ./outdir ]
then
echo `date` > ./outdir/${FILE}.addts
fi

# create tablespace roll datafile '$1' size $2 reuse extent management local uniform size 40K nologging ;

svrmgrl <<!
connect internal
create tablespace roll datafile '$1' size $2 reuse;

exit;
!

if [ -d ./outdir ]
then
echo `date` >> ./outdir/${FILE}.addts
fi
```

#### addts.sh

```
#!/bin/ksh
#====#+
# Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#====#+
# FILENAME
# addts.sh
# DESCRIPTION
# Add tablespace to database.
# USAGE
# addts.sh <tablespace> <data file> <size>
#====#+
FILE='basename $2'

if [ -d ./outdir ]
then
echo `date` > ./outdir/${FILE}.addts
fi

svrmgrl <<!
connect internal
set echo on
create tablespace $1 datafile '$2' size $3 reuse extent management local uniform size $4 nologging ;

exit;
!

if [ -d ./outdir ]
then
echo `date` >> ./outdir/${FILE}.addts
fi
```

#### alter\_temp.sh

```
#!/bin/ksh

# alter_temp 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
#====#+
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#====#+
# NAME
# alter_temp
# DESCRIPTION
# Usage: alter_temp.sh [options]
#====#+
#
```

```

setenv

sqlplus system/manager <<!
alter user tpcc temporary tablespace temp;
quit;
!

svrmgrl <<!
connect internal
alter tablespace temp
  default storage (initial 500M next 500M pctincrease 0);
exit;
!

benchdb.sh

#!/bin/ksh
#
# benchdb.sh 8030100 98/7/7 15:45 vmakhija
# Copyr (c) 1998 Oracle
#
#
=====
# Copyright (c) 1997 Oracle Corp. Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
=====
# FILENAME
# benchdb.sh
# DESCRIPTION
# Usage: benchdb.sh [options]
# -n do not create new tpcc database
# -c do not run catalog scripts
#
#
setenv

while [ "$#" != "0" ]
do
  case $1 in
    -n) shift
      NO_CREATE="y"
      ;;
    -c) shift
      NO_CAT="y"
      ;;
    *) echo "Bad arg: $1"
      exit 1;
      ;;
  esac
done

#
# Create database if NO_CREATE unset
#
if [ "$NO_CREATE" = "" ]
then
svrmgrl <<!
set echo on
connect internal
startup pfile=$TPCC_ADMIN/p_create.ora nomount
create database tpcc controlfile reuse maxdatafiles 700
datafile '/dev/rlvsys1' size 2700M reuse
logfile '/dev/rlvlog1' size 4000M reuse,
'/dev/rlvlog2' size 4000M reuse;
exit
!
#
# Create more rollback segments
#
svrmgrl <<!
connect internal
create rollback segment s1 storage (initial 200k minextents 2 next 200k);
create rollback segment s2 storage (initial 200k minextents 2 next 200k);
create rollback segment s3 storage (initial 200k minextents 2 next 200k);
create rollback segment s4 storage (initial 200k minextents 2 next 200k);
create rollback segment s5 storage (initial 200k minextents 2 next 200k);
create rollback segment s6 storage (initial 200k minextents 2 next 200k);
create rollback segment s7 storage (initial 200k minextents 2 next 200k);
create rollback segment s8 storage (initial 200k minextents 2 next 200k);
create rollback segment s9 storage (initial 200k minextents 2 next 200k);
create rollback segment s10 storage (initial 200k minextents 2 next 200k);
create rollback segment s11 storage (initial 200k minextents 2 next 200k);
create rollback segment s12 storage (initial 200k minextents 2 next 200k);
create rollback segment s13 storage (initial 200k minextents 2 next 200k);
create rollback segment s14 storage (initial 200k minextents 2 next 200k);
create rollback segment s15 storage (initial 200k minextents 2 next 200k);
create rollback segment s16 storage (initial 200k minextents 2 next 200k);
create rollback segment s17 storage (initial 200k minextents 2 next 200k);
create rollback segment s18 storage (initial 200k minextents 2 next 200k);
create rollback segment s19 storage (initial 200k minextents 2 next 200k);
create rollback segment s20 storage (initial 200k minextents 2 next 200k);

create rollback segment s21 storage (initial 200k minextents 2 next 200k);
create rollback segment s22 storage (initial 200k minextents 2 next 200k);
create rollback segment s23 storage (initial 200k minextents 2 next 200k);
create rollback segment s24 storage (initial 200k minextents 2 next 200k);
create rollback segment s25 storage (initial 200k minextents 2 next 200k);
create rollback segment s26 storage (initial 200k minextents 2 next 200k);
create rollback segment s27 storage (initial 200k minextents 2 next 200k);
create rollback segment s28 storage (initial 200k minextents 2 next 200k);
create rollback segment s29 storage (initial 200k minextents 2 next 200k);
create rollback segment s30 storage (initial 200k minextents 2 next 200k);
shutdown;
exit;
!
fi
#
# Startup database with params file that includes new rollback segments
#
svrmgrl <<!
connect internal
startup pfile=$TPCC_ADMIN/p_build.ora;
connect system/manager
create tablespace temp datafile
  '/dev/rlvtemp1' size 1999M reuse;
exit;
!
#
# Add tablespaces in parallel
#
addroll.sh /dev/rlvroll1 800M &
addts.sh hist /dev/rlvhist1 3005M 150M &
addts.sh ware /dev/rlvware 1000M 1M &
addts.sh cust /dev/rlvcust1 2301M 1150M &
addts.sh items /dev/rlvitem 201M 20M &
addts.sh ord /dev/rlvord1 1510M 150M &
addts.sh nord /dev/rlvnord1 1435M 143M &
addts.sh ordl /dev/rlvordl1 2875M 143M &
addts.sh stocks /dev/rlvstock1 2301M 1150M &
addts.sh icust1 /dev/rlvi1cust1 3005M 150M &
addts.sh icust2 /dev/rlvi2cust1 1615M 100M &
addts.sh istk /dev/rlvistk1 2940M 146M &
addts.sh iord1 /dev/rlvi1ord1 1820M 90M &
addts.sh iord2 /dev/rlvi2ord1 2015M 100M &
wait
#
# Add datafiles to tablespaces in parallel
#
adddfs.sh roll /dev/rlvroll 800M 1
wait
adddfs.sh hist /dev/rlvhist 3005M 9
wait
adddfs.sh cust /dev/rlvcust 2301M 143
wait
adddfs.sh ord /dev/rlvord 1510M 13
wait
adddfs.sh nord /dev/rlvnord 1435M 3
wait
adddfs.sh ordl /dev/rlvordl 2875M 139
wait
adddfs.sh stocks /dev/rlvstock 2301M 199
wait
adddfs.sh icust1 /dev/rlvi1cust 3005M 4
wait
adddfs.sh icust2 /dev/rlvi2cust 1615M 11
wait
adddfs.sh istk /dev/rlvistk 2940M 14
wait
adddfs.sh iord1 /dev/rlvi1ord 1820M 7
wait
adddfs.sh iord2 /dev/rlvi2ord 2015M 11
wait
adddfs.sh temp /dev/rlvtemp 1999M 49
wait
#
# run catalog if NO_CAT unset
#
if [ "$NO_CAT" = "" ]
then
svrmgrl <<!
set echo off;

```

```
connect sys/change_on_install;
@?/rdbs/admin/catalog;
@?/rdbs/admin/catproc;
@?/rdbs/admin/catparr;
exit;
```

## benchsetup.sh

```
#!/bin/ksh
```

```
# benchsetup 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
```

```
=====+
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
=====+
```

```
# NAME
# benchsetup
# DESCRIPTION
# Usage: benchsetup.sh [options]
```

```
#
# setenv
STEP=0
START=0
END=0
CONTINUE=1
PROGNAME=$0
```

```
function usage {
echo ""
echo "Usage: $PROGNAME [<start> <stop>] [<start>] [-step <stepno>]"
echo "    [<start> <stop>] - allows user to run a specified"
echo "                    range of steps."
echo "    [<start>]         - runs from step number <start> till"
echo "                    the end of the script."
echo "    [-step <stepno>] - runs only step number <stepno> and"
echo "                    then stops."
echo ""
echo "    STEP  FUNCTION"
echo "-----"
echo "    0    Create DB."
echo "    1    Create user tpcc."
echo "    2    Create warehouse table."
echo "    3    Create district table."
echo "    4    Create history table."
echo "    5    Create Orders table."
echo "    6    Create New-order table."
echo "    7    Create Orderline table."
echo "    8    Create Item table."
echo "    9    Create Customer table."
echo "    10   Create Stock table."
echo "    11   Create rollback segments."
echo "    12   Load New-order."
echo "    13   Load History."
echo "    14   Load Order/Orderline."
echo "    15   Load Warehouse."
echo "    16   Load District."
echo "    17   Load Item."
echo "    18   Load Customer."
echo "    19   Load Stock."
echo "    20   Alter temp space."
echo "    21   Create Warehouse index."
echo "    22   Create District index."
echo "    23   Create Item index."
echo "    24   Create Customer index."
echo "    25   Create Customer2 index."
echo "    26   Create Stock index."
echo "    27   Create Orders index."
echo "    28   Create Orders2 index."
echo "    29   Create New-order index."
echo "    30   Create Orderline index."
echo "    31   Re-alter temp space."
echo "    32   Analyze."
echo "    33   Create TPC-C reports tables."
echo "    34   Create stored procs."
echo "    35   Space rpts / etc."
echo "    36   Alter extents and Lock tables."
echo "    37   Run catalog scripts."
echo "    38   Shutdown database."
echo "-----"
}
```

```
exit 1;
```

```
function runnable {
if [ -a "./stop" ]
then
exit 1;
fi
```

```
if [ $STEP -ge $START ]
then
if [ $STEP -le $END ]
then
STEP=`expr $STEP + 1`;
return 0;
else
if [ $CONTINUE = 0 ]
then
STEP=`expr $STEP + 1`;
return 0;
fi
fi
fi
STEP=`expr $STEP + 1`;
return 1;
}
```

```
case $# in
0) usage
CONTINUE=0
;;
1) case $1 in
-h) usage
;;
*) START=$1
CONTINUE=0
;;
esac
;;
2) case $1 in
-step) shift
START=$1
END=$1
CONTINUE=1
;;
*) START=$1
shift
END=$1
CONTINUE=1
;;
esac
;;
*) usage
;;
esac
```

```
if [ ! -d $BUILD_HOME ]
then
mkdir $BUILD_HOME
fi
```

```
if [ ! -d $LOAD_SCRIPTS ]
then
mkdir $LOAD_SCRIPTS
fi
```

```
if [ ! -d $OUTDIR ]
then
mkdir $OUTDIR
fi
```

```
date

if runnable
then
${LOAD_SCRIPTS}/benchdb.sh > ${OUTDIR}/benchdb.out 2>&1
echo "Switching Logs ..."
${TPCC_UTILS}/switchlog.sh >> ${OUTDIR}/switchlog.out 2>&1
fi
```

```
if runnable
then
echo "Creating user ..."
${LOAD_SCRIPTS}/create_user.sh > ${OUTDIR}/create_user.out 2>&1
fi
```

```
if runnable
then
echo "Creating warehouse ..."
${LOAD_SCRIPTS}/create_ware.sh > ${OUTDIR}/create_ware.out 2>&1
fi
```

```
if runnable
then
echo "Creating district ..."
${LOAD_SCRIPTS}/create_dist.sh > ${OUTDIR}/create_dist.out 2>&1
fi
```

```
if runnable
then
echo "Creating history ..."
${LOAD_SCRIPTS}/create_hist.sh > ${OUTDIR}/create_hist.out 2>&1
fi
```

```
if runnable
```

```

then
echo "Creating orders ..."
${LOAD_SCRIPTS}/create_ordr.sh > ${OUTDIR}/create_ordr.out 2>&1
fi

if runnable
then
echo "Creating new-order ..."
${LOAD_SCRIPTS}/create_nord.sh > ${OUTDIR}/create_nord.out 2>&1
fi

if runnable
then
echo "Creating order-line ..."
${LOAD_SCRIPTS}/create_ordl.sh > ${OUTDIR}/create_ordl.out 2>&1
fi

if runnable
then
echo "Creating item ..."
${LOAD_SCRIPTS}/create_item.sh > ${OUTDIR}/create_item.out 2>&1
fi

if runnable
then
echo "Creating customer ..."
${LOAD_SCRIPTS}/create_cust.sh > ${OUTDIR}/create_cust.out 2>&1 &
fi

if runnable
then
echo "Creating stock ..."
${LOAD_SCRIPTS}/create_stok.sh > ${OUTDIR}/create_stok.out 2>&1 &
fi

wait

if runnable
then
echo "Creating rollback segment ..."
${LOAD_SCRIPTS}/tpcc_ro1.sh > ${OUTDIR}/tpcc_ro1.out 2>&1 &
fi

echo "Switching Logs ..."
${TPCC_UTILS}/switchlog.sh >> ${OUTDIR}/switchlog.out 2>&1

if runnable
then
echo "Loading new-order ..."
${LOAD_SCRIPTS}/load_nord.sh > ${OUTDIR}/load_nord.out 2>&1
fi

if runnable
then
echo "Loading history ..."
${LOAD_SCRIPTS}/load_hist.sh > ${OUTDIR}/load_hist.out 2>&1
fi

if runnable
then
echo "Loading orders and order-line ..."
${LOAD_SCRIPTS}/load_ordr.sh > ${OUTDIR}/load_ordr.out 2>&1
fi

echo "Switching Logs ..."
${TPCC_UTILS}/switchlog.sh >> ${OUTDIR}/switchlog.out 2>&1

if runnable
then
echo "Loading warehouse ..."
${LOAD_SCRIPTS}/load_ware.sh > ${OUTDIR}/load_ware.out 2>&1
fi

if runnable
then
echo "Loading district ..."
${LOAD_SCRIPTS}/load_dist.sh > ${OUTDIR}/load_dist.out 2>&1
fi

if runnable
then
echo "Loading item ..."
${LOAD_SCRIPTS}/load_item.sh > ${OUTDIR}/load_item.out 2>&1
fi

if runnable
then
echo "Loading customer ..."
${LOAD_SCRIPTS}/load_cust.sh > ${OUTDIR}/load_cust.out 2>&1
fi

if runnable
then
echo "Loading stock ..."
${LOAD_SCRIPTS}/load_stok.sh > ${OUTDIR}/load_stok.out 2>&1
fi

wait

```

```

echo "Switching Logs ..."
${TPCC_UTILS}/switchlog.sh >> ${OUTDIR}/switchlog.out 2>&1

if runnable
then
echo "Alter temp ..."
${LOAD_SCRIPTS}/alter_temp.sh > ${OUTDIR}/alter_temp.out 2>&1
fi

if runnable
then
echo "Creating warehouse index ..."
${LOAD_SCRIPTS}/create_iware.sh > ${OUTDIR}/create_iware.out 2>&1
fi

if runnable
then
echo "Creating district index ..."
${LOAD_SCRIPTS}/create_idist.sh > ${OUTDIR}/create_idist.out 2>&1
fi

if runnable
then
echo "Creating item index ..."
${LOAD_SCRIPTS}/create_iitem.sh > ${OUTDIR}/create_iitem.out 2>&1
fi

if runnable
then
echo "Creating customer index ..."
${LOAD_SCRIPTS}/create_icust.sh > ${OUTDIR}/create_icust.out 2>&1
fi

if runnable
then
echo "Creating customer2 index ..."
${LOAD_SCRIPTS}/create_icust2.sh > ${OUTDIR}/create_icust2.out 2>&1
fi

if runnable
then
echo "Creating stock index ..."
${LOAD_SCRIPTS}/create_istok.sh > ${OUTDIR}/create_istok.out 2>&1
fi

if runnable
then
echo "Creating orders index ..."
${LOAD_SCRIPTS}/create_iordr.sh > ${OUTDIR}/create_iordr.out 2>&1
fi

if runnable
then
echo "Creating orders2 index ..."
${LOAD_SCRIPTS}/create_iordr2.sh > ${OUTDIR}/create_iordr2.out 2>&1
fi

if runnable
then
echo "No need to create inord"
#${LOAD_SCRIPTS}/create_inord.sh > ${OUTDIR}/create_inord.out 2>&1
fi

if runnable
then
echo "No need to create iordl"
#${LOAD_SCRIPTS}/create_iordl.sh > ${OUTDIR}/create_iordl.out 2>&1
fi

if runnable
then
echo "Re-alter temp ..."
${LOAD_SCRIPTS}/realter_temp.sh > ${OUTDIR}/realter_temp.out 2>&1
fi

if runnable
then
echo "Analyze ..."
sqlplus tpcc/tpcc @$TPCC_SQL/tpcc_ana > ${OUTDIR}/tpcc_ana.out 2>&1
fi

if runnable
then
echo "Creating report tables ..."
${LOAD_SCRIPTS}/tpcc_reports.sh > ${OUTDIR}/tpcc_reports.out 2>&1
fi

if runnable
then
echo "Creating stored procs ..."
${LOAD_SCRIPTS}/tpcc_stored_proc.sh > ${OUTDIR}/tpcc_stored_prod.out 2>&1
$TPCC_UTILS/create_cache_views.sh > ${OUTDIR}/create_cache_views.out 2>&1
fi

if runnable
then
echo "Space rpts / etc. ..."

```

```

${LOAD_SCRIPTS}/tpcc_misc.sh > ${OUTDIR}/tpcc_misc.out 2>&1
fi

if runnable
then
echo "Alter extents and lock ..."
${LOAD_SCRIPTS}/alter.sh > ${OUTDIR}/alter.out 2>&1
${LOAD_SCRIPTS}/altundef.sh > ${OUTDIR}/altundef.out 2>&1
${TPCC_UTILS}/dml.sh > ${OUTDIR}/dml.out 2>&1
fi

```

```

if runnable
then
echo "Running catalog scripts ..."
${LOAD_SCRIPTS}/cat.sh > ${OUTDIR}/cat.out 2>&1
fi

```

```

if runnable
then
svrmgrl <<!
connect internal;
alter system switch logfile;
alter system switch logfile;
shutdown;
exit;
!
fi

```

```
date
```

### cat.sh

```

#!/bin/ksh

# benchsetup 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#

```

```

=====
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
=====

```

```

# NAME
# benchsetup
# DESCRIPTION
# Usage: benchsetup.sh [options]
#

```

```
.setenv
```

```

svrmgrl <<!
set echo off;
connect sys/change_on_install;
@?/rdms/admin/catparr;
exit;
!

```

### create\_cust.sh

```

#!/bin/ksh

#
# load <obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#

```

```

=====
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
=====

```

```

# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#

```

```
.setenv
```

```
sqlplus tpcc/tpcc @cust
```

### create\_dist.sh

```

#!/bin/ksh

#
# load <obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#

```

```

=====
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |

```

```

# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
=====
# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#
#
.setenv

sqlplus tpcc/tpcc @dist

```

### create\_hist.sh

```

#!/bin/ksh

#
# load <obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#

```

```

=====
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
=====

```

```

# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#

```

```
.setenv
```

```
sqlplus tpcc/tpcc @hist
```

### create\_icust.sh

```

#!/bin/ksh

#
# load <obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#

```

```

=====
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
=====

```

```

# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#

```

```
.setenv
```

```
sqlplus tpcc/tpcc @icust
```

### create\_icust2.sh

```

#!/bin/ksh

#
# load <obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#

```

```

=====
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
=====

```

```

# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#

```

```
.setenv
```

```
sqlplus tpcc/tpcc @icust2
```

### create\_idist.sh

```

#!/bin/ksh

```

```

#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
=====
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#
# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#
#
# setenv
#
# sqlplus tpcc/tpcc @idist

create_iitem.sh

#!/bin/ksh
#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
=====
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#
# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#
#
# setenv
#
# sqlplus tpcc/tpcc @iitem

create_iordr.sh

#!/bin/ksh
#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
=====
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#
# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#
#
# setenv
#
# sqlplus tpcc/tpcc @iordr

create_iordr2.sh

#!/bin/ksh
#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
=====
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#
# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#
#
# setenv
#
# sqlplus tpcc/tpcc @iordr2

```

```

create_istok.sh

#!/bin/ksh
#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
=====
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#
# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#
#
# setenv
#
# sqlplus tpcc/tpcc @istok

create.item.sh

#!/bin/ksh
#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
=====
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#
# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#
#
# setenv
#
# sqlplus tpcc/tpcc @item

create_iware.sh

#!/bin/ksh
#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
=====
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#
# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#
#
# setenv
#
# sqlplus tpcc/tpcc @iware

create_nordr.sh

#!/bin/ksh
#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
=====
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#
# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#
#
# setenv

```

```
sqlplus tpcc/tpcc @nord
```

### create\_ordl.sh

```
#!/bin/ksh
#
# load <obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#=====#
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#=====#
# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#
#
# setenv
```

```
sqlplus tpcc/tpcc @ordl
```

### create\_orldr.sh

```
#!/bin/ksh
#
# load <obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#=====#
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#=====#
# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#
#
# setenv
```

```
sqlplus tpcc/tpcc @ordr
```

### create\_stok.sh

```
#!/bin/ksh
#
# load <obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#=====#
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#=====#
# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#
#
# setenv
```

```
sqlplus tpcc/tpcc @stok
```

### create\_user.sh

```
#!/bin/ksh
#
# load <obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#=====#
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#=====#
# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#
#
```

```
. setenv
```

```
svrmgrl <<!
```

```
rem
rem=====#
rem Copyright (c) 1997 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem=====#
rem FILENAME
rem tpcc_user.sql
rem DESCRIPTION
rem Create user for TPC-C database.
rem=====#
rem
```

```
rem
rem Create TPCC userid and connect to it.
rem
connect internal;
grant connect,resource,unlimited tablespace to tpcc identified by tpcc;
alter user tpcc temporary tablespace temp;
connect tpcc/tpcc;
exit;
!
```

### create\_ware.sh

```
#!/bin/ksh
#
# load <obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#=====#
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#=====#
# FILENAME
# create <obj>.sh
# DESCRIPTION
# Usage: create <obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#
#
# setenv
```

```
sqlplus tpcc/tpcc @ware
```

### dml.sh

```
#
# $Header: dml.sh 7030100.1 96/05/02 10:22:52 plai Generic<base> $ Copyr (c) 1995 Oracle
#
#=====#
# Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#=====#
# FILENAME
# dml.sh
# DESCRIPTION
# Disable table locks for TPC-C tables.
# USAGE
# dml.sh
#=====#
```

```
sqlplus tpcc/tpcc <<!
alter table warehouse disable table lock;
alter table district disable table lock;
alter table customer disable table lock;
alter table history disable table lock;
alter table item disable table lock;
alter table stock disable table lock;
alter table orders disable table lock;
alter table new_order disable table lock;
alter table order_line disable table lock;
quit;
!
```

### lload\_cust.sh

```
#!/bin/ksh
#
# load <obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
#=====#
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#=====#
# NAME
```

```

# load <obj>.sh
# DESCRIPTION
# Usage: load_<obj>.sh [options]
=====
#
setenv

#
# Load customer table (in parallel with loading stock table)
#
I=1
SW=1
INC=21
let EW=$SW+$INC-1
J=1
X=$J
while [ $J -le 10 ]
do
let X=$J*50
while [ $I -le $X ]
do
tpccload -M $MULT -c -b $SW -e $EW > ${OUTDIR}/cust${I}.out 2>&1 &
I='expr $I + 1'
SW='expr $SW + $INC'
EW='expr $EW + $INC'
done
wait
J='expr $J + 1'
done
INC=20
let EW=$SW+$INC-1
J=1
while [ $J -le 10 ]
do
tpccload -M $MULT -c -b $SW -e $EW > ${OUTDIR}/cust${I}.out 2>&1 &
I='expr $I + 1'
SW='expr $SW + $INC'
EW='expr $EW + $INC'
J='expr $J + 1'
done
done

Wait

load_dist.sh

#!/bin/ksh

# load <obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
#
=====+
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
=====+
# NAME
# load <obj>.sh
# DESCRIPTION
# Usage: load_<obj>.sh [options]
#
#
setenv

tpccload -M $MULT -d

load_hist.sh

#!/bin/ksh

#
# load <obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#
=====+
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
=====+
# FILENAME
# pload.sh
# DESCRIPTION
# Usage: load_<obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#
#
setenv

if echo "`c`" | grep c >/dev/null 2>&1; then
N='n'
else
C='c'
fi
export N C

```

```

while [ "$#" != "0" ]
do
case $1 in
-mu) shift
if [ "$1" != "" ]
then
MULT=$1
shift
fi
;;
-nd) shift
NO_DB="y"
;;
-nt) shift
NO_TAB="y"
;;
-nx) shift
NO_IND="y"
;;
*) echo "Bad arg: $1"
exit 1;
;;
esac
done

if [ "$MULT" = "" ]
then
echo $N "Database multiplier (# of warehouses)? [1]" $C
read MULT
if [ "$MULT" = "" ]
then
MULT=1
fi
fi

if [ ! -d $BUILD_HOME ]
then
mkdir $BUILD_HOME
fi

if [ ! -d $LOAD_SCRIPTS ]
then
mkdir $LOAD_SCRIPTS
fi

if [ ! -d $LDIR ]
then
mkdir $LDIR
fi

if [ ! -d $OUTDIR ]
then
mkdir $OUTDIR
fi

#
# Load history table
#
I=1
SW=1
INC=107
let EW=$SW+$INC-1
J=1
X=$J
while [ $J -le 10 ]
do
let X=$J*10
while [ $I -le $X ]
do
tpccload -M $MULT -h -b $SW -e $EW > ${LDIR}/hist${I}.dat 2>\
${OUTDIR}/hist${I}.out &
I='expr $I + 1'
SW='expr $SW + $INC'
EW='expr $EW + $INC'
done
wait
J='expr $J + 1'
done
Wait

load_item.sh

#!/bin/ksh

# load <obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
#
=====+
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
=====+
# NAME
# load_<obj>.sh

```



```

# DESCRIPTION
# Usage: load_<obj>.sh [options]
#
#
# .setenv
#
tpccload -M $MULT -i

                                load_nord.sh

#!/bin/ksh
#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#
#====+
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#====+
# FILENAME
# pload.sh
# DESCRIPTION
# Usage: load_<obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#
#
# .setenv
#
if echo "`c" | grep c >/dev/null 2>&1; then
  N='-n'
else
  C='c'
fi
export N C

while [ "$#" != "0" ]
do
  case $1 in
    -mu) shift
      if [ "$1" != "" ]
      then
        MULT=$1
        shift
      fi
      ;;
    -nd) shift
      NO_DB="y"
      ;;
    -nt) shift
      NO_TAB="y"
      ;;
    -nx) shift
      NO_IND="y"
      ;;
    *) echo "Bad arg: $1"
      exit 1;
      ;;
  esac
done

if [ "$MULT" = "" ]
then
  echo $N "Database multiplier (# of warehouses)? [1]" $C
  read MULT
  if [ "$MULT" = "" ]
  then
    MULT=1
  fi
fi

if [ ! -d $BUILD_HOME ]
then
  mkdir $BUILD_HOME
fi

if [ ! -d $LOAD_SCRIPTS ]
then
  mkdir $LOAD_SCRIPTS
fi

if [ ! -d $LDIR ]
then
  mkdir $LDIR
fi

if [ ! -d $OUTDIR ]
then
  mkdir $OUTDIR
fi

#
# Load new-order table
#
l=1
SW=1

```

```

INC=107
let EW=$SW+$INC-1
J=1
X=$J
while [ $J -le 10 ]
do
  let X=$J*10
  while [ $I -le $X ]
  do
    tpccload -M $MULT -n -b $SW -e $EW > ${LDIR}/neword${I}.dat 2>\
      ${OUTDIR}/neword${I}.out &
    I=`expr $I + 1`
    SW=`expr $SW + $INC`
    EW=`expr $EW + $INC`
  done
  wait
  J=`expr $J + 1`
done
Wait

                                load_ordr.sh

#!/bin/ksh
#
# load_<obj>.sh 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle Corp.
#
#
#====+
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
#====+
# FILENAME
# pload.sh
# DESCRIPTION
# Usage: load_<obj>.sh [options]
# -mu <multiplier> (# of warehouses)
#
#
# .setenv
#
if echo "`c" | grep c >/dev/null 2>&1; then
  N='-n'
else
  C='c'
fi
export N C

while [ "$#" != "0" ]
do
  case $1 in
    -mu) shift
      if [ "$1" != "" ]
      then
        MULT=$1
        shift
      fi
      ;;
    -nd) shift
      NO_DB="y"
      ;;
    -nt) shift
      NO_TAB="y"
      ;;
    -nx) shift
      NO_IND="y"
      ;;
    *) echo "Bad arg: $1"
      exit 1;
      ;;
  esac
done

if [ "$MULT" = "" ]
then
  echo $N "Database multiplier (# of warehouses)? [1]" $C
  read MULT
  if [ "$MULT" = "" ]
  then
    MULT=1
  fi
fi

if [ ! -d $BUILD_HOME ]
then
  mkdir $BUILD_HOME
fi

if [ ! -d $LOAD_SCRIPTS ]
then
  mkdir $LOAD_SCRIPTS
fi

if [ ! -d $LDIR ]
then
  mkdir $LDIR
fi

```

```

fi
if [ ! -d $OUTDIR ]
then
  mkdir $OUTDIR
fi
#
# Load order and order-line table
#
I=1
SW=1
INC=21
let EW=$SW+$INC-1
J=1
X=$J
while [ $J -le 20 ]
do
  let X=$J*25
  while [ $I -le $X ]
  do
    tpcpload -M $MULT -o ${LDIR}/ordline${I}.dat -b $SW -e $EW > \
      ${LDIR}/order${I}.dat 2> ${OUTDIR}/order${I}.out &
    I=`expr $I + 1`
    SW=`expr $SW + $INC`
    EW=`expr $EW + $INC`
  done
  wait
  J=`expr $J + 1`
done
INC=20
let EW=$SW+$INC-1
J=1
while [ $J -le 10 ]
do
  tpcpload -M $MULT -o ${LDIR}/ordline${I}.dat -b $SW -e $EW > \
    ${LDIR}/order${I}.dat 2> ${OUTDIR}/order${I}.out &
  I=`expr $I + 1`
  SW=`expr $SW + $INC`
  EW=`expr $EW + $INC`
  J=`expr $J + 1`
done
Wait

```

### load\_stok.sh

```

#!/bin/ksh

# load <obj> 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
#=====  

# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |  

# OPEN SYSTEMS PERFORMANCE GROUP |  

# All Rights Reserved |  

#=====  

# NAME  

# load <obj>.sh  

# DESCRIPTION  

# Usage: load_<obj>.sh [options]  

#=====  

#  

# setenv  

#  

# Load stock table (in parallel with loading customer table)  

#  

I=1
SI=1
EI=200
INC=200
J=1
X=$J
while [ $J -le 10 ]
do
  let X=$J*50
  while [ $I -le $X ]
  do
    tpcpload -M $MULT -S -j $SI -k $EI > ${OUTDIR}/stk${I}.out 2>&1 &  

    I=`expr $I + 1`  

    SI=`expr $SI + $INC`  

    EI=`expr $EI + $INC`  

  done
  wait
  J=`expr $J + 1`
done
Wait

```

### load\_ware.sh

```
#!/bin/ksh
```

```

# load_<obj> 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
#=====  

# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |  

# OPEN SYSTEMS PERFORMANCE GROUP |  

# All Rights Reserved |  

#=====  

# NAME  

# load <obj>.sh  

# DESCRIPTION  

# Usage: load_<obj>.sh [options]  

#=====  

#  

# setenv  

#  

tpcpload -M $MULT -w

```

### realter\_temp.sh

```

#!/bin/ksh

# realter_temp 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
#=====  

# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |  

# OPEN SYSTEMS PERFORMANCE GROUP |  

# All Rights Reserved |  

#=====  

# NAME  

# realter_temp  

# DESCRIPTION  

# Usage: realter_temp.sh [options]  

#=====  

#  

# setenv  

#  

svrmgrl <<!  

connect internal  

alter tablespace temp  

  default storage (initial 20K next 20K pctincrease 50);  

exit;  

!
```

### switchlog.sh

```

#
# $Header: switchlog.sh 7030100.1 96/05/02 10:20:11 plai Generic<base> $ Copyr (c) 1995 Oracle
#
#=====  

# Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |  

# OPEN SYSTEMS PERFORMANCE GROUP |  

# All Rights Reserved |  

#=====  

# FILENAME  

# switchlog.sh  

# DESCRIPTION  

# Switch to next log file twice.  

# USAGE  

# switchlog.sh  

#=====#*/

svrmgrl lmode=y <<!  

connect internal;  

alter system switch logfile;  

alter system switch logfile;  

exit;  

!
```

### tpcc\_rol.sh

```

#!/bin/ksh
#
#=====  

# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |  

# OPEN SYSTEMS PERFORMANCE GROUP |  

# All Rights Reserved |  

#=====  

# FILENAME  

# tpccrol.sh  

# DESCRIPTION  

# Script file for creating the roll;back segments.
#=====#

```

```

#
setenv

svrmgrl <<!  

connect internal;  

@$LOAD_SCRIPTS/tpcc_rol;  

exit;  

!  

tpcc_rol.sql  

rem  

=====+  

rem Copyright (c) 1996 Oracle Corp. Redwood Shores, CA |  

rem OPEN SYSTEMS PERFORMANCE GROUP |  

rem All Rights Reserved |  

rem =====+  

rem FILENAME  

rem tpcc_rol.sql  

rem DESCRIPTION  

rem Create rollback segments for TPCC database.  

rem =====+  

rem  

set timing on;  

host date;  

CREATE ROLLBACK SEGMENT t1  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t2  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t3  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t4  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t5  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t6  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t7  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t8  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t9  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t10  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t11  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t12  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t13  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t14  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t15  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t16  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t17  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t18  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t19  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t20  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t21  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t22  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t23  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t24  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t25  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t26  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t27  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t28  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t29  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t30  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t31  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t32  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t33  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t34  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t35  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t36  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t37  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t38  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t39  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t40  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t41  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t42  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t43  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t44  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t45  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t46  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t47  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t48  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t49  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t50  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t51  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t52  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t53  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t54  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t55  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t56  

TABLESPACE roll  

STORAGE (initial 100K next 100K minextents 2);  

CREATE ROLLBACK SEGMENT t57  

TABLESPACE roll

```





















```

STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t652
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t653
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t654
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t655
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t656
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t657
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t658
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t659
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t660
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t661
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t662
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t663
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t664
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t665
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t666
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t667
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t668
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t669
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t670
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t671
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t672
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t673
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t674
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t675
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t676
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t677
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t678
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t679
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t680
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t681
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t682
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t683
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t684
TABLESPACE roll

```

```

STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t685
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t686
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t687
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t688
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t689
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t690
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t691
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t692
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t693
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t694
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t695
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t696
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t697
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t698
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t699
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
CREATE ROLLBACK SEGMENT t700
TABLESPACE roll
STORAGE (initial 100K next 100K minextents 2);
host date;

```

Exit;

## tpcc\_stored\_proc.sh

```

#!/bin/ksh

# tpcc_stored_proc 80301 98/7/7 15:45 vmakhija
# Copyright (c) 1998 Oracle
#
# =====
# Copyright (c) 1998 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
# =====
# NAME
# tpcc_stored_proc
# DESCRIPTION
# Usage: tpcc_stored_proc.sh [options]
#
. setenv

sqlplus tpcc/tpcc @$TPCC_BLOCKS/views
#sqlplus tpcc/tpcc @$TPCC_BLOCKS/pay
#sqlplus tpcc/tpcc @$TPCC_BLOCKS/payz
sqlplus tpcc/tpcc @$TPCC_BLOCKS/initpay
sqlplus tpcc/tpcc @$TPCC_BLOCKS/tkvcin

```

## undml.sh

```

#
# $Header: undml.sh 7030100.2 96/05/02 10:29:30 plai Generic<base> $ Copyr (c) 1995 Oracle
#
# =====
# Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
# OPEN SYSTEMS PERFORMANCE GROUP |
# All Rights Reserved |
# =====
# FILENAME
# undml.sh
# DESCRIPTION
# Enable table locks for TPC-C tables.

```

```
# USAGE
# undml.sh
#=====*/
sqlplus tpcc/tpcc <<!
alter table warehouse enable table lock;
alter table district enable table lock;
alter table customer enable table lock;
alter table history enable table lock;
alter table item enable table lock;
alter table stock enable table lock;
alter table orders enable table lock;
alter table new_order enable table lock;
alter table order_line enable table lock;
quit;
```

## C.2 SQL Scripts

### cust.sql

```
rem
rem =====
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====
rem FILENAME
rem tpcc tab2.sql
rem DESCRIPTION
rem Create customer table for TPC-C database.
rem =====
rem
rem
rem DROP all first
rem
rem drop cluster ccluster including tables;
rem drop table customer;
rem
rem set timing on
rem
rem CUSTOMER table
rem
rem
rem create cluster ccluster (
rem c_id number(5,0),
rem c_d_id number(2,0),
rem c_w_id number(5,0)
rem )
rem single table
rem hashkeys 321000000
rem hash is ((c_id * 107000) + (c_w_id * 10) + c_d_id)
rem size 850
rem initrans 3
rem pctfree 0
rem tablespace cust
rem storage (buffer_pool recycle);
rem
rem create table customer (
rem c_id number(5,0),
rem c_d_id number(2,0),
rem c_w_id number(5,0),
rem c_discount number,
rem c_credit char(2),
rem c_last varchar2(16),
rem c_first varchar2(16),
rem c_credit_lim number,
rem c_balance number,
rem c_ytd_payment number,
rem c_payment_cnt number,
rem c_delivery_cnt number,
rem c_street_1 varchar2(20),
rem c_street_2 varchar2(20),
rem c_city varchar2(20),
rem c_state char(2),
rem c_zip char(9),
rem c_phone char(16),
rem c_since date,
rem c_middle char(2),
rem c_data varchar2(500)
rem )
rem cluster ccluster (c_id, c_d_id, c_w_id);
rem
rem
rem done
rem
rem
rem exit;
```

### dist.sql

```
rem
rem =====
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====
rem FILENAME
rem dist.sql
rem DESCRIPTION
rem Create customer table for TPC-C database.
rem =====
rem
rem
rem DROP all first
rem
rem drop table district;
rem drop cluster dcluster including tables;
rem
rem set timing on
rem
rem DISTRICT table
rem
rem
rem create cluster dcluster (
rem d_w_id number(5,0),
rem d_id number(2,0)
rem )
rem single table
rem hashkeys 107000
rem hash is (d_w_id * 10 + d_id)
rem size 1536
rem initrans 3
rem pctfree 0
rem tablespace ware;
rem
rem create table district (
rem d_id number(2,0),
rem d_w_id number(5,0),
rem d_ytd number,
rem d_tax number,
rem d_next_o_id number,
rem d_name varchar2(10),
rem d_street_1 varchar2(20),
rem d_street_2 varchar2(20),
rem d_city varchar2(20),
rem d_state char(2),
rem d_zip char(9)
rem )
rem cluster dcluster (d_w_id, d_id);
rem
rem done
rem
rem
rem exit;
```

### hist.sql

```
rem
rem =====
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====
rem FILENAME
rem hist.sql
rem DESCRIPTION
rem Create customer table for TPC-C database.
rem =====
rem
rem
rem DROP all first
rem
rem drop table history;
rem
rem set timing on
rem
rem HISTORY table
rem
rem
rem create table history (
rem h_c_id number,
rem h_c_d_id number,
rem h_c_w_id number,
rem h_d_id number,
rem h_w_id number,
rem h_date date,
rem h_amount number,
rem h_data varchar2(24)
rem )
rem tablespace hist
```

```

initrans 4
pctfree 5
storage (freelist groups 43 freelists 19
        buffer_pool recycle);

rem
rem done
rem
exit;

                                icust.sql

rem
rem =====+
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====+
rem FILENAME
rem icust.sql
rem DESCRIPTION
rem Create customer index for TPC-C database.
rem =====+
rem
rem
rem DROP all first
rem
drop index icustomer;

set timing on

rem
rem ICUST1 index
rem

create unique index icustomer on customer(c_w_id, c_d_id, c_id)
tablespace icust1
initrans 3
parallel 32
pctfree 1
storage (freelist groups 13 freelists 22);

rem
rem done
rem
exit;

                                icust2.sql

rem
rem =====+
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====+
rem FILENAME
rem icust.sql
rem DESCRIPTION
rem Create customer index for TPC-C database.
rem =====+
rem
rem
rem DROP all first
rem
drop index icustomer;

set timing on

rem
rem ICUST1 index
rem

create unique index icustomer on customer(c_w_id, c_d_id, c_id)
tablespace icust1
initrans 3
parallel 32
pctfree 1
storage (freelist groups 13 freelists 22);

rem
rem done
rem
exit;

                                idist.sql

rem
rem =====+

```

```

rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====+
rem FILENAME
rem dist.sql
rem DESCRIPTION
rem Create customer table for TPC-C database.
rem =====+
rem
rem
rem DROP all first
rem
drop index idistrict;

set timing on

rem
rem DISTRICT index
rem

create unique index idistrict on district(d_w_id, d_id)
tablespace ware
initrans 3
parallel 10
pctfree 5
storage (freelist groups 13 freelists 22);

rem
rem done
rem
exit;

                                iitem.sql

rem
rem =====+
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====+
rem FILENAME
rem item.sql
rem DESCRIPTION
rem Create customer table for TPC-C database.
rem =====+
rem
rem
rem DROP all first
rem
drop index iitem;

set timing on

rem
rem ITEM index
rem

create unique index iitem on item(i_id)
tablespace items
initrans 4
pctfree 5
storage (freelist groups 13 freelists 22
        buffer_pool keep);

rem
rem done
rem
exit;

                                iordr.sql

rem
rem =====+
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====+
rem FILENAME
rem iordr.sql
rem DESCRIPTION
rem Create orders index for TPC-C database.
rem =====+
rem
rem
rem DROP all first
rem
drop index iorders;

set timing on

```

```

rem
rem ORDERS index
rem

create unique index iorders on orders(o_w_id, o_d_id, o_id)
tablespace iord1
intrans 3
parallel 64
pctfree 1
storage (freelist groups 13 freelists 22);

```

```

rem
rem done
rem

Exit;

```

## iordr2.sql

```

rem
rem =====+
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====+
rem FILENAME
rem iordr2.sql
rem DESCRIPTION
rem Create orders index 2 for TPC-C database.
rem =====+
rem

```

```

rem
rem DROP all first
rem
drop index iorders2;

```

set timing on

```

rem
rem ORDERS index 2
rem

```

```

create unique index iorders2 on orders(o_c_id, o_d_id, o_w_id, o_id)
tablespace iord2
intrans 4
parallel 64
storage (freelist groups 13 freelists 22)
pctfree 25;

```

```

rem
rem done
rem

exit;

```

## istok.sql

```

rem
rem =====+
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====+
rem FILENAME
rem istok.sql
rem DESCRIPTION
rem Create stock index for TPC-C database.
rem =====+
rem

```

```

rem
rem DROP all first
rem
drop index istock;

```

set timing on

```

rem
rem STOCK index
rem

```

```

create unique index istock on stock(s_i_id, s_w_id)
tablespace istk
intrans 3
parallel 16
pctfree 1
storage (freelist groups 13 freelists 22);

```

```

rem
rem done

```

```

rem

exit;

```

## item.sql

```

rem
rem =====+
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====+
rem FILENAME
rem item.sql
rem DESCRIPTION
rem Create ITEM table for TPC-C database.
rem =====+
rem

```

```

rem
rem DROP item cluster and table
rem
drop cluster icluster including tables;
drop table item;

```

set timing on;

```

rem
rem ITEM table
rem

```

```

create cluster icluster (
i_id number (6,0)
)
single table
hashkeys 100000
hash is i_id
size 120
intrans 3
pctfree 0
tablespace items
storage (buffer_pool keep);

```

```

create table item (
i_id number(6,0),
i_name varchar2(24),
i_price number,
i_data varchar2(50),
i_im_id number
)
cluster icluster(i_id);

```

```

rem
rem done
rem

exit;

```

## iware.sql

```

rem
rem =====+
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====+
rem FILENAME
rem iware.sql
rem DESCRIPTION
rem Create warehouse index for TPC-C database.
rem =====+
rem

```

```

rem
rem DROP all first
rem
drop index iwarehouse;

```

set timing on

```

rem
rem WAREHOUSE index
rem

```

```

create unique index iwarehouse on warehouse (w_id)
tablespace ware
intrans 3
pctfree 1;

```

```

rem
rem done
rem

exit;

```

## nord.sql



```

rem
rem =====+
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====+
rem FILENAME
rem nord.sql
rem DESCRIPTION
rem Create customer table for TPC-C database.
rem =====+
rem
rem
rem DROP all first
rem
rem drop table new_order;

set timing on

rem
rem NEW_ORDER table
rem
create table new_order (
  no_w_id number,
  no_d_id number,
  no_o_id number,
  constraint inord primary key (no_w_id, no_d_id, no_o_id)
)
  organization index tablespace nord
  intrans 4
  pctfree 5
  storage ( freelist groups 43 freelists 19);

rem
rem done
rem
exit;

```

## ordl.sql

```

rem
rem =====+
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====+
rem FILENAME
rem odr.sql
rem DESCRIPTION
rem Create customer table for TPC-C database.
rem =====+
rem
rem
rem DROP all first
rem
rem drop table order_line;

set timing on

rem
rem ORDER_LINE table
rem
create table order_line (
  ol_w_id number,
  ol_d_id number,
  ol_o_id number,
  ol_number number,
  ol_i_id number,
  ol_delivery_d date,
  ol_amount number,
  ol_supply_w_id number,
  ol_quantity number,
  ol_dist_info char(24),
  constraint iordl primary key (ol_w_id, ol_d_id, ol_o_id, ol_number)
)
  organization index tablespace ordl
  intrans 4
  pctfree 5
  storage ( freelist groups 43 freelists 19);

rem
rem done
rem
exit;

```

## odr.sql

```

rem
rem =====+
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====+
rem FILENAME
rem odr.sql
rem DESCRIPTION
rem Create customer table for TPC-C database.
rem =====+
rem
rem
rem DROP all first
rem
rem drop table orders;

set timing on

rem
rem ORDERS table
rem
create table orders (
  o_id number,
  o_w_id number,
  o_d_id number,
  o_c_id number,
  o_carrier_id number,
  o_ol_cnt number,
  o_all_local number,
  o_entry_d date
)
  tablespace ord
  intrans 4
  pctfree 5
  storage ( freelist groups 43 freelists 19);

rem
rem done
rem
exit;

```

## stok.sql

```

rem
rem =====+
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====+
rem FILENAME
rem tpcc_tab3.sql
rem DESCRIPTION
rem Create stock table for TPC-C database.
rem =====+
rem
rem
rem DROP all first
rem
rem drop cluster scluster including tables;
rem drop table stock;

set timing on

rem
rem STOCK table
rem
create cluster scluster (
  s_i_id number(6,0),
  s_w_id number(5,0)
)
  single table
  hashkeys 107000000
  hash is (s_i_id * 10700 + s_w_id)
  size 350
  intrans 3
  pctfree 0
  tablespace stocks
  storage (freelist groups 4 freelists 11 buffer_pool keep);

create table stock (
  s_i_id number(6,0),
  s_w_id number(5,0),
  s_quantity number,
  s_ytd number,
  s_order_cnt number,
  s_remote_cnt number,
  s_data varchar2(50),
  s_dist_01 char(24),
  s_dist_02 char(24),
  s_dist_03 char(24),
  s_dist_04 char(24),

```

```

s_dist_05 char(24),
s_dist_06 char(24),
s_dist_07 char(24),
s_dist_08 char(24),
s_dist_09 char(24),
s_dist_10 char(24)
)
cluster scluster (s_i_id, s_w_id);

rem
rem done
rem

exit;

tpcc_ana.sql

rem
rem =====
rem Copyright (c) 1995 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====
rem FILENAME
rem tpcc_ana.sql
rem DESCRIPTION
rem Analyze all tables and indexes of TPC-C database.
rem
rem
set timing on;
analyze table warehouse compute statistics;
analyze table district compute statistics;
analyze table item estimate statistics;
analyze table history estimate statistics;
analyze table customer estimate statistics;
analyze table stock estimate statistics;
analyze table orders estimate statistics;
analyze table new_order estimate statistics;
analyze table order_line estimate statistics;
analyze cluster icluster estimate statistics;
analyze cluster scluster estimate statistics;
analyze cluster ccluster estimate statistics;
analyze index iwarehouse compute statistics;
analyze index idistrict compute statistics;
analyze index icustomer estimate statistics;
analyze index icustomer2 estimate statistics;
analyze index istock estimate statistics;
analyze index item estimate statistics;
analyze index iorders estimate statistics;
analyze index iorders2 estimate statistics;
analyze index inew_order estimate statistics;
analyze index iorder_line estimate statistics;
quit;

ware.sql

rem
rem =====
rem Copyright (c) 1996 Oracle Corp, Redwood Shores, CA |
rem OPEN SYSTEMS PERFORMANCE GROUP |
rem All Rights Reserved |
rem =====
rem FILENAME
rem ware.sql
rem DESCRIPTION
rem Create customer table for TPC-C database.
rem
rem
rem DROP all first
rem
drop table warehouse;
drop cluster wcluster including tables;

set timing on

rem
rem WAREHOUSE table
rem
create cluster wcluster (
w_id number (5,0)
)
single table
hashkeys 10700
hash is w_id
size 1536
intrans 3
pctfree 0

tablespace ware;

create table warehouse (
w_id number(5,0),
w_ytd number,
w_tax number,

```

```

w_name varchar2(10),
w_street_1 varchar2(20),
w_street_2 varchar2(20),
w_city varchar2(20),
w_state char(2),
w_zip char(9)
)
cluster wcluster (w_id);

rem
rem done
rem

exit;

C.3 Data Generation Code

tpccload.c

#ifdef RCSID
static char *RCSid =
"SHheader: 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 warehouses> [options]
options: -A load all tables
-w load warehouse table
-d load district table
-c load customer table
-i load item table
-s load stock table (cluster around s_w_id)
-S load stock table (cluster around s_i_id)
-h load history table
-n load new-order table
-o <oline file> load order and order-line table
-b <ware#> beginning warehouse number
-e <ware#> ending warehouse number
-j <item#> beginning item number (with -S)
-k <item#> ending item number (with -S)
-g generate rows to standard output
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <time.h>
#include <sys/types.h>
#include "tpcc.h"

#define DISTARR 10 /* district insert array size */
#define CUSTARR 100 /* customer insert array size */
#define STOCARR 100 /* stock insert array size */
#define ITEMARR 100 /* item insert array size */
#define HISTARR 100 /* history insert array size */
#define ORDEARR 100 /* order insert array size */
#define NEWOARR 100 /* new order insert array size */

#define DISTFAC 10 /* max. district id */
#define CUSTFAC 3000 /* max. customer id */
#define STOCFAC 100000 /* max. stock 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 SQLTXTW "INSERT INTO warehouse (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 SQLXTXD "INSERT INTO district (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, 30000000, :d_tax, \ 3001, :d_name, :d_street_1, :d_street_2, :d_city, :d_state, :d_zip)"

```

<pre> #define SQLTXTC "INSERT INTO customer (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 history (h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id, h_date, h_amount, h_data) VALUES (:h_c_id, :h_c_d_id, :h_c_w_id, \ :h_d_id, :h_w_id, SYSDATE, 1000, :h_data)"  #define SQLTXTS "INSERT INTO stock (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 SQLXTI "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 SQLXTO1 "INSERT INTO orders (O_ID, O_D_ID, O_W_ID, O_C_ID, O_ENTRY_D, O_CARRIER_ID, O_OL_CNT, O_ALL_LOCAL) \ VALUES (:o_id, :o_d_id, :o_w_id, :o_c_id, \ SYSDATE, :o_carrier_id, :o_ol_cnt, 1)"  #define SQLXTO2 "INSERT INTO orders (O_ID, O_D_ID, O_W_ID, O_C_ID, O_ENTRY_D, O_CARRIER_ID, O_OL_CNT, O_ALL_LOCAL) \ VALUES (:o_id, :o_d_id, :o_w_id, :o_c_id, \ SYSDATE, 11, :o_ol_cnt, 1)"  #define SQLXTOL1 "INSERT INTO order_line (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 SQLXTOL2 "INSERT INTO order_line (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 SQLXTNO "INSERT INTO new_order (no_o_id, no_d_id, no_w_id) VALUES (:no_o_id, no_d_id, :no_w_id)"  ldadef tpclda; csrdef curw, curd, curc, curh, curs, curi, curo1, curo2, curo11, curo12, curno; unsigned long tpclda[256];  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];  myusage() { printf(stderr, "\n"); printf(stderr, "Usage:\tpccload -M &lt;multiplier&gt; [options]\n"); printf(stderr, "options:\n"); printf(stderr, "\t-A :tload all tables\n"); printf(stderr, "\t-w :tload warehouse table\n"); printf(stderr, "\t-d :tload district table\n"); printf(stderr, "\t-c :tload customer table\n"); printf(stderr, "\t-i :tload item table\n"); printf(stderr, "\t-s :tload stock table (cluster around s_w_id)\n"); printf(stderr, "\t-S :tload stock table (cluster around s_i_id)\n"); printf(stderr, "\t-h :tload history table\n"); printf(stderr, "\t-n :tload new-order table\n"); printf(stderr, "\t-o &lt;oline file&gt; :tload order and order-line table\n"); printf(stderr, "\t-b &lt;ware#&gt; :tbeginning warehouse number\n"); printf(stderr, "\t-e &lt;ware#&gt; :tending warehouse number\n"); printf(stderr, "\t-j &lt;item#&gt; :tbeginning item number (with -S)\n"); printf(stderr, "\t-k &lt;item#&gt; :tending item number (with -S)\n"); printf(stderr, "\t-g :tgenerate rows to standard output\n"); printf(stderr, "\n"); exit(1); } </pre>	<pre> }  errrpt (lda, cur)  csrdef *lda; csrdef *cur;  { text msg[2048];  if (cur-&gt;rc) { oerhms (lda, cur-&gt;rc, msg, 2048); fprintf (stderr, "TPC-C load error: %s\n", msg); }  }  quit ()  { if (oclose (&amp;curw)) errrpt (&amp;tpclda, &amp;curw);  if (oclose (&amp;curd)) errrpt (&amp;tpclda, &amp;curd);  if (oclose (&amp;curc)) errrpt (&amp;tpclda, &amp;curc);  if (oclose (&amp;curh)) errrpt (&amp;tpclda, &amp;curh);  if (oclose (&amp;curs)) errrpt (&amp;tpclda, &amp;curs);  if (oclose (&amp;curi)) errrpt (&amp;tpclda, &amp;curi);  if (oclose (&amp;curo1)) errrpt (&amp;tpclda, &amp;curo1);  if (oclose (&amp;curo2)) errrpt (&amp;tpclda, &amp;curo2);  if (oclose (&amp;curo11)) errrpt (&amp;tpclda, &amp;curo11);  if (oclose (&amp;curo12)) errrpt (&amp;tpclda, &amp;curo12);  if (oclose (&amp;curno)) errrpt (&amp;tpclda, &amp;curno);  if (ologof (&amp;tpclda)) fprintf (stderr, "TPC-C load error: Error in logging off\n"); }  main (argc, argv)  int argc; char *argv[];  { char *uid="tpcc/tpcc"; text sqlbuf[1024]; 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; </pre>
---	---

```

int d_id[10];
int d_w_id[10];
char d_name[10][11];
char d_street_1[10][21];
char d_street_2[10][21];
char d_city[10][21];
char d_state[10][2];
char d_zip[10][9];
float d_tax[10];

int c_id[100];
int c_d_id[100];
int c_w_id[100];
char c_first[100][17];
char c_last[100][17];
char c_street_1[100][21];
char c_street_2[100][21];
char c_city[100][21];
char c_state[100][2];
char c_zip[100][9];
char c_phone[100][16];
char c_credit[100][2];
float c_discount[100];
char c_data[100][501];

int i_id[100];
int i_im_id[100];
int i_price[100];
char i_name[100][25];
char i_data[100][51];

int s_i_id[100];
int s_w_id[100];
int s_quantity[100];
char s_dist_01[100][24];
char s_dist_02[100][24];
char s_dist_03[100][24];
char s_dist_04[100][24];
char s_dist_05[100][24];
char s_dist_06[100][24];
char s_dist_07[100][24];
char s_dist_08[100][24];
char s_dist_09[100][24];
char s_dist_10[100][24];
char s_data[100][51];

int h_w_id[100];
int h_d_id[100];
int h_c_id[100];
char h_data[100][25];

int o_id[100];
int o_d_id[100];
int o_w_id[100];
int o_c_id[100];
int o_carrier_id[100];
int o_ol_cnt[100];

int ol_o_id[15];
int ol_d_id[15];
int ol_w_id[15];
int ol_number[15];
int ol_i_id[15];
int ol_supply_w_id[15];
int ol_amount[15];
char ol_dist_info[15][24];

int no_o_id[100];
int no_d_id[100];
int no_w_id[100];

char sdate[30];

double begin_time, end_time;
double begin_cpu, end_cpu;
double gettime(), getcpu();

extern int getopt();
extern char *optarg;
extern int optind, opterr;

char *argstr="M:AwdeisShno:bre;j:k:g";
int opt;
int do_A=0;
int do_w=0;
int do_d=0;
int do_i=0;
int do_c=0;
int do_s=0;
int do_S=0;
int do_h=0;
int do_o=0;
int do_n=0;
int gen=0;
int bware=1;
int aware=0;
int bitem=1;
int eitem=0;

```

```

FILE *ofp=NULL;
char olname[100];
#define FIRSTNAME_WITH_CLAST
#ifndef FIRSTNAME_WITH_CLAST
char firstname_with_clast[100];
sprintf(firstname_with_clast, "C_LAST=%d", CNUM1);
#endif /* FIRSTNAME_WITH_CLAST */

/*-----+
| Parse command line -- look for scale factor.
+-----*/

if (argc == 1) {
myusage ();
}

while ((opt = getopt (argc, argv, argstr)) != -1) {
switch (opt) {
case '?': myusage ();
break;
case 'M': scale = atoi (optarg);
break;
case 'A': do_A = 1;
break;
case 'w': do_w = 1;
break;
case 'd': do_d = 1;
break;
case 'c': do_c = 1;
break;
case 'i': do_i = 1;
break;
case 's': do_s = 1;
break;
case 'S': do_S = 1;
break;
case 'h': do_h = 1;
break;
case 'n': do_n = 1;
break;
case 'o': do_o = 1;
strcpy (olname, optarg);
break;
case 'b': bware = atoi (optarg);
break;
case 'e': aware = atoi (optarg);
break;
case 'j': bitem = atoi (optarg);
break;
case 'k': eitem = atoi (optarg);
break;
case 'g': gen = 1;
break;
default: fprintf (stderr, "THIS SHOULD NEVER HAPPEN!!!\n");
fprintf (stderr, "reached default case in getopt ()\n");
myusage ();
}
}

/*-----*
| Rudimentary error checking
+-----*/

if (scale < 1) {
fprintf (stderr, "Invalid scale factor: %d\n", scale);
myusage ();
}

if (!(do_A || do_w || do_d || do_c || do_i || do_s || do_S || do_h || do_o ||
do_n)) {
fprintf (stderr, "What should I load???\n");
myusage ();
}

if (gen && (do_A || (do_w + do_d + do_c + do_i + do_s + do_S + do_h + do_o +
do_n > 1))) {
fprintf (stderr, "Can only generate table one at a time\n");
myusage ();
}

if (do_S && (do_A || do_s)) {
fprintf (stderr, "Cluster stock table around s_w_id or s_i_id?\n");
myusage ();
}

if (aware <= 0)
aware = scale;
if (eitem <= 0)
eitem = STOCFAC;

if (do_S) {
if ((bitem < 1) || (bitem > STOCFAC)) {
fprintf (stderr, "Invalid beginning item number: %d\n", bitem);
myusage ();
}
}

if ((eitem < bitem) || (eitem > STOCFAC)) {
fprintf (stderr, "Invalid ending item number: %d\n", eitem);
}

```

```

mysusage ();
}
}

if ((bware < 1) || (bware > scale)) {
    fprintf(stderr, "Invalid beginning warehouse number: %d\n", bware);
    mysusage ();
}

if ((eware < bware) || (eware > scale)) {
    fprintf(stderr, "Invalid ending warehouse number: %d\n", eware);
    mysusage ();
}

if (gen && do_o) {
    if ((olfp = fopen(olfname, "w")) == NULL) {
        fprintf(stderr, "Can't open '%s' for writing order lines\n", olfname);
        mysusage ();
    }
}

*-----+
| Prepare to insert into database.
+-----*/

sysdate (sdate);
if (!gen) {

    /* log on to Oracle */

    if (orlon (&tpclda, (ub1 *) tpchda, (text *) uid, -1, (text *) 0, -1, 0)) {
        fprintf(stderr, "TPC-C load error: Error in logging on\n");
        errrpt (&tpclda, &tpclda);
        exit (1);
    }

    fprintf(stderr, "\nConnected to Oracle userid '%s'.\n", uid);

    /* turn off auto-commit */

    if (ocof (&tpclda)) {
        errrpt (&tpclda, &tpclda);
        ologof (&tpclda);
        exit (1);
    }

    /* open cursors */

    if (oopen (&curw, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
        errrpt (&tpclda, &curw);
        ologof (&tpclda);
        exit (1);
    }

    if (oopen (&curd, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
        errrpt (&tpclda, &curd);
        oclose (&curw);
        ologof (&tpclda);
        exit (1);
    }

    if (oopen (&curc, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
        errrpt (&tpclda, &curc);
        oclose (&curw);
        oclose (&curd);
        ologof (&tpclda);
        exit (1);
    }

    if (oopen (&curh, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
        errrpt (&tpclda, &curh);
        oclose (&curw);
        oclose (&curd);
        oclose (&curc);
        ologof (&tpclda);
        exit (1);
    }

    if (oopen (&curs, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
        errrpt (&tpclda, &curs);
        oclose (&curw);
        oclose (&curd);
        oclose (&curc);
        oclose (&curh);
        ologof (&tpclda);
        exit (1);
    }

    if (oopen (&curi, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
        errrpt (&tpclda, &curi);
        oclose (&curw);
        oclose (&curd);
        oclose (&curc);
        oclose (&curh);
        oclose (&curs);
        ologof (&tpclda);
        exit (1);
    }
}

```

```

if (oopen (&curol, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &curol);
    oclose (&curw);
    oclose (&curd);
    oclose (&curc);
    oclose (&curh);
    oclose (&curs);
    oclose (&curi);
    ologof (&tpclda);
    exit (1);
}

if (oopen (&curol2, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &curol2);
    oclose (&curw);
    oclose (&curd);
    oclose (&curc);
    oclose (&curh);
    oclose (&curs);
    oclose (&curi);
    oclose (&curol);
    ologof (&tpclda);
    exit (1);
}

if (oopen (&curol1, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &curol1);
    oclose (&curw);
    oclose (&curd);
    oclose (&curc);
    oclose (&curh);
    oclose (&curi);
    oclose (&curol);
    oclose (&curol2);
    ologof (&tpclda);
    exit (1);
}

if (oopen (&curol2, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &curol2);
    oclose (&curw);
    oclose (&curd);
    oclose (&curc);
    oclose (&curh);
    oclose (&curs);
    oclose (&curi);
    oclose (&curol);
    oclose (&curol2);
    ologof (&tpclda);
    exit (1);
}

if (oopen (&cuorno, &tpclda, (text *) 0, -1, -1, (text *) uid, -1)) {
    errrpt (&tpclda, &cuorno);
    oclose (&curw);
    oclose (&curd);
    oclose (&curc);
    oclose (&curh);
    oclose (&curs);
    oclose (&curi);
    oclose (&curol);
    oclose (&curol2);
    oclose (&curol1);
    ologof (&tpclda);
    exit (1);
}

/* parse statements */

sprintf ((char *) sqlbuf, SQLTXTW);
if (oparse (&curw, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
}

sprintf ((char *) sqlbuf, SQLTXTD);
if (oparse (&curd, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
}

sprintf ((char *) sqlbuf, SQLTXTC);
if (oparse (&curc, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

sprintf ((char *) sqlbuf, SQLTXTH);
if (oparse (&curh, sqlbuf, -1, 0, 1)) {
    errrpt (&tpclda, &curh);
    quit ();
    exit (1);
}
}

```

```

sprintf ((char *) sqlbuf, SQLTXTS);
if (oparse (&curs, sqlbuf, -1, 0, 1) {
    errrpt (&tpclda, &curs);
    quit ();
    exit (1);
}

sprintf ((char *) sqlbuf, SQLXTXI);
if (oparse (&curi, sqlbuf, -1, 0, 1) {
    errrpt (&tpclda, &curi);
    quit ();
    exit (1);
}

sprintf ((char *) sqlbuf, SQLTXTO1);
if (oparse (&curo1, sqlbuf, -1, 0, 1) {
    errrpt (&tpclda, &curo1);
    quit ();
    exit (1);
}

sprintf ((char *) sqlbuf, SQLTXTO2);
if (oparse (&curo2, sqlbuf, -1, 0, 1) {
    errrpt (&tpclda, &curo2);
    quit ();
    exit (1);
}

sprintf ((char *) sqlbuf, SQLTXTOL1);
if (oparse (&curo1, sqlbuf, -1, 0, 1) {
    errrpt (&tpclda, &curo1);
    quit ();
    exit (1);
}

sprintf ((char *) sqlbuf, SQLTXTOL2);
if (oparse (&curo2, sqlbuf, -1, 0, 1) {
    errrpt (&tpclda, &curo2);
    quit ();
    exit (1);
}

sprintf ((char *) sqlbuf, SQLTXTNO);
if (oparse (&curno, sqlbuf, -1, 0, 1) {
    errrpt (&tpclda, &curno);
    quit ();
    exit (1);
}

/* bind variables */

/* warehouse */

if (obndrv (&curw, (text *) "w_id", -1, (ub1 *) &w_id, sizeof (w_id),
    SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
}

if (obndrv (&curw, (text *) "w_name", -1, (ub1 *) w_name, 11,
    SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
}

if (obndrv (&curw, (text *) "w_street_1", -1, (ub1 *) w_street_1, 21,
    SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
}

if (obndrv (&curw, (text *) "w_street_2", -1, (ub1 *) w_street_2, 21,
    SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
}

if (obndrv (&curw, (text *) "w_city", -1, (ub1 *) w_city, 21,
    SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
}

if (obndrv (&curw, (text *) "w_state", -1, (ub1 *) w_state, 2,
    SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
}

if (obndrv (&curw, (text *) "w_zip", -1, (ub1 *) w_zip, 9,
    SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curw);
}

```

```

quit ();
exit (1);
}

if (obndrv (&curw, (text *) "w_tax", -1, (ub1 *) &w_tax, sizeof (w_tax),
    SQLT_FLT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curw);
    quit ();
    exit (1);
}

/* district */

if (obndrv (&curd, (text *) "d_id", -1, (ub1 *) d_id, sizeof (int),
    SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
}

if (obndrv (&curd, (text *) "d_w_id", -1, (ub1 *) d_w_id, sizeof (int),
    SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
}

if (obndrv (&curd, (text *) "d_name", -1, (ub1 *) d_name, 11,
    SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
}

if (obndrv (&curd, (text *) "d_street_1", -1, (ub1 *) d_street_1, 21,
    SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
}

if (obndrv (&curd, (text *) "d_street_2", -1, (ub1 *) d_street_2, 21,
    SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
}

if (obndrv (&curd, (text *) "d_city", -1, (ub1 *) d_city, 21,
    SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
}

if (obndrv (&curd, (text *) "d_state", -1, (ub1 *) d_state, 2,
    SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
}

if (obndrv (&curd, (text *) "d_zip", -1, (ub1 *) d_zip, 9,
    SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
}

if (obndrv (&curd, (text *) "d_tax", -1, (ub1 *) d_tax, sizeof (int),
    SQLT_FLT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curd);
    quit ();
    exit (1);
}

/* customer */

if (obndrv (&curc, (text *) "c_id", -1, (ub1 *) c_id, sizeof (int),
    SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "c_d_id", -1, (ub1 *) c_d_id, sizeof (int),
    SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "c_w_id", -1, (ub1 *) c_w_id, sizeof (int),
    SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

```

```

if (obndrv (&curc, (text *) "c_first", -1, (ub1 *) c_first, 17,
    SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "c_last", -1, (ub1 *) c_last, 17,
    SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "c_street_1", -1, (ub1 *) c_street_1, 21,
    SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "c_street_2", -1, (ub1 *) c_street_2, 21,
    SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "c_city", -1, (ub1 *) c_city, 21,
    SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "c_state", -1, (ub1 *) c_state, 2,
    SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "c_zip", -1, (ub1 *) c_zip, 9,
    SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "c_phone", -1, (ub1 *) c_phone, 16,
    SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "c_credit", -1, (ub1 *) c_credit, 2,
    SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "c_discount", -1, (ub1 *) c_discount,
    sizeof (int), SQT_FLT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "c_data", -1, (ub1 *) c_data, 501,
    SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

/* item */

if (obndrv (&curi, (text *) "i_id", -1, (ub1 *) i_id, sizeof (int),
    SQT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curi);
    quit ();
    exit (1);
}

if (obndrv (&curi, (text *) "i_im_id", -1, (ub1 *) i_im_id, sizeof (int),
    SQT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curi);
    quit ();
    exit (1);
}

if (obndrv (&curi, (text *) "i_name", -1, (ub1 *) i_name, 25,
    SQT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curi);
    quit ();
    exit (1);
}

```

```

}

if (obndrv (&curi, (text *) "i_price", -1, (ub1 *) i_price,
    sizeof (int), SQT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curi);
    quit ();
    exit (1);
}

if (obndrv (&curi, (text *) "i_data", -1, (ub1 *) i_data, 51,
    SQT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curi);
    quit ();
    exit (1);
}

/* stock */

if (obndrv (&curc, (text *) "s_i_id", -1, (ub1 *) s_i_id, sizeof (int),
    SQT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "s_w_id", -1, (ub1 *) s_w_id, sizeof (int),
    SQT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "s_quantity", -1, (ub1 *) s_quantity,
    sizeof (int), SQT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "s_dist_01", -1, (ub1 *) s_dist_01, 24,
    SQT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "s_dist_02", -1, (ub1 *) s_dist_02, 24,
    SQT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "s_dist_03", -1, (ub1 *) s_dist_03, 24,
    SQT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "s_dist_04", -1, (ub1 *) s_dist_04, 24,
    SQT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "s_dist_05", -1, (ub1 *) s_dist_05, 24,
    SQT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "s_dist_06", -1, (ub1 *) s_dist_06, 24,
    SQT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "s_dist_07", -1, (ub1 *) s_dist_07, 24,
    SQT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "s_dist_08", -1, (ub1 *) s_dist_08, 24,
    SQT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
    quit ();
    exit (1);
}

if (obndrv (&curc, (text *) "s_dist_09", -1, (ub1 *) s_dist_09, 24,
    SQT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
    errrpt (&tpclda, &curc);
}

```

```

quit ();
exit (1);
}

if (obndrv (&curs, (text *) ":s_dist_10", -1, (ub1 *) s_dist_10, 24,
SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curs);
quit ();
exit (1);
}

if (obndrv (&curs, (text *) ":s_data", -1, (ub1 *) s_data, 51,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curs);
quit ();
exit (1);
}

/* history */

if (obndrv (&curh, (text *) ":h_c_id", -1, (ub1 *) h_c_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curh);
quit ();
exit (1);
}

if (obndrv (&curh, (text *) ":h_c_d_id", -1, (ub1 *) h_d_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curh);
quit ();
exit (1);
}

if (obndrv (&curh, (text *) ":h_c_w_id", -1, (ub1 *) h_w_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curh);
quit ();
exit (1);
}

if (obndrv (&curh, (text *) ":h_d_id", -1, (ub1 *) h_d_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curh);
quit ();
exit (1);
}

if (obndrv (&curh, (text *) ":h_w_id", -1, (ub1 *) h_w_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curh);
quit ();
exit (1);
}

if (obndrv (&curh, (text *) ":h_data", -1, (ub1 *) h_data, 25,
SQLT_STR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curh);
quit ();
exit (1);
}

/* order_line (delivered) */

if (obndrv (&curol1, (text *) ":ol_o_id", -1, (ub1 *) ol_o_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol1);
quit ();
exit (1);
}

if (obndrv (&curol1, (text *) ":ol_d_id", -1, (ub1 *) ol_d_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol1);
quit ();
exit (1);
}

if (obndrv (&curol1, (text *) ":ol_w_id", -1, (ub1 *) ol_w_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol1);
quit ();
exit (1);
}

if (obndrv (&curol1, (text *) ":ol_number", -1, (ub1 *) ol_number,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol1);
quit ();
exit (1);
}

if (obndrv (&curol1, (text *) ":ol_i_id", -1, (ub1 *) ol_i_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol1);
quit ();
exit (1);
}

```

```

if (obndrv (&curol1, (text *) ":ol_supply_w_id", -1,
(ub1 *) ol_supply_w_id, sizeof (int), SQLT_INT, -1,
(sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol1);
quit ();
exit (1);
}

if (obndrv (&curol1, (text *) ":ol_dist_info", -1, (ub1 *) ol_dist_info,
24, SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol1);
quit ();
exit (1);
}

/* order_line (not delivered) */

if (obndrv (&curol2, (text *) ":ol_o_id", -1, (ub1 *) ol_o_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol2);
quit ();
exit (1);
}

if (obndrv (&curol2, (text *) ":ol_d_id", -1, (ub1 *) ol_d_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol2);
quit ();
exit (1);
}

if (obndrv (&curol2, (text *) ":ol_w_id", -1, (ub1 *) ol_w_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol2);
quit ();
exit (1);
}

if (obndrv (&curol2, (text *) ":ol_number", -1, (ub1 *) ol_number,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol2);
quit ();
exit (1);
}

if (obndrv (&curol2, (text *) ":ol_i_id", -1, (ub1 *) ol_i_id,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol2);
quit ();
exit (1);
}

if (obndrv (&curol2, (text *) ":ol_supply_w_id", -1,
(ub1 *) ol_supply_w_id, sizeof (int), SQLT_INT, -1,
(sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol2);
quit ();
exit (1);
}

if (obndrv (&curol2, (text *) ":ol_amount", -1, (ub1 *) ol_amount,
sizeof (int), SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol2);
quit ();
exit (1);
}

if (obndrv (&curol2, (text *) ":ol_dist_info", -1, (ub1 *) ol_dist_info,
24, SQLT_CHR, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol2);
quit ();
exit (1);
}

/* orders (delivered) */

if (obndrv (&curol1, (text *) ":o_id", -1, (ub1 *) o_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol1);
quit ();
exit (1);
}

if (obndrv (&curol1, (text *) ":o_d_id", -1, (ub1 *) o_d_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol1);
quit ();
exit (1);
}

if (obndrv (&curol1, (text *) ":o_w_id", -1, (ub1 *) o_w_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errprt (&tpclda, &curol1);
quit ();
exit (1);
}

if (obndrv (&curol1, (text *) ":o_c_id", -1, (ub1 *) o_c_id, sizeof (int),
SQLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {

```



```

errrpt (&tpclda, &куро1);
quit ();
exit (1);
}

if (obndrv (&куро1, (text *) "o_carrier_id", -1, (ub1 *) o_carrier_id,
sizeof (int), SFLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &куро1);
quit ();
exit (1);
}

if (obndrv (&куро1, (text *) "o_ol_cnt", -1, (ub1 *) o_ol_cnt,
sizeof (int), SFLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &куро1);
quit ();
exit (1);
}

/* orders (not delivered) */

if (obndrv (&куро2, (text *) "o_id", -1, (ub1 *) o_id, sizeof (int),
SFLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &куро2);
quit ();
exit (1);
}

if (obndrv (&куро2, (text *) "o_d_id", -1, (ub1 *) o_d_id, sizeof (int),
SFLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &куро2);
quit ();
exit (1);
}

if (obndrv (&куро2, (text *) "o_w_id", -1, (ub1 *) o_w_id, sizeof (int),
SFLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &куро2);
quit ();
exit (1);
}

if (obndrv (&куро2, (text *) "o_c_id", -1, (ub1 *) o_c_id, sizeof (int),
SFLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &куро2);
quit ();
exit (1);
}

if (obndrv (&куро2, (text *) "o_ol_cnt", -1, (ub1 *) o_ol_cnt,
sizeof (int), SFLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &куро2);
quit ();
exit (1);
}

/* new order */

if (obndrv (&куроno, (text *) "no_o_id", -1, (ub1 *) no_o_id,
sizeof (int), SFLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &куроno);
quit ();
exit (1);
}

if (obndrv (&куроno, (text *) "no_d_id", -1, (ub1 *) no_d_id,
sizeof (int), SFLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &куроno);
quit ();
exit (1);
}

if (obndrv (&куроno, (text *) "no_w_id", -1, (ub1 *) no_w_id,
sizeof (int), SFLT_INT, -1, (sb2 *) 0, (text *) 0, -1, -1)) {
errrpt (&tpclda, &куроno);
quit ();
exit (1);
}
}

/*-----+
| Initialize random number generator
+-----*/

srand (SEED);
srand48 (SEED);
initperm ();

/*-----+
| Load the WAREHOUSE table.
+-----*/

if (do_A || do_w) {
nrows = aware - bware + 1;

fprintf (stderr, "Loading/generating warehouse: w%d - w%d (%d rows)\n",
bware, aware, nrows);
}

```

```

begin_time = gettimeofday ();
begin_cpu = getcpu ();

for (loop = bware; loop <= aware; loop++) {

w_tax = (rand () % 2001);
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\n", loop, w_tax,
w_name, w_street_1, w_street_2, w_city, str2, num9);
flush (stdout);
}
else {
w_id = loop;
strncpy (w_state, str2, 2);
strncpy (w_zip, num9, 9);

if (oexec (&curw)) {
errrpt (&tpclda, &curw);
orol (&tpclda);
fprintf (stderr, "Aborted at warehouse %d\n", loop);
quit ();
exit (1);
}
else if (ocom (&tpclda)) {
errrpt (&tpclda, &tpclda);
orol (&tpclda);
fprintf (stderr, "Aborted at warehouse %d\n", loop);
quit ();
exit (1);
}
}

end_time = gettimeofday ();
end_cpu = getcpu ();
fprintf (stderr, "Done. %d rows loaded/generated in %10.2f sec. (%10.2f cpu)\n",
nrows, end_time - begin_time, end_cpu - begin_cpu);
}

/*-----+
| Load the DISTRICT table.
+-----*/

if (do_A || do_d) {
nrows = (aware - bware + 1) * DISTFAC;

fprintf (stderr, "Loading/generating district: w%d - w%d (%d rows)\n",
bware, aware, nrows);

begin_time = gettimeofday ();
begin_cpu = getcpu ();

dwid = bware - 1;

for (row = 0; row < nrows; ) {
dwid++;

for (i = 0; i < DISTARR; i++, row++) {
d_tax[i] = (rand () % 2001);
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 %s %s %s %s %s %d 30000.0 3001\n",
i + 1, dwid, d_name[i], d_street_1[i], d_street_2[i],
d_city[i], str2, num9, d_tax[i]); */
/* Reordered columns */
printf ("%d %d 30000000 %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) {
flush (stdout);
}
}
else {
if (oexn (&curd, DISTARR, 0)) {
errrpt (&tpclda, &curd);
orol (&tpclda);
}
}
}

```



```

fprintf(stderr, "Loading/generating stock: w%d - w%d (%d rows)n ",
        bware, aware, nrows);

begin_time = gettimeofday ();
begin_cpu = getcpu ();

sid = 0;
swid = bware;
loopcount = 0;

for (row = 0; row < nrows; ) {
    for (i = 0; i < STOCARR; i++, row++) {
        if (++sid > STOCFAC) { /* cheap mod */
            sid = 1;
            swid++;
        }
        s_quantity[i] = (rand () % 91) + 10;
        randstr (str24[0], 24, 24);
        randstr (str24[1], 24, 24);
        randstr (str24[2], 24, 24);
        randstr (str24[3], 24, 24);
        randstr (str24[4], 24, 24);
        randstr (str24[5], 24, 24);
        randstr (str24[6], 24, 24);
        randstr (str24[7], 24, 24);
        randstr (str24[8], 24, 24);
        randstr (str24[9], 24, 24);
        randdatastr (s_data[i], 26, 50);

        if (gen) {
            printf ("%d %d %d %s %s %s %s %s %s %s %s %s %s %s 0 0 0 %s\n",
                    sid, swid, s_quantity[i], str24[0], str24[1], str24[2],
                    str24[3], str24[4], str24[5], str24[6], str24[7],
                    str24[8], str24[9], s_data[i]);
        }
        else {
            s_i_id[i] = sid;
            s_w_id[i] = swid;
            strncpy (s_dist_01[i], str24[0], 24);
            strncpy (s_dist_02[i], str24[1], 24);
            strncpy (s_dist_03[i], str24[2], 24);
            strncpy (s_dist_04[i], str24[3], 24);
            strncpy (s_dist_05[i], str24[4], 24);
            strncpy (s_dist_06[i], str24[5], 24);
            strncpy (s_dist_07[i], str24[6], 24);
            strncpy (s_dist_08[i], str24[7], 24);
            strncpy (s_dist_09[i], str24[8], 24);
            strncpy (s_dist_10[i], str24[9], 24);
        }
    }
}

if (gen) {
    fflush (stdout);
}
else {
    if (oexn (&curs, STOCARR, 0)) {
        errprt (&tpclda, &curs);
        orol (&tpclda);
        fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
                s_i_id[0]);
        quit ();
        exit (1);
    }
    else if (ocom (&tpclda)) {
        errprt (&tpclda, &tpclda);
        orol (&tpclda);
        fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
                s_i_id[0]);
        quit ();
        exit (1);
    }
}

if ((++loopcount) % 50)
    fprintf (stderr, ".");
else
    fprintf (stderr, "%d rows committed\n ", row);
}

end_time = gettimeofday ();
end_cpu = getcpu ();
fprintf (stderr, "Done. %d rows loaded/generated in %10.2f sec. (%10.2f cpu)\n\n",
        nrows, end_time - begin_time, end_cpu - begin_cpu);
}

/*-----+
| Load the STOCK table (cluster around s_i_id). |
+-----*/

if (do_S) {
    nrows = (citem - bitem + 1) * (eware - bware + 1);

    fprintf (stderr, "Loading/generating stock: i%d - i%d, w%d - w%d (%d rows)n ",
            bitem, eitem, bware, aware, nrows);

    begin_time = gettimeofday ();
    begin_cpu = getcpu ();

```

```

sid = bitem;
swid = bware - 1;
loopcount = 0;

for (row = 0; row < nrows; ) {
    for (i = 0; i < STOCARR; i++, row++) {
        if (++swid > aware) { /* cheap mod */
            swid = bware;
            sid++;
        }
        s_quantity[i] = (rand () % 91) + 10;
        randstr (str24[0], 24, 24);
        randstr (str24[1], 24, 24);
        randstr (str24[2], 24, 24);
        randstr (str24[3], 24, 24);
        randstr (str24[4], 24, 24);
        randstr (str24[5], 24, 24);
        randstr (str24[6], 24, 24);
        randstr (str24[7], 24, 24);
        randstr (str24[8], 24, 24);
        randstr (str24[9], 24, 24);
        randdatastr (s_data[i], 26, 50);

        if (gen) {
            printf ("%d %d %d %s %s %s %s %s %s %s %s %s %s %s 0 0 0 %s\n",
                    sid, swid, s_quantity[i], str24[0], str24[1], str24[2],
                    str24[3], str24[4], str24[5], str24[6], str24[7],
                    str24[8], str24[9], s_data[i]);
        }
        else {
            s_i_id[i] = sid;
            s_w_id[i] = swid;
            strncpy (s_dist_01[i], str24[0], 24);
            strncpy (s_dist_02[i], str24[1], 24);
            strncpy (s_dist_03[i], str24[2], 24);
            strncpy (s_dist_04[i], str24[3], 24);
            strncpy (s_dist_05[i], str24[4], 24);
            strncpy (s_dist_06[i], str24[5], 24);
            strncpy (s_dist_07[i], str24[6], 24);
            strncpy (s_dist_08[i], str24[7], 24);
            strncpy (s_dist_09[i], str24[8], 24);
            strncpy (s_dist_10[i], str24[9], 24);
        }
    }
}

if (gen) {
    fflush (stdout);
}
else {
    if (oexn (&curs, STOCARR, 0)) {
        errprt (&tpclda, &curs);
        orol (&tpclda);
        fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
                s_i_id[0]);
        quit ();
        exit (1);
    }
    else if (ocom (&tpclda)) {
        errprt (&tpclda, &tpclda);
        orol (&tpclda);
        fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
                s_i_id[0]);
        quit ();
        exit (1);
    }
}

if ((++loopcount) % 50)
    fprintf (stderr, ".");
else
    fprintf (stderr, "%d rows committed\n ", row);
}

end_time = gettimeofday ();
end_cpu = getcpu ();
fprintf (stderr, "Done. %d rows loaded/generated in %10.2f sec. (%10.2f cpu)\n\n",
        nrows, end_time - begin_time, end_cpu - begin_cpu);
}

/*-----+
| Load the HISTORY table. |
+-----*/

if (do_A || do_h) {
    nrows = (aware - bware + 1) * HISTFAC;

    fprintf (stderr, "Loading/generating history: w%d - w%d (%d rows)n ",
            bware, aware, nrows);

    begin_time = gettimeofday ();
    begin_cpu = getcpu ();

    cid = 0;
    cidid = 1;
    cwid = bware;
    loopcount = 0;

    for (row = 0; row < nrows; ) {
        for (i = 0; i < HISTARR; i++, row++) {

```



<pre> } } else { if (oexn (&amp;curo2, ORDEARR, 0)) { errprt (&amp;tpclda, &amp;curo2); orol (&amp;tpclda); fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n ", cwid, cdid, cid); quit (); exit (1); } else if (ocom (&amp;tpclda)) { errprt (&amp;tpclda, &amp;tpclda); orol (&amp;tpclda); fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n ", cwid, cdid, cid); quit (); exit (1); } } }  if ((++loopcount) % 50) fprintf (stderr, "."); else fprintf (stderr, " %d orders committed\n ", row); }  end_time = gettime (); end_cpu = getcpu (); fprintf (stderr, "Done. %d orders loaded/generated in %10.2f sec. (%10.2f cpu)\n\n", nrows, end_time - begin_time, end_cpu - begin_cpu); }  +-----+   Load the NEW-ORDER table.   +-----*/  if (do_A    do_n) { nrows = (eware - bware + 1) * NEWOFAC * DISTFAC;  fprintf (stderr, "Loading/generating new-order: w%d - w%d (%d rows)\n ", bware, ewart, nrows);  begin_time = gettime (); begin_cpu = getcpu ();  cid = 0; cdid = 1; cwid = bware; loopcount = 0;  for (row = 0; row &lt; nrows; ) { for (i = 0; i &lt; NEWOARR; i++, row++) { cid++; if (cid &gt; NEWOFAC) { cid = 1; cdid++; if (cdid &gt; 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 { if (oexn (&amp;curno, NEWOARR, 0)) { errprt (&amp;tpclda, &amp;curno); orol (&amp;tpclda); fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n ", cwid, cdid, cid + 2100); quit (); exit (1); } else if (ocom (&amp;tpclda)) { errprt (&amp;tpclda, &amp;tpclda); orol (&amp;tpclda); fprintf (stderr, "Aborted at w_id %d, d_id %d, o_id %d\n ", cwid, cdid, cid + 2100); quit (); exit (1); } } }  if ((++loopcount) % 45) fprintf (stderr, "."); else </pre>	<pre> 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); }  initperm () { int i; int pos; int temp;  /* init randperm3000 */  for (i = 0; i &lt; 3000; i++) randperm3000[i] = i + 1; for (i = 3000; i &gt; 0; i--) { pos = rand () % i; temp = randperm3000[i - 1]; randperm3000[i - 1] = randperm3000[pos]; randperm3000[pos] = temp; } }  randstr (str, x, y)  char *str; int x; int y;  { int i, j; int len;  len = (rand () % (y - x + 1)) + x; for (i = 0; i &lt; len; i++) { j = rand () % 62; if (j &lt; 26) str[i] = (char) (j + 'a'); else if (j &lt; 52) str[i] = (char) (j - 26 + 'A'); else str[i] = (char) (j - 52 + '0'); } str[len] = '\0'; }  randdatastr (str, x, y)  char *str; int x; int y;  { int i, j; int len; int pos;  len = (rand () % (y - x + 1)) + x; for (i = 0; i &lt; len; i++) { j = rand () % 62; if (j &lt; 26) str[i] = (char) (j + 'a'); else if (j &lt; 52) str[i] = (char) (j - 26 + 'A'); else str[i] = (char) (j - 52 + '0'); } str[len] = '\0'; if ((rand () % 10) == 0) { pos = (rand () % (len - 8)); str[pos] = 'O'; } } </pre>
--	---

```

str[pos + 1] = 'R';
str[pos + 2] = 'I';
str[pos + 3] = 'G';
str[pos + 4] = 'T';
str[pos + 5] = 'N';
str[pos + 6] = 'A';
str[pos + 7] = 'L';
}
}

randnum (str, len)

char *str;
int len;

{
    int i;

    for (i = 0; i < len; i++)
        str[i] = (char) (rand () % 10 + '0');
    str[len] = '\0';
}

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]);
}

NURand (A, x, y, cnum)

int A, x, y, cnum;

{
    int a, b;

    a = Irand48 () % (A + 1);
    b = (Irand48 () % (y - x + 1)) + x;
    return (((a | b) + cnum) % (y - x + 1)) + x;
}

sysdate (sdate)

char *sdate;

{
    time_t tp;
    struct tm *tmptr;

    time (&tp);
    tmptr = localtime (&tp);
    strftime (sdate, 29, "%d-%b-%Y", tmptr);
}

```

## C4. Loader

### dist.ctl

-- \$Header: cust.ctl 7030100.1 95/08/07 15:46:36 plai Generic<base> \$ Copyr (c) 1994 Oracle

```

-----+-----
--          Copyright (c) 1994 Oracle Corp. Redwood Shores, CA
--          OPEN SYSTEMS PERFORMANCE GROUP
--          All Rights Reserved
-----+-----

-- FILENAME
-- cust.ctl
-- DESCRIPTION
-- This is a SQL*Loader control file. It is used for
-- loading customers to the tpcc database.
-- USAGE
-- sqlldr[st] <user_name> </password> <SQL*Loader control file>
-----+-----

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE district
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
    D_ID                integer external,
    D_W_ID              integer external,
    D_YTD               integer external,
    D_TAX               integer external,
    D_NEXT_O_ID         integer external,
    D_NAME              CHAR(10),
    D_STREET_1          CHAR(20),
    D_STREET_2          CHAR(20),
    D_CITY              CHAR(20),
    D_STATE             CHAR(2),
    D_ZIP               CHAR(9)
)

Hist.ctl

--
-- $Header: hist.ctl 7030100.1 95/08/07 15:47:23 plai Generic<base> $ Copyr (c) 1994 Oracle
--
-----+-----
--          Copyright (c) 1994 Oracle Corp. Redwood Shores, CA
--          OPEN SYSTEMS PERFORMANCE GROUP
--          All Rights Reserved
-----+-----

-- FILENAME
-- hist.ctl
-- DESCRIPTION
-- This is a SQL*Loader control file. It is used for
-- loading history rows to the tpcc database.
-- USAGE
-- sqlldr[st] <user_name> </password> <SQL*Loader control file>
-----+-----

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE history
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
    h_c_id              integer external,
    h_c_d_id            integer external,
    h_c_w_id            integer external,
    h_d_id              integer external,
    h_w_id              integer external,
    h_date              date "DD-Mon-YYYY",
    h_amount            integer external,
    h_data              char(24)
)

Neword.ctl

--
-- $Header: neword.ctl 7030100.1 95/08/07 15:47:44 plai Generic<base> $ Copyr (c) 1994 Oracle
--
-----+-----
--          Copyright (c) 1994 Oracle Corp. Redwood Shores, CA
--          OPEN SYSTEMS PERFORMANCE GROUP
--          All Rights Reserved
-----+-----

-- FILENAME
-- neword.ctl
-- DESCRIPTION
-- This is a SQL*Loader control file. It is used for
-- loading new orders to the tpcc database.

```

```
-- USAGE
-- sqlldr[st] <user_name>/<password> <SQL*Loader control file>
=====*/

OPTIONS (DIRECT = TRUE, PARALLEL = FALSE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE new_order
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  no_o_id      integer external,
  no_d_id      integer external,
  no_w_id      integer external
)


```

## Order.ctl

```
-- $Header: order.ctl 7030100.1 95/08/07 15:54:01 plai Generic<base> $ Copyr (c) 1994 Oracle
-----
Copyright (c) 1994 Oracle Corp, Redwood Shores, CA |
OPEN SYSTEMS PERFORMANCE GROUP |
All Rights Reserved |
-----
FILENAME
-- order.ctl
DESCRIPTION
-- This is a SQL*Loader control file. It is used for
-- loading orders to the tpc database.
USAGE
-- sqlldr[st] <user_name>/<password> <SQL*Loader control file>
=====*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE orders
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  o_id      integer external,
  o_d_id    integer external,
  o_w_id    integer external,
  o_c_id    integer external,
  o_entry_d date "DD-Mon-YYYY",
  o_carrier_id integer external,
  o_ol_cnt  integer external,
  o_all_local integer external
)


```

## Ordline.ctl

```
-- $Header: ordline.ctl 7030100.1 95/08/07 15:54:11 plai Generic<base> $ Copyr (c) 1994 Oracle
-----
Copyright (c) 1994 Oracle Corp, Redwood Shores, CA |
OPEN SYSTEMS PERFORMANCE GROUP |
All Rights Reserved |
-----
FILENAME
-- ordline.ctl
DESCRIPTION
-- This is a SQL*Loader control file. It is used for
-- loading order lines to the tpc database.
USAGE
-- sqlldr[st] <user_name>/<password> <SQL*Loader control file>
=====*/

OPTIONS (DIRECT = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE order_line
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  ol_o_id      integer external,
  ol_d_id      integer external,
  ol_w_id      integer external,
  ol_number    integer external,
```

```
  ol_delivery_d date "DD-Mon-YYYY",
  ol_i_id       integer external,
  ol_supply_w_id integer external,
  ol_quantity   integer external,
  ol_amount     integer external,
  ol_dist_info  char(24)
)


```

## Stock.ctl

```
--
-- $Header: stock.ctl 7030100.1 95/08/07 15:54:18 plai Osd<base> $ Copyr (c) 1994 Oracle
-----
Copyright (c) 1994 Oracle Corp, Redwood Shores, CA |
OPEN SYSTEMS PERFORMANCE GROUP |
All Rights Reserved |
-----
FILENAME
-- stock.ctl
DESCRIPTION
-- This is a SQL*Loader control file. It is used for
-- loading stocks to the tpc database.
USAGE
-- sqlldr[st] <user_name>/<password> <SQL*Loader control file>
=====*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE stock
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  s_i_id      integer external,
  s_w_id      integer external,
  s_quantity  integer external,
  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),
  s_ytd       integer external,
  s_order_cnt integer external,
  s_remote_cnt integer external,
  s_data      char(50)
)


```

## Ware.ctl

```
--
-- $Header: cust.ctl 7030100.1 95/08/07 15:46:36 plai Generic<base> $ Copyr (c) 1994 Oracle
-----
Copyright (c) 1994 Oracle Corp, Redwood Shores, CA |
OPEN SYSTEMS PERFORMANCE GROUP |
All Rights Reserved |
-----
FILENAME
-- cust.ctl
DESCRIPTION
-- This is a SQL*Loader control file. It is used for
-- loading customers to the tpc database.
USAGE
-- sqlldr[st] <user_name>/<password> <SQL*Loader control file>
=====*/

OPTIONS (DIRECT = TRUE, PARALLEL = TRUE)

UNRECOVERABLE

LOAD DATA
APPEND

INTO TABLE warehouse
APPEND
FIELDS TERMINATED BY WHITESPACE OPTIONALLY ENCLOSED BY ""
(
  W_ID      integer external,
  W_YTD     integer external,
  W_TAX     float external,
  W_NAME    CHAR(10),
  W_STREET_1 CHAR(20)
```

W\_STREET\_2  
W\_CITY  
W\_STATE  
W\_ZIP  
)

CHAR(20)  
CHAR(20)  
CHAR(2)  
CHAR(9)



# Appendix D: RTE Scripts

## D.1 RTE Parameters

\* For Oracle in the tpcload program C\_LAST =1. C-delta be the difference \*/  
\*between C-LOAD and C-Run. C-Delta must be a value between 65..119 including the \*/  
\* values of 65 and 119 and excluding the value of 96 and 112 \*/

LASTC=86  
MEASUREMENT="1"

```
#define MASTER_NUM1 1  
#define MASTER_NUM2 0  
#define MASTER_NUM3 0
```

```
#if MASTER_NUM1  
MASTER "jeopardy.austin.ibm.com"  
#elif MASTER_NUM2  
MASTER "plasma.austin.ibm.com"  
#elif MASTER_NUM3  
MASTER "ozarka.austin.ibm.com"  
#endif
```

SUT = "plum.austin.ibm.com"

```
#if MASTER_NUM1  
MAIN_CLIENT = client1  
SLAVES driver1,driver2,driver3,driver4,driver5,driver6,driver7,driver8,driver9,driver10  
CLIENT_REAL = "client1 client2 client3 client4 client5"
```

```
#elif MASTER_NUM2  
MAIN_CLIENT = client8  
SLAVES driver15a,driver15b,driver15c,driver15d,driver15e,driver15f,driver15g,driver15h,driver15i,  
driver15j,driver15k,driver15l,driver16a,driver16b,driver16c,driver16d,driver16e,driver16f,driver16g,driver16h,  
driver16i,driver16j,driver16k,driver16l,driver16m,driver16n,driver16o,driver16p,driver16q,driver16r,driver16s,  
driver16t,driver16u,driver16v,driver16w,driver16x,driver16y,driver16z,driver17a,driver17b,driver17c,driver17d,  
driver17e,driver17f,driver17g,driver17h,driver17i,driver17j,driver17k,driver17l,driver17m,driver17n,driver17o,  
driver17p,driver17q,driver17r,driver17s,driver17t,driver17u,driver17v,driver17w,driver17x,driver17y,driver17z,  
driver18a,driver18b,driver18c,driver18d,driver18e,driver18f,driver18g,driver18h,driver18i,driver18j,driver18k,  
driver18l,driver18m,driver18n,driver18o,driver18p,driver18q,driver18r,driver18s,driver18t,driver18u,driver18v,  
driver18w,driver18x,driver18y,driver18z,driver19a,driver19b,driver19c,driver19d,driver19e,driver19f,driver19g,  
driver19h,driver19i,driver19j,driver19k,driver19l,driver19m,driver19n,driver19o,driver19p,driver19q,driver19r,  
driver19s,driver19t,driver19u,driver19v,driver19w,driver19x,driver19y,driver19z,driver20a,driver20b,driver20c,  
driver20d,driver20e,driver20f,driver20g,driver20h,driver20i,driver20j,driver20k,driver20l,driver20m,driver20n,  
driver20o,driver20p,driver20q,driver20r,driver20s,driver20t,driver20u,driver20v,driver20w,driver20x,driver20y,  
driver20z,driver21a,driver21b,driver21c,driver21d,driver21e,driver21f,driver21g,driver21h,driver21i,driver21j,  
driver21k,driver21l,driver21m,driver21n,driver21o,driver21p,driver21q,driver21r,driver21s,driver21t,driver21u,  
driver21v,driver21w,driver21x,driver21y,driver21z,driver22a,driver22b,driver22c,driver22d,driver22e,driver22f,  
driver22g,driver22h,driver22i,driver22j,driver22k,driver22l,driver22m,driver22n,driver22o,driver22p,driver22q,  
driver22r,driver22s,driver22t,driver22u,driver22v,driver22w,driver22x,driver22y,driver22z,driver23a,driver23b,  
driver23c,driver23d,driver23e,driver23f,driver23g,driver23h,driver23i,driver23j,driver23k,driver23l,driver23m,  
driver23n,driver23o,driver23p,driver23q,driver23r,driver23s,driver23t,driver23u,driver23v,driver23w,driver23x,  
driver23y,driver23z,driver24a,driver24b,driver24c,driver24d,driver24e,driver24f,driver24g,driver24h,driver24i,  
driver24j,driver24k,driver24l,driver24m,driver24n,driver24o,driver24p,driver24q,driver24r,driver24s,driver24t,  
driver24u,driver24v,driver24w,driver24x,driver24y,driver24z,driver25a,driver25b,driver25c,driver25d,driver25e,  
driver25f,driver25g,driver25h,driver25i,driver25j,driver25k,driver25l,driver25m,driver25n,driver25o,driver25p,  
driver25q,driver25r,driver25s,driver25t,driver25u,driver25v,driver25w,driver25x,driver25y,driver25z,driver26a,  
driver26b,driver26c,driver26d,driver26e,driver26f,driver26g,driver26h,driver26i
```

CLIENT\_REAL = "client8 client9 client10 client11 client12 client13"

```
#elif MASTER_NUM3  
MAIN_CLIENT = client6  
SLAVES driver11,driver12,driver13,driver14,driver27,driver28,driver29,driver30,driver31  
CLIENT_REAL = "client6 client7 client14 client15"  
#endif
```

```
#if MASTER_NUM1  
CLIENT client1w1 oracle orifldb  
CLIENT client1x1 oracle orifldb  
CLIENT client1y1 oracle orifldb  
CLIENT client1z1 oracle orifldb  
CLIENT client1w2 oracle orifldb  
CLIENT client1x2 oracle orifldb  
CLIENT client1y2 oracle orifldb
```

```
CLIENT client2w3 oracle orifldb  
CLIENT client2x3 oracle orifldb  
CLIENT client2y3 oracle orifldb  
CLIENT client2z3 oracle orifldb  
CLIENT client2w4 oracle orifldb  
CLIENT client2x4 oracle orifldb  
CLIENT client2y4 oracle orifldb
```

```
CLIENT client3w5 oracle orifldb  
CLIENT client3x5 oracle orifldb  
CLIENT client3y5 oracle orifldb  
CLIENT client3z5 oracle orifldb  
CLIENT client3w6 oracle orifldb  
CLIENT client3x6 oracle orifldb  
CLIENT client3y6 oracle orifldb
```

```
CLIENT client4w7 oracle orifldb  
CLIENT client4x7 oracle orifldb  
CLIENT client4y7 oracle orifldb  
CLIENT client4z7 oracle orifldb  
CLIENT client4w8 oracle orifldb  
CLIENT client4x8 oracle orifldb  
CLIENT client4y8 oracle orifldb
```

```
CLIENT client5w9 oracle orifldb  
CLIENT client5x9 oracle orifldb  
CLIENT client5y9 oracle orifldb  
CLIENT client5z9 oracle orifldb  
CLIENT client5w10 oracle orifldb
```

```
CLIENT client5x10 oracle orifldb  
CLIENT client5y10 oracle orifldb
```

```
#elif MASTER_NUM3  
CLIENT client6w11 oracle orifldb  
CLIENT client6x11 oracle orifldb  
CLIENT client6y11 oracle orifldb  
CLIENT client6z11 oracle orifldb  
CLIENT client6w12 oracle orifldb  
CLIENT client6x12 oracle orifldb  
CLIENT client6y12 oracle orifldb  
CLIENT client6z12 oracle orifldb
```

```
CLIENT client7w13 oracle orifldb  
CLIENT client7x13 oracle orifldb  
CLIENT client7y13 oracle orifldb  
CLIENT client7z13 oracle orifldb  
CLIENT client7w14 oracle orifldb  
CLIENT client7x14 oracle orifldb  
CLIENT client7y14 oracle orifldb  
CLIENT client7z14 oracle orifldb
```

```
#elif MASTER_NUM2  
CLIENT client8w15 oracle orifldb  
CLIENT client8x15 oracle orifldb  
CLIENT client8y15 oracle orifldb  
CLIENT client8z15 oracle orifldb  
CLIENT client8w16 oracle orifldb  
CLIENT client8x16 oracle orifldb  
CLIENT client8y16 oracle orifldb
```

```
CLIENT client9w17 oracle orifldb  
CLIENT client9x17 oracle orifldb  
CLIENT client9y17 oracle orifldb  
CLIENT client9z17 oracle orifldb  
CLIENT client9w18 oracle orifldb  
CLIENT client9x18 oracle orifldb  
CLIENT client9y18 oracle orifldb
```

```
CLIENT client10w19 oracle orifldb  
CLIENT client10x19 oracle orifldb  
CLIENT client10y19 oracle orifldb  
CLIENT client10z19 oracle orifldb  
CLIENT client10w20 oracle orifldb  
CLIENT client10x20 oracle orifldb  
CLIENT client10y20 oracle orifldb
```

```
CLIENT client11w21 oracle orifldb  
CLIENT client11x21 oracle orifldb  
CLIENT client11y21 oracle orifldb  
CLIENT client11z21 oracle orifldb  
CLIENT client11w22 oracle orifldb  
CLIENT client11x22 oracle orifldb  
CLIENT client11y22 oracle orifldb
```

```
CLIENT client12w23 oracle orifldb  
CLIENT client12x23 oracle orifldb  
CLIENT client12y23 oracle orifldb  
CLIENT client12z23 oracle orifldb  
CLIENT client12w24 oracle orifldb  
CLIENT client12x24 oracle orifldb  
CLIENT client12y24 oracle orifldb
```

```
CLIENT client13w25 oracle orifldb  
CLIENT client13x25 oracle orifldb  
CLIENT client13y25 oracle orifldb  
CLIENT client13z25 oracle orifldb  
CLIENT client13w26 oracle orifldb  
CLIENT client13x26 oracle orifldb  
CLIENT client13y26 oracle orifldb  
#endif
```

```
#if MASTER_NUM3  
CLIENT client14w27 oracle orifldb  
CLIENT client14x27 oracle orifldb  
CLIENT client14y27 oracle orifldb  
CLIENT client14z27 oracle orifldb  
CLIENT client14w28 oracle orifldb  
CLIENT client14x28 oracle orifldb  
CLIENT client14y28 oracle orifldb  
CLIENT client14z28 oracle orifldb
```

```
CLIENT client15w29 oracle orifldb  
CLIENT client15x29 oracle orifldb  
CLIENT client15y29 oracle orifldb  
CLIENT client15z29 oracle orifldb  
CLIENT client15w30 oracle orifldb  
CLIENT client15x30 oracle orifldb  
CLIENT client15y30 oracle orifldb  
CLIENT client15z30 oracle orifldb  
#endif
```

```
TELNET telnet 23  
SOCKET socket 199703
```

WAREHOUSES=10700

```
#if MASTER_NUM1  
SOCKET_NETWORK socket1 6700 driver1
```



```

SOCKET_NETWORK socket185 6716 driver14
SOCKET_NETWORK socket186 6717 driver14
SOCKET_NETWORK socket187 6718 driver14
SOCKET_NETWORK socket188 6719 driver14
SOCKET_NETWORK socket189 6720 driver14
SOCKET_NETWORK socket190 6721 driver14
SOCKET_NETWORK socket191 6722 driver14
SOCKET_NETWORK socket192 6723 driver14
SOCKET_NETWORK socket193 6724 driver14
SOCKET_NETWORK socket194 6725 driver14
SOCKET_NETWORK socket195 6726 driver14
SOCKET_NETWORK socket196 6727 driver14

#elif MASTER_NUM2
/*AMIDALA*/
SOCKET_NETWORK socket148 6700 driver15a
SOCKET_NETWORK socket149 6701 driver15b
SOCKET_NETWORK socket150 6702 driver15c
SOCKET_NETWORK socket151 6703 driver15d
SOCKET_NETWORK socket152 6704 driver15e
SOCKET_NETWORK socket153 6705 driver15f
SOCKET_NETWORK socket154 6706 driver15g
SOCKET_NETWORK socket155 6707 driver15h
SOCKET_NETWORK socket156 6708 driver15i
SOCKET_NETWORK socket157 6709 driver15j
SOCKET_NETWORK socket158 6710 driver15k
SOCKET_NETWORK socket159 6711 driver15l

SOCKET_NETWORK socket160 6712 driver16a
SOCKET_NETWORK socket161 6713 driver16b
SOCKET_NETWORK socket162 6714 driver16c
SOCKET_NETWORK socket163 6715 driver16d
SOCKET_NETWORK socket164 6716 driver16e
SOCKET_NETWORK socket165 6717 driver16f
SOCKET_NETWORK socket166 6718 driver16g
SOCKET_NETWORK socket167 6719 driver16h
SOCKET_NETWORK socket168 6720 driver16i

SOCKET_NETWORK socket169 6700 driver17a
SOCKET_NETWORK socket170 6701 driver17b
SOCKET_NETWORK socket171 6702 driver17c
SOCKET_NETWORK socket172 6703 driver17d
SOCKET_NETWORK socket173 6704 driver17e
SOCKET_NETWORK socket174 6705 driver17f
SOCKET_NETWORK socket175 6706 driver17g
SOCKET_NETWORK socket176 6707 driver17h
SOCKET_NETWORK socket177 6708 driver17i
SOCKET_NETWORK socket178 6709 driver17j
SOCKET_NETWORK socket179 6710 driver17k
SOCKET_NETWORK socket180 6711 driver17l

SOCKET_NETWORK socket181 6712 driver18a
SOCKET_NETWORK socket182 6713 driver18b
SOCKET_NETWORK socket183 6714 driver18c
SOCKET_NETWORK socket184 6715 driver18d
SOCKET_NETWORK socket185 6716 driver18e
SOCKET_NETWORK socket186 6717 driver18f
SOCKET_NETWORK socket187 6718 driver18g
SOCKET_NETWORK socket188 6719 driver18h
SOCKET_NETWORK socket189 6720 driver18i

SOCKET_NETWORK socket190 6700 driver19a
SOCKET_NETWORK socket191 6701 driver19b
SOCKET_NETWORK socket192 6702 driver19c
SOCKET_NETWORK socket193 6703 driver19d
SOCKET_NETWORK socket194 6704 driver19e
SOCKET_NETWORK socket195 6705 driver19f
SOCKET_NETWORK socket196 6706 driver19g
SOCKET_NETWORK socket197 6707 driver19h
SOCKET_NETWORK socket198 6708 driver19i
SOCKET_NETWORK socket199 6709 driver19j
SOCKET_NETWORK socket200 6710 driver19k
SOCKET_NETWORK socket201 6711 driver19l

SOCKET_NETWORK socket202 6712 driver20a
SOCKET_NETWORK socket203 6713 driver20b
SOCKET_NETWORK socket204 6714 driver20c
SOCKET_NETWORK socket205 6715 driver20d
SOCKET_NETWORK socket206 6716 driver20e
SOCKET_NETWORK socket207 6717 driver20f
SOCKET_NETWORK socket208 6718 driver20g
SOCKET_NETWORK socket209 6719 driver20h
SOCKET_NETWORK socket210 6720 driver20i
/*LEIA*/
SOCKET_NETWORK socket211 6700 driver21a
SOCKET_NETWORK socket212 6701 driver21b
SOCKET_NETWORK socket213 6702 driver21c
SOCKET_NETWORK socket214 6703 driver21d
SOCKET_NETWORK socket215 6704 driver21e
SOCKET_NETWORK socket216 6705 driver21f
SOCKET_NETWORK socket217 6706 driver21g
SOCKET_NETWORK socket218 6707 driver21h
SOCKET_NETWORK socket219 6708 driver21i
SOCKET_NETWORK socket220 6709 driver21j
SOCKET_NETWORK socket221 6710 driver21k
SOCKET_NETWORK socket222 6711 driver21l

SOCKET_NETWORK socket223 6712 driver22a
SOCKET_NETWORK socket224 6713 driver22b

SOCKET_NETWORK socket225 6714 driver22c
SOCKET_NETWORK socket226 6715 driver22d
SOCKET_NETWORK socket227 6716 driver22e
SOCKET_NETWORK socket228 6717 driver22f
SOCKET_NETWORK socket229 6718 driver22g
SOCKET_NETWORK socket230 6719 driver22h
SOCKET_NETWORK socket231 6720 driver22i

SOCKET_NETWORK socket232 6700 driver23a
SOCKET_NETWORK socket233 6701 driver23b
SOCKET_NETWORK socket234 6702 driver23c
SOCKET_NETWORK socket235 6703 driver23d
SOCKET_NETWORK socket236 6704 driver23e
SOCKET_NETWORK socket237 6705 driver23f
SOCKET_NETWORK socket238 6706 driver23g
SOCKET_NETWORK socket239 6707 driver23h
SOCKET_NETWORK socket240 6708 driver23i
SOCKET_NETWORK socket241 6709 driver23j
SOCKET_NETWORK socket242 6710 driver23k
SOCKET_NETWORK socket243 6711 driver23l

SOCKET_NETWORK socket244 6712 driver24a
SOCKET_NETWORK socket245 6713 driver24b
SOCKET_NETWORK socket246 6714 driver24c
SOCKET_NETWORK socket247 6715 driver24d
SOCKET_NETWORK socket248 6716 driver24e
SOCKET_NETWORK socket249 6717 driver24f
SOCKET_NETWORK socket250 6718 driver24g
SOCKET_NETWORK socket251 6719 driver24h
SOCKET_NETWORK socket252 6720 driver24i

SOCKET_NETWORK socket253 6700 driver25a
SOCKET_NETWORK socket254 6701 driver25b
SOCKET_NETWORK socket255 6702 driver25c
SOCKET_NETWORK socket256 6703 driver25d
SOCKET_NETWORK socket257 6704 driver25e
SOCKET_NETWORK socket258 6705 driver25f
SOCKET_NETWORK socket259 6706 driver25g
SOCKET_NETWORK socket260 6707 driver25h
SOCKET_NETWORK socket261 6708 driver25i
SOCKET_NETWORK socket262 6709 driver25j
SOCKET_NETWORK socket263 6710 driver25k
SOCKET_NETWORK socket264 6711 driver25l

SOCKET_NETWORK socket265 6712 driver26a
SOCKET_NETWORK socket266 6713 driver26b
SOCKET_NETWORK socket267 6714 driver26c
SOCKET_NETWORK socket268 6715 driver26d
SOCKET_NETWORK socket269 6716 driver26e
SOCKET_NETWORK socket270 6717 driver26f
SOCKET_NETWORK socket271 6718 driver26g
SOCKET_NETWORK socket272 6719 driver26h
SOCKET_NETWORK socket273 6720 driver26i

#endif

#if MASTER_NUM3
/*2nd set*/
SOCKET_NETWORK socket365 6700 driver27
SOCKET_NETWORK socket366 6701 driver27
SOCKET_NETWORK socket367 6702 driver27
SOCKET_NETWORK socket368 6703 driver27
SOCKET_NETWORK socket369 6704 driver27
SOCKET_NETWORK socket370 6705 driver27
SOCKET_NETWORK socket371 6706 driver27
SOCKET_NETWORK socket372 6707 driver27
SOCKET_NETWORK socket373 6708 driver27
SOCKET_NETWORK socket374 6709 driver27
SOCKET_NETWORK socket375 6710 driver27
SOCKET_NETWORK socket376 6711 driver27
SOCKET_NETWORK socket377 6712 driver27
SOCKET_NETWORK socket378 6713 driver27
SOCKET_NETWORK socket379 6714 driver27
SOCKET_NETWORK socket380 6715 driver27

SOCKET_NETWORK socket381 6716 driver28
SOCKET_NETWORK socket382 6717 driver28
SOCKET_NETWORK socket383 6718 driver28
SOCKET_NETWORK socket384 6719 driver28
SOCKET_NETWORK socket385 6720 driver28
SOCKET_NETWORK socket386 6721 driver28
SOCKET_NETWORK socket387 6722 driver28
SOCKET_NETWORK socket388 6723 driver28
SOCKET_NETWORK socket389 6724 driver28
SOCKET_NETWORK socket390 6725 driver28
SOCKET_NETWORK socket391 6726 driver28
SOCKET_NETWORK socket392 6727 driver28

SOCKET_NETWORK socket393 6700 driver29
SOCKET_NETWORK socket394 6701 driver29
SOCKET_NETWORK socket395 6702 driver29
SOCKET_NETWORK socket396 6703 driver29
SOCKET_NETWORK socket397 6704 driver29
SOCKET_NETWORK socket398 6705 driver29
SOCKET_NETWORK socket399 6706 driver29
SOCKET_NETWORK socket400 6707 driver29
SOCKET_NETWORK socket401 6708 driver29
SOCKET_NETWORK socket402 6709 driver29
SOCKET_NETWORK socket403 6710 driver29

```

<pre> SOCKET_NETWORK socket404 6711 driver29 SOCKET_NETWORK socket405 6712 driver29 SOCKET_NETWORK socket406 6713 driver29 SOCKET_NETWORK socket407 6714 driver29 SOCKET_NETWORK socket408 6715 driver29  SOCKET_NETWORK socket409 6716 driver30 SOCKET_NETWORK socket410 6717 driver30 SOCKET_NETWORK socket411 6718 driver30 SOCKET_NETWORK socket412 6719 driver30 SOCKET_NETWORK socket413 6720 driver30 SOCKET_NETWORK socket414 6721 driver30 SOCKET_NETWORK socket415 6722 driver30 SOCKET_NETWORK socket416 6723 driver30 SOCKET_NETWORK socket417 6724 driver30 SOCKET_NETWORK socket418 6725 driver30 SOCKET_NETWORK socket419 6726 driver30 SOCKET_NETWORK socket420 6727 driver30  SOCKET_NETWORK socket421 6728 driver31 SOCKET_NETWORK socket422 6729 driver31 SOCKET_NETWORK socket423 6728 driver31 SOCKET_NETWORK socket424 6729 driver31 SOCKET_NETWORK socket425 6728 driver31 SOCKET_NETWORK socket426 6729 driver31 SOCKET_NETWORK socket427 6728 driver31 SOCKET_NETWORK socket428 6729 driver31  #endif  OUTPUTNAME="./runs/condor"  CPU=24  #if 0 BEGIN_WAIT=25:00 RAMPUP=30:00 RUNTIME=30:00 RAMPDOWN_WAIT=5:00 RAMPDOWN=15:00 #else BEGIN_WAIT=22:30 RAMPUP=56:00 RUNTIME=30:00 RAMPDOWN_WAIT=5:00 RAMPDOWN=15:00 #endif INTERVAL=1:00 /* Interval to calculate mix from */  LOGIN_MAX_LOAD = 4 LOGIN_BEGIN = 0 /* skip login state if set to 1 */ NOBEGIN = 1 KEYSTROKE_PACKET_SIZE = 0  MAX_CONCURRENT_SPAWN = 10 SPAWN_COUNT = 5  MIN_PORT = 8088 MAX_PORT = 8089  /* User variables. Think, Emulex Delay, %desired, %min, %max */ #if 0 NEWORDER = "12.03, 0, 0" PAYMENT = "12.03, 0, 0, 43.07, 43.07, 43.07 " ORDSTAT = "10.03, 0, 0, 4.05, 4.05, 4.05 " DELIVERY = "05.03, 0, 0, 4.05, 4.05, 4.05 " STOCKLEV = "05.03, 0, 0, 4.05, 4.05, 4.05 " #else NEWORDER = "12.01, 0, 0" PAYMENT = "12.01, 0, 0, 43.01, 43.01, 43.01 " ORDSTAT = "10.01, 0, 0, 4.01, 4.01, 4.01 " DELIVERY = "05.01, 0, 0, 4.01, 4.01, 4.01 " STOCKLEV = "05.01, 0, 0, 4.01, 4.01, 4.01 " #endif  #if MASTER_NUM1 START_RANGE client1w1 socket1 250 0-25 START_RANGE client1x1 socket2 250 25-50 START_RANGE client1y1 socket3 250 50-75 START_RANGE client1z1 socket4 250 75-100 START_RANGE client1w1 socket5 250 100-125 START_RANGE client1x1 socket6 250 125-150 START_RANGE client1y1 socket7 250 150-175 START_RANGE client1z1 socket8 250 175-200 START_RANGE client1w1 socket9 250 200-225 START_RANGE client1x1 socket10 250 225-250 START_RANGE client1y1 socket11 250 250-275 START_RANGE client1z1 socket12 250 275-300 START_RANGE client1w1 socket13 250 300-325 START_RANGE client1x1 socket14 250 325-350 START_RANGE client1y1 socket15 250 350-375 START_RANGE client1z1 socket16 250 375-400  START_RANGE client1w2 socket17 250 400-425 START_RANGE client1x2 socket18 250 425-450 START_RANGE client1y2 socket19 250 450-475 START_RANGE client1w2 socket20 250 475-500 START_RANGE client1x2 socket21 250 500-525 START_RANGE client1y2 socket22 250 525-550 </pre>	<pre> START_RANGE client1w2 socket23 250 550-575 START_RANGE client1x2 socket24 250 575-600 START_RANGE client1y2 socket25 250 600-625 START_RANGE client1w2 socket26 250 625-650 START_RANGE client1x2 socket27 250 650-675 START_RANGE client1y2 socket28 250 675-700  START_RANGE client2w3 socket29 250 700-725 START_RANGE client2x3 socket30 250 725-750 START_RANGE client2y3 socket31 250 750-775 START_RANGE client2z3 socket32 250 775-800 START_RANGE client2w3 socket33 250 800-825 START_RANGE client2x3 socket34 250 825-850 START_RANGE client2y3 socket35 250 850-875 START_RANGE client2z3 socket36 250 875-900 START_RANGE client2w3 socket37 250 900-925 START_RANGE client2x3 socket38 250 925-950 START_RANGE client2y3 socket39 250 950-975 START_RANGE client2z3 socket40 250 975-1000 START_RANGE client2w3 socket41 250 1000-1025 START_RANGE client2x3 socket42 250 1025-1050 START_RANGE client2y3 socket43 250 1050-1075 START_RANGE client2z3 socket44 250 1075-1100  START_RANGE client2w4 socket45 250 1100-1125 START_RANGE client2x4 socket46 250 1125-1150 START_RANGE client2y4 socket47 250 1150-1175 START_RANGE client2w4 socket48 250 1175-1200 START_RANGE client2x4 socket49 250 1200-1225 START_RANGE client2y4 socket50 250 1225-1250 START_RANGE client2w4 socket51 250 1250-1275 START_RANGE client2x4 socket52 250 1275-1300 START_RANGE client2y4 socket53 250 1300-1325 START_RANGE client2w4 socket54 250 1325-1350 START_RANGE client2x4 socket55 250 1350-1375 START_RANGE client2y4 socket56 250 1375-1400  START_RANGE client3w5 socket57 250 1400-1425 START_RANGE client3x5 socket58 250 1425-1450 START_RANGE client3y5 socket59 250 1450-1475 START_RANGE client3z5 socket60 250 1475-1500 START_RANGE client3w5 socket61 250 1500-1525 START_RANGE client3x5 socket62 250 1525-1550 START_RANGE client3y5 socket63 250 1550-1575 START_RANGE client3z5 socket64 250 1575-1600 START_RANGE client3w5 socket65 250 1600-1625 START_RANGE client3x5 socket66 250 1625-1650 START_RANGE client3y5 socket67 250 1650-1675 START_RANGE client3z5 socket68 250 1675-1700 START_RANGE client3w5 socket69 250 1700-1725 START_RANGE client3x5 socket70 250 1725-1750 START_RANGE client3y5 socket71 250 1750-1775 START_RANGE client3z5 socket72 250 1775-1800  START_RANGE client3w6 socket73 250 1800-1825 START_RANGE client3x6 socket74 250 1825-1850 START_RANGE client3y6 socket75 250 1850-1875 START_RANGE client3w6 socket76 250 1875-1900 START_RANGE client3x6 socket77 250 1900-1925 START_RANGE client3y6 socket78 250 1925-1950 START_RANGE client3w6 socket79 250 1950-1975 START_RANGE client3x6 socket80 250 1975-2000 START_RANGE client3y6 socket81 250 2000-2025 START_RANGE client3w6 socket82 250 2025-2050 START_RANGE client3x6 socket83 250 2050-2075 START_RANGE client3y6 socket84 250 2075-2100  START_RANGE client4w7 socket85 250 2100-2125 START_RANGE client4x7 socket86 250 2125-2150 START_RANGE client4y7 socket87 250 2150-2175 START_RANGE client4z7 socket88 250 2175-2200 START_RANGE client4w7 socket89 250 2200-2225 START_RANGE client4x7 socket90 250 2225-2250 START_RANGE client4y7 socket91 250 2250-2275 START_RANGE client4z7 socket92 250 2275-2300 START_RANGE client4w7 socket93 250 2300-2325 START_RANGE client4x7 socket94 250 2325-2350 START_RANGE client4y7 socket95 250 2350-2375 START_RANGE client4z7 socket96 250 2375-2400 START_RANGE client4w7 socket97 250 2400-2425 START_RANGE client4x7 socket98 250 2425-2450 START_RANGE client4y7 socket99 250 2450-2475 START_RANGE client4z7 socket100 250 2475-2500  START_RANGE client4w8 socket101 250 2500-2525 START_RANGE client4x8 socket102 250 2525-2550 START_RANGE client4y8 socket103 250 2550-2575 START_RANGE client4w8 socket104 250 2575-2600 START_RANGE client4x8 socket105 250 2600-2625 START_RANGE client4y8 socket106 250 2625-2650 START_RANGE client4w8 socket107 250 2650-2675 START_RANGE client4x8 socket108 250 2675-2700 START_RANGE client4y8 socket109 250 2700-2725 START_RANGE client4w8 socket110 250 2725-2750 START_RANGE client4x8 socket111 250 2750-2775 START_RANGE client4y8 socket112 250 2775-2800  START_RANGE client5w9 socket113 250 2800-2825 START_RANGE client5x9 socket114 250 2825-2850 </pre>
--	---

```

START_RANGE client5y9 socket115 250 2850-2875
START_RANGE client5z9 socket116 250 2875-2900
START_RANGE client5w9 socket117 250 2900-2925
START_RANGE client5x9 socket118 250 2925-2950
START_RANGE client5y9 socket119 250 2950-2975
START_RANGE client5z9 socket120 250 2975-3000
START_RANGE client5w9 socket121 250 3000-3025
START_RANGE client5x9 socket122 250 3025-3050
START_RANGE client5y9 socket123 250 3050-3075
START_RANGE client5z9 socket124 250 3075-3100
START_RANGE client5w9 socket125 250 3100-3125
START_RANGE client5x9 socket126 250 3125-3150
START_RANGE client5y9 socket127 250 3150-3175
START_RANGE client5z9 socket128 250 3175-3200

START_RANGE client5w10 socket129 250 3200-3225
START_RANGE client5x10 socket130 250 3225-3250
START_RANGE client5y10 socket131 250 3250-3275
START_RANGE client5w10 socket132 250 3275-3300
START_RANGE client5x10 socket133 250 3300-3325
START_RANGE client5y10 socket134 250 3325-3350
START_RANGE client5w10 socket135 250 3350-3375
START_RANGE client5x10 socket136 250 3375-3400
START_RANGE client5y10 socket137 250 3400-3425
START_RANGE client5w10 socket138 250 3425-3450
START_RANGE client5x10 socket139 250 3450-3475
START_RANGE client5y10 socket140 250 3475-3500

#elif MASTER_NUM3

START_RANGE client6w11 socket141 250 3500-3525
START_RANGE client6x11 socket142 250 3525-3550
START_RANGE client6y11 socket143 250 3550-3575
START_RANGE client6z11 socket144 250 3575-3600
START_RANGE client6w11 socket145 250 3600-3625
START_RANGE client6x11 socket146 250 3625-3650
START_RANGE client6y11 socket147 250 3650-3675
START_RANGE client6z11 socket148 250 3675-3700
START_RANGE client6w11 socket149 250 3700-3725
START_RANGE client6x11 socket150 250 3725-3750
START_RANGE client6y11 socket151 250 3750-3775
START_RANGE client6z11 socket152 250 3775-3800
START_RANGE client6w11 socket153 250 3800-3825
START_RANGE client6x11 socket154 250 3825-3850
START_RANGE client6y11 socket155 250 3850-3875
START_RANGE client6z11 socket156 250 3875-3900

START_RANGE client6w12 socket157 250 3900-3925
START_RANGE client6x12 socket158 250 3925-3950
START_RANGE client6y12 socket159 250 3950-3975
START_RANGE client6w12 socket160 250 3975-4000
START_RANGE client6x12 socket161 250 4000-4025
START_RANGE client6y12 socket162 250 4025-4050
START_RANGE client6w12 socket163 250 4050-4075
START_RANGE client6x12 socket164 250 4075-4100
START_RANGE client6y12 socket165 250 4100-4125
START_RANGE client6w12 socket166 250 4125-4150
START_RANGE client6x12 socket167 250 4150-4175
START_RANGE client6y12 socket168 250 4175-4200

START_RANGE client6v31 socket421 250 10500-10525
START_RANGE client6v31 socket422 250 10525-10550

START_RANGE client7w13 socket169 250 4200-4225
START_RANGE client7x13 socket170 250 4225-4250
START_RANGE client7y13 socket171 250 4250-4275
START_RANGE client7z13 socket172 250 4275-4300
START_RANGE client7w13 socket173 250 4300-4325
START_RANGE client7x13 socket174 250 4325-4350
START_RANGE client7y13 socket175 250 4350-4375
START_RANGE client7z13 socket176 250 4375-4400
START_RANGE client7w13 socket177 250 4400-4425
START_RANGE client7x13 socket178 250 4425-4450
START_RANGE client7y13 socket179 250 4450-4475
START_RANGE client7z13 socket180 250 4475-4500
START_RANGE client7w13 socket181 250 4500-4525
START_RANGE client7x13 socket182 250 4525-4550
START_RANGE client7y13 socket183 250 4550-4575
START_RANGE client7z13 socket184 250 4575-4600

START_RANGE client7w14 socket185 250 4600-4625
START_RANGE client7x14 socket186 250 4625-4650
START_RANGE client7y14 socket187 250 4650-4675
START_RANGE client7w14 socket188 250 4675-4700
START_RANGE client7x14 socket189 250 4700-4725
START_RANGE client7y14 socket190 250 4725-4750
START_RANGE client7w14 socket191 250 4750-4775
START_RANGE client7x14 socket192 250 4775-4800
START_RANGE client7y14 socket193 250 4800-4825
START_RANGE client7w14 socket194 250 4825-4850
START_RANGE client7x14 socket195 250 4850-4875
START_RANGE client7y14 socket196 250 4875-4900

START_RANGE client7v31 socket423 250 10550-10575
START_RANGE client7v31 socket424 250 10575-10600

#elif MASTER_NUM2

START_RANGE client8w15 socket148 330 4900-4933

```

```

START_RANGE client8x15 socket149 330 4933-4966
START_RANGE client8y15 socket150 330 4966-5000
START_RANGE client8z15 socket151 330 5000-5033
START_RANGE client8w15 socket152 330 5033-5066
START_RANGE client8x15 socket153 330 5066-5100
START_RANGE client8y15 socket154 330 5100-5133
START_RANGE client8z15 socket155 330 5133-5166
START_RANGE client8w15 socket156 330 5166-5200
START_RANGE client8x15 socket157 330 5200-5233
START_RANGE client8y15 socket158 330 5233-5266
START_RANGE client8z15 socket159 330 5266-5300

START_RANGE client8w16 socket160 330 5300-5333
START_RANGE client8x16 socket161 330 5333-5366
START_RANGE client8y16 socket162 330 5366-5400
START_RANGE client8z16 socket163 330 5400-5433
START_RANGE client8w16 socket164 330 5433-5466
START_RANGE client8x16 socket165 330 5466-5500
START_RANGE client8y16 socket166 330 5500-5533
START_RANGE client8z16 socket167 330 5533-5566
START_RANGE client8w16 socket168 330 5566-5600

START_RANGE client9w17 socket169 330 5600-5633
START_RANGE client9x17 socket170 330 5633-5666
START_RANGE client9y17 socket171 330 5666-5700
START_RANGE client9z17 socket172 330 5700-5733
START_RANGE client9w17 socket173 330 5733-5766
START_RANGE client9x17 socket174 330 5766-5800
START_RANGE client9y17 socket175 330 5800-5833
START_RANGE client9z17 socket176 330 5833-5866
START_RANGE client9w17 socket177 330 5866-5900
START_RANGE client9x17 socket178 330 5900-5933
START_RANGE client9y17 socket179 330 5933-5966
START_RANGE client9z17 socket180 330 5966-6000

START_RANGE client9w18 socket181 330 6000-6033
START_RANGE client9x18 socket182 330 6033-6066
START_RANGE client9y18 socket183 330 6066-6100
START_RANGE client9z18 socket184 330 6100-6133
START_RANGE client9w18 socket185 330 6133-6166
START_RANGE client9x18 socket186 330 6166-6200
START_RANGE client9y18 socket187 330 6200-6233
START_RANGE client9z18 socket188 330 6233-6266
START_RANGE client9w18 socket189 330 6266-6300

START_RANGE client10w19 socket190 330 6300-6333
START_RANGE client10x19 socket191 330 6333-6366
START_RANGE client10y19 socket192 330 6366-6400
START_RANGE client10z19 socket193 330 6400-6433
START_RANGE client10w19 socket194 330 6433-6466
START_RANGE client10x19 socket195 330 6466-6500
START_RANGE client10y19 socket196 330 6500-6533
START_RANGE client10z19 socket197 330 6533-6566
START_RANGE client10w19 socket198 330 6566-6600
START_RANGE client10x19 socket199 330 6600-6633
START_RANGE client10y19 socket200 330 6633-6666
START_RANGE client10z19 socket201 330 6666-6700

START_RANGE client10w20 socket202 330 6700-6733
START_RANGE client10x20 socket203 330 6733-6766
START_RANGE client10y20 socket204 330 6766-6800
START_RANGE client10z20 socket205 330 6800-6833
START_RANGE client10w20 socket206 330 6833-6866
START_RANGE client10x20 socket207 330 6866-6900
START_RANGE client10y20 socket208 330 6900-6933
START_RANGE client10z20 socket209 330 6933-6966
START_RANGE client10w20 socket210 330 6966-7000

START_RANGE client11w21 socket211 330 7000-7033
START_RANGE client11x21 socket212 330 7033-7066
START_RANGE client11y21 socket213 330 7066-7100
START_RANGE client11z21 socket214 330 7100-7133
START_RANGE client11w21 socket215 330 7133-7166
START_RANGE client11x21 socket216 330 7166-7200
START_RANGE client11y21 socket217 330 7200-7233
START_RANGE client11z21 socket218 330 7233-7266
START_RANGE client11w21 socket219 330 7266-7300
START_RANGE client11x21 socket220 330 7300-7333
START_RANGE client11y21 socket221 330 7333-7366
START_RANGE client11z21 socket222 330 7366-7400

START_RANGE client11w22 socket223 330 7400-7433
START_RANGE client11x22 socket224 330 7433-7466
START_RANGE client11y22 socket225 330 7466-7500
START_RANGE client11z22 socket226 330 7500-7533
START_RANGE client11w22 socket227 330 7533-7566
START_RANGE client11x22 socket228 330 7566-7600
START_RANGE client11y22 socket229 330 7600-7633
START_RANGE client11z22 socket230 330 7633-7666
START_RANGE client11w22 socket231 330 7666-7700

START_RANGE client12w23 socket232 330 7700-7733
START_RANGE client12x23 socket233 330 7733-7766
START_RANGE client12y23 socket234 330 7766-7800
START_RANGE client12z23 socket235 330 7800-7833
START_RANGE client12w23 socket236 330 7833-7866
START_RANGE client12x23 socket237 330 7866-7900
START_RANGE client12y23 socket238 330 7900-7933
START_RANGE client12z23 socket239 330 7933-7966

```

```

START_RANGE client12w23 socket240 340 7966-8000
START_RANGE client12x23 socket241 330 8000-8033
START_RANGE client12y23 socket242 330 8033-8066
START_RANGE client12z23 socket243 340 8066-8100

START_RANGE client12w24 socket244 330 8100-8133
START_RANGE client12x24 socket245 330 8133-8166
START_RANGE client12y24 socket246 340 8166-8200
START_RANGE client12w24 socket247 330 8200-8233
START_RANGE client12x24 socket248 330 8233-8266
START_RANGE client12y24 socket249 340 8266-8300
START_RANGE client12w24 socket250 330 8300-8333
START_RANGE client12x24 socket251 330 8333-8366
START_RANGE client12y24 socket252 340 8366-8400

START_RANGE client13w25 socket253 330 8400-8433
START_RANGE client13x25 socket254 330 8433-8466
START_RANGE client13y25 socket255 340 8466-8500
START_RANGE client13z25 socket256 330 8500-8533
START_RANGE client13w25 socket257 330 8533-8566
START_RANGE client13x25 socket258 340 8566-8600
START_RANGE client13y25 socket259 330 8600-8633
START_RANGE client13z25 socket260 330 8633-8666
START_RANGE client13w25 socket261 340 8666-8700
START_RANGE client13x25 socket262 330 8700-8733
START_RANGE client13y25 socket263 330 8733-8766
START_RANGE client13z25 socket264 340 8766-8800

START_RANGE client13w26 socket265 330 8800-8833
START_RANGE client13x26 socket266 330 8833-8866
START_RANGE client13y26 socket267 340 8866-8900
START_RANGE client13w26 socket268 330 8900-8933
START_RANGE client13x26 socket269 330 8933-8966
START_RANGE client13y26 socket270 340 8966-9000
START_RANGE client13w26 socket271 330 9000-9033
START_RANGE client13x26 socket272 330 9033-9066
START_RANGE client13y26 socket273 340 9066-9100

#endif
#if MASTER_NUM3

START_RANGE client14w27 socket365 250 9100-9125
START_RANGE client14x27 socket366 250 9125-9150
START_RANGE client14y27 socket367 250 9150-9175
START_RANGE client14z27 socket368 250 9175-9200
START_RANGE client14w27 socket369 250 9200-9225
START_RANGE client14x27 socket370 250 9225-9250
START_RANGE client14y27 socket371 250 9250-9275
START_RANGE client14z27 socket372 250 9275-9300
START_RANGE client14w27 socket373 250 9300-9325
START_RANGE client14x27 socket374 250 9325-9350
START_RANGE client14y27 socket375 250 9350-9375
START_RANGE client14z27 socket376 250 9375-9400
START_RANGE client14w27 socket377 250 9400-9425
START_RANGE client14x27 socket378 250 9425-9450
START_RANGE client14y27 socket379 250 9450-9475
START_RANGE client14z27 socket380 250 9475-9500

START_RANGE client14w28 socket381 250 9500-9525
START_RANGE client14x28 socket382 250 9525-9550
START_RANGE client14y28 socket383 250 9550-9575
START_RANGE client14z28 socket384 250 9575-9600
START_RANGE client14x28 socket385 250 9600-9625
START_RANGE client14y28 socket386 250 9625-9650
START_RANGE client14w28 socket387 250 9650-9675
START_RANGE client14x28 socket388 250 9675-9700
START_RANGE client14y28 socket389 250 9700-9725
START_RANGE client14w28 socket390 250 9725-9750
START_RANGE client14x28 socket391 250 9750-9775
START_RANGE client14y28 socket392 250 9775-9800

START_RANGE client14v31 socket425 250 10600-10625
START_RANGE client14v31 socket426 250 10625-10650

START_RANGE client15w29 socket393 250 9800-9825
START_RANGE client15x29 socket394 250 9825-9850
START_RANGE client15y29 socket395 250 9850-9875
START_RANGE client15z29 socket396 250 9875-9900
START_RANGE client15w29 socket397 250 9900-9925
START_RANGE client15x29 socket398 250 9925-9950
START_RANGE client15y29 socket399 250 9950-9975
START_RANGE client15z29 socket400 250 9975-10000
START_RANGE client15w29 socket401 250 10000-10025
START_RANGE client15x29 socket402 250 10025-10050
START_RANGE client15y29 socket403 250 10050-10075
START_RANGE client15z29 socket404 250 10075-10100
START_RANGE client15w29 socket405 250 10100-10125
START_RANGE client15x29 socket406 250 10125-10150
START_RANGE client15y29 socket407 250 10150-10175
START_RANGE client15z29 socket408 250 10175-10200

START_RANGE client15w30 socket409 250 10200-10225
START_RANGE client15x30 socket410 250 10225-10250
START_RANGE client15y30 socket411 250 10250-10275
START_RANGE client15z30 socket412 250 10275-10300
START_RANGE client15x30 socket413 250 10300-10325
START_RANGE client15y30 socket414 250 10325-10350
START_RANGE client15w30 socket415 250 10350-10375
START_RANGE client15x30 socket416 250 10375-10400

```

```

START_RANGE client15y30 socket417 250 10400-10425
START_RANGE client15w30 socket418 250 10425-10450
START_RANGE client15x30 socket419 250 10450-10475
START_RANGE client15y30 socket420 250 10475-10500

START_RANGE client15v31 socket427 250 10650-10675
START_RANGE client15v31 socket428 250 10675-10700

#endif

#define TES_FLAG_TRACE 0x00000010
#define TES_FLAG_KEYSTROKE_TIME 0x00000200
#define TES_FLAG_LOCAL_LOG 0x00000400
#define TES_FLAG_LOCAL_TRACE 0x00000800
#define TES_FLAG_LOCAL_IPRINT 0x00004000

#if 0
/* SETFLAG ALL TES_FLAG_TRACE */
SETFLAG ALL TES_FLAG_LOCAL_TRACE
SETFLAG ALL TES_FLAG_LOCAL_IPRINT
#endif

#if 0
SETFLAG client1x telnet 1 TES_FLAG_KEYSTROKE_TIME
#endif

```

## D.2 user master.C

```

/*****
 * user_master.C Audit: 05/30/96 */
*****/

static char *rcsid="$Id: user_master.C,v 1.1 1999/02/22 06:31:05 channui Exp $";

#include <iostream.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#define H_CUR01
#include <cur00.h>
#undef H_CUR01
extern "C" {
#include "data/cur01.h"
int wrefresh (WINDOW *);
int wclrtoeol(WINDOW *);
int setupterm(char*,FILE*,int*);
int nodelay(int);
int keypad(int);
int wgetch(WINDOW *);
}
#include "data/rte.h"
#include "data/95t5.h"
#include "data/misc.h"
#include "user_tpc.h"

struct header_s {
int slave;
int num;
int type;
int num_timestamps;
int user_data_length;
int data_type;
};

char *get_variable(char *name);
int get_variable(char *name, int *number);
int send_global_data(void);
int make_ratios (double *buffer);
extern int ramp_up_complete;
extern int interval_start_time, interval_stop_time;
extern "C" int strcasecmp(char *s1, char *s2);
extern "C" int strncasecmp(char *s1, char *s2, int n);

struct UserSpawnData {
int Warehouse;
int District;
};

/* user_master.C */
int user_statistics_print(void);
// int user_spawn(int *length, char *buffer);
int user_spawn(int min, int max, int number, int *length, char *buffer);
int user_finished(int length, char *buffer);

extern SlaveStatus slave_status[MAX_SLAVES];

extern Stats status[MAX_TRAN_TYPE][MAX_TIMES];
extern WINDOW *statistics_win;
extern UserGlobal *shmglobal;

```

```

/* Transaction mix parameters */
double ratio_desired[6], ratio_min[6], ratio_max[6], ratio_range[6];
char *ratio_names[] = { "RTE", "NEWORDER", "PAYMENT", "ORDSTAT", "DELIVERY",
    "STOCKLEV", NULL };
char *Status_Names[] = {"Menu", "Keying", "Response", "Think"};

char *transaction_names[] = { "RTE", "New Order", "Payment", "Order Stat",
    "Delivery", "Stock Level", NULL };

static int current_status = 2, status_needs_refresh = 1;

int user_statistics_print(void) {
    int i;
    static int count = 0;
    double ratios[6];
    if (status_needs_refresh) {
        count = 0;
        status_needs_refresh = 0;
        wmove (statistics_win, 0, 0);
        wprintw (statistics_win, "%11s %8s %8s %8s %8s %6s %6s %6s",
            Status_Names[current_status], "90%", "Avg", "Min", "Max",
            "Samples", "Ratio", "Mix", "Think");
    }
    make_ratios(ratios);

    for (i = 1; i <= 5; i++) {
        /* The reason we do this is because calculating the percentiles
           is expensive */
        if (count % 10 == 0) {
            wmove (statistics_win, i, 0);
            wprintw (statistics_win, "%11s %8.2f",
                transaction_names[i], status[i][current_status].ninety()/1000.0);
            count = 0;
        }
        wmove (statistics_win, i, 21);
        wprintw (statistics_win, "%8.2f %8.2f %8.2f %8d %6.2f %6.2f %6.2f",
            status[i][current_status].average()/1000.0,
            status[i][current_status].min()/1000.0,
            status[i][current_status].max()/1000.0,
            status[i][current_status].samples(),
            ratios[i], shmglobal->chances[i],
            status[i][3].average()/1000.0);
    }
    wmove (statistics_win, 7, 0);

    extern int runtime_counts[MAX_TRAN_TYPE];
    extern int begin_time, ramp_up, run_time;
    int start = interval_start_time;
    int stop = interval_stop_time;
    double interval = ((double)(stop-start) / (1000*60));
    double samples = status[1][2].samples();
    if (interval <= 0 || samples <= 0) {
        wprintw (statistics_win, "TPM-C: %7s / ", "-----");
    } else {
        wprintw (statistics_win, "TPM-C: %7.2f / ", samples/interval);
    }
    samples = runtime_counts[1];
    if (samples > 0) {
        start = begin_time + ((ramp_up >= 0) ? ramp_up : 0);
        if (run_time > 0 && stop > begin_time + ramp_up + run_time) {
            stop = begin_time + ramp_up + run_time;
        }
        interval = (double)(stop - start) / (1000.0*60.0);
        wprintw (statistics_win, "%7.2f", samples/interval);
    } else {
        wprintw (statistics_win, "-----");
    }

    count++;
    return RTE_OK;
}

extern int login_begin;
int login_max_load;

const int MAX_WAREHOUSES=20000;
/* All of this 10 stuff is district size. Should be a constant.
   Maybe fix that later */
int num_warehouses = -1;
int warehouses[MAX_WAREHOUSES*10];
int user_spawn(int min, int max, int number, int *length, char *buffer) {
    /*int user_spawn(int number, int *length, char *buffer) {
    int i, min_index;
    int adj_wh = num_warehouses; // adjusted warehouse number
    UserSpawnData *ptr = (UserSpawnData *)buffer;
    *length = sizeof(ptr);

    // min_index = 0;
    // for (i = 1; i < (num_warehouses)*10 && i < MAX_WAREHOUSES*10; i++) {
    //
    // if both min and max are zero, running START, otherwise running
    // START_RANGE. Must also determine what the ending warehouse number
    // will be for said range
    //
    if (min == 0 && max == 0) {

```

```

        min++;
        min_index = 0;
    } else {
        adj_wh = max; // inclusive range of wh-s
        min = min * 10;
        min_index = min;
    }
    for (i = min ; i < (adj_wh)*10 && i < MAX_WAREHOUSES*10; i++) {
        if (warehouses[i] < warehouses[min_index]) {
            min_index = i;
        }
    }

    ptr->Warehouse = min_index / 10 + 1;
    ptr->District = min_index % 10 + 1;
    warehouses[min_index]++;
    /* iprint (IPRINT_INFO, "Driver for Warehouse %d, District %d started. warehouses[%d] += %d\n",
        ptr->Warehouse, ptr->District, min_index, warehouses[min_index]); */
    return RTE_OK;
}

int user_finished(int length, char *buffer) {
    UserSpawnData *ptr = (UserSpawnData *)buffer;
    int temp = (ptr->Warehouse-1)*10+ptr->District-1;
    warehouses[temp]--;
    /* iprint (IPRINT_INFO, "Driver for Warehouse %d, District %d died. warehouses[%d] -= %d\n",
        ptr->Warehouse, ptr->District, temp, warehouses[temp]); */
    return RTE_OK;
}

double limit(double min, double max, double val) {
    if (val < min)
        return min;
    if (val > max)
        return max;
    return val;
}

int make_ratios (double *buffer) {
    int neword = status[NEWORDER][0].samples();
    int payment = status[PAYMENT][0].samples();
    int ordstat = status[ORDSTAT][0].samples();
    int delivery = status[DELIVERY][0].samples();
    int stocklev = status[STOCKLEV][0].samples();
    int total = neword + payment + ordstat + delivery + stocklev;
    int i;

    if (total == 0) {
        buffer[NEWORDER] = 100.0;
        for (i = 2; i < 6; i++) {
            buffer[i] = ratio_desired[i];
            buffer[NEWORDER] -= buffer[i];
        }
        return 0;
    }

    buffer[PAYMENT] = (double)payment / (double)total * 100.0;
    buffer[ORDSTAT] = (double)ordstat / (double)total * 100.0;
    buffer[DELIVERY] = (double)delivery / (double)total * 100.0;
    buffer[STOCKLEV] = (double)stocklev / (double)total * 100.0;
    buffer[NEWORDER] = 100.0 - buffer[PAYMENT] - buffer[ORDSTAT] -
        buffer[DELIVERY] - buffer[STOCKLEV];

    return total;
}

int user_global_update(int *length, char *buffer) {
    UserGlobal *shmglobal = (UserGlobal *)buffer;
    static double last[6];
    static last_test_state = 0;
    static int users_last=1;
    double ratios[6];
    double current[6];
    int i, different = 0;
    int desired = 0;
    int host_busy, all_zero;

    *length = sizeof(*shmglobal);

    make_ratios(ratios);

    /* Calculate ratios we want for next time */
    /* Note: we just keep on with the desired values until ramp-up is complete
       this at least starts us out without any humps or spikes in the
       graph */
    if (ramp_up_complete) {
        current[NEWORDER] = 100.0;
        for (i = 2; i < 6; i++) {
            if (ratio_desired[i] > ratios[i]) {
                current[i] = ratio_max[i];
            } else {
                current[i] = 2*ratio_desired[i] - ratios[i];
                if (current[i] < ratio_min[i])
                    current[i] = ratio_min[i];
            }
        }
        current[NEWORDER] -= current[i];
    }
}

```

```

} else {
    for (i = 1; i < 6; i++) {
        current[i] = ratio_desired[i];
    }
}

/* Add up all the users */
/* This needs to be changed to be more transparent */
shmglobal->total_users = 0;
for (i = 0; i < MAX_SLAVES; i++) {
    shmglobal->total_users += slave_status[i].active;
    desired += slave_status[i].desired;
}

/* Count up number of warehouses we WANT to have */
if (num_warehouses < 0) {
    num_warehouses = (desired-1)/10+1;
}
shmglobal->max_warehouses = num_warehouses;

host_busy = 0;
all_zero = 1;
for (i = 1; i <= 5; i++) {
    if (status[i][current_status].average() != 0) {
        all_zero = 0;
    }
    if ( status[i][current_status].average()/1000.0 > login_max_load ) {
        host_busy = 1;
    }
}
if (shmglobal->host_busy && all_zero) {
    host_busy = 1;
}

if (host_busy != shmglobal->host_busy) {
    shmglobal->host_busy = host_busy;
    different = 1;
}

for (i = 2; i < 6; i++) {
    if (current[i] != last[i])
        different = 1;
}

if (last_test_state != shmglobal->test_state) {
    different = 1;
    last_test_state = shmglobal->test_state;
}

// Don't send if it's the same as last time
if ( !different && shmglobal->total_users == users_last ) {
    return RTE_ERROR;
}

users_last = shmglobal->total_users;
for (i = 1; i < 6; i++) {
    shmglobal->chances[i] = last[i] = current[i];
}

return RTE_OK;
}

int user_isbusy() {
    return shmglobal->host_busy;
}

int parse_array(char *string, int max, int *buffer) {
    int i, rc;
    char *ptr;
    char *temp = strdup(string);
    ptr = strtok(temp, ",");
    for (i = 0; ptr && i < max; i++) {
        rc = sscanf(ptr, "%d", &buffer[i]);
        if (rc < 1) {
            free(temp);
            return i;
        }
        ptr = strtok(NULL, ",");
    }
    free(temp);
    return i;
}

int parse_array(char *string, int max, double *buffer) {
    int i, rc;
    char *ptr;
    char *temp = strdup(string);
    ptr = strtok(temp, ",");
    for (i = 0; ptr && i < max; i++) {
        rc = sscanf(ptr, "%lf", &buffer[i]);
        if (rc < 1) {
            free(temp);
            return i;
        }
        ptr = strtok(NULL, ",");
    }
    free(temp);
    return i;
}

int user_init() {
    double dbuffer[32];
    int rc, i;
    char *ptr;

    if (get_variable("KEYSTROKE_SLEEP", &shmglobal->keystroke_sleep) != RTE_OK) {
        shmglobal->keystroke_sleep = 0;
    }
    if (get_variable("LOGIN_TIMEOUT", &shmglobal->login_timeout) != RTE_OK) {
        shmglobal->login_timeout = 120; /* 2 minutes */
    }
    if (get_variable("KEYSTROKE_PACKET_SIZE", &shmglobal->keystroke_packet_size) != RTE_OK) {
        shmglobal->keystroke_packet_size = 0;
    }
    shmglobal->login_timeout *= 1000;
    if (get_variable("LOGIN_MAX_LOAD", &login_max_load) != RTE_OK) {
        login_max_load = 2;
    }
    if (get_variable("WAREHOUSES", &num_warehouses) != RTE_OK) {
        num_warehouses = -1;
    }
    if (get_variable("LASTC", &shmglobal->lastc) != RTE_OK) {
        shmglobal->lastc = 193; /* 2 minutes */
    }
    fprintf(IPRINT_INFO, "Login Timeout = %s\n", mstoa(shmglobal->login_timeout, 0));
    fprintf(IPRINT_INFO, "Keystroke Sleep = %s\n", mstoa(shmglobal->keystroke_sleep*1000, 0));
    fprintf(IPRINT_INFO, "Keystroke Packet Size= %d\n", shmglobal->keystroke_packet_size);
    if (num_warehouses >= 0) {
        fprintf(IPRINT_INFO, "Fixed Warehouses to = %d\n", num_warehouses);
    }
}

if (!ptr = get_variable("NEWORDER")) {
    fprintf_error ("Error. NEWORDER variable not found\n");
    exit (1);
}

if (parse_array(ptr, 3, dbuffer)!=3) {
    fprintf_error ("Error. NEWORDER should be think, emulex_menu, emulex_response");
    exit (1);
}

shmglobal->think [NEWORDER] = dbuffer[0];
shmglobal->emulex_menu [NEWORDER] = dbuffer[1];
shmglobal->emulex_response[NEWORDER] = dbuffer[2];
shmglobal->test_state = 0;

for (i = 2; i < 6; i++) {
    if (!ptr = get_variable(ratio_names[i])) ||
        (parse_array(ptr, 6, dbuffer)!=6)) {
        fprintf(_FILE_, _LINE_, IPRINT_ERROR,
            "Error. %s should be think, emulex_menu, emulex_response, desired, min, max",
            ratio_names[i]);
        exit (1);
    }
    shmglobal->think[i] = dbuffer[0];
    shmglobal->emulex_menu[i] = dbuffer[1];
    shmglobal->emulex_response[i] = dbuffer[2];
    ratio_desired[i] = dbuffer[3];
    ratio_min[i] = dbuffer[4];
    ratio_max[i] = dbuffer[5];
    ratio_range[i] = ratio_max[i]-ratio_min[i];
}

return RTE_OK;
}

int user_extra_data(header_s *header) {
    int i;
    int num_timestamps;

    if (header->data_type != RTE_ITEM_KEYSTROKE_TIMES)
        return RTE_OK;
    int *times = (int *)((char *)header+sizeof(struct header_s));
    num_timestamps = header->user_data_length / 4 - 1;

    fprintf (IPRINT_TRACE, "Keystroke times = ");
    for (i = 0; i < num_timestamps; i++) {
        fprintf (IPRINT_TRACE, "%d ", times[i]);
    }
    fprintf (IPRINT_TRACE, "\n", times[i]);

    return RTE_OK;
}

int user_process_command(char *command) {
    char buffer[256], *ptr;
    int i, found, len;
    strncpy (buffer, command, 256);
    ptr = strtok (buffer, " \t");
    found = 0;
    printf ("user_process_command(%s)\n", ptr);
    if (strcmp(ptr, "pause") == 0) {
        shmglobal->test_state = 1;
    } else if (strcmp(ptr, "warmup") == 0) {
        shmglobal->test_state = 2;
    } else if (strcmp(ptr, "notest") == 0) {
        shmglobal->test_state = 0;
    } else if (strcmp(ptr, "login_max_load?") == 0) {
        fprintf (IPRINT_WARNING, "Current LOGIN_MAX_LOAD = %d\n", login_max_load);
    }
}

```



```

} else if (!strncasecmp (command, "login_max_load=", 15)) {
    login_max_load=atoi(command+15);
    iprint (IPRINT_WARNING, "Set LOGIN_MAX_LOAD = %d\n", login_max_load);
} else if (!strcasecmp (ptr, "display")) {
    while (ptr && (ptr = strtok(NULL, "\t"))) {
        if (*ptr == "0")
            continue;
        for (i = 0; i < 5; i++) {
            len = min(strlen(Status_Names[i]), strlen(ptr));
            if (!strncasecmp (ptr, Status_Names[i], len)) {
                status_needs_refresh = found = 1;
                current_status = i;
                return RTE_OK;
            }
        }
        iprint (IPRINT_WARNING, "Unknown type to display: %s\n", ptr);
    }
} else {
    iprint (IPRINT_WARNING, "Unknown Command: %s\n", command);
    return RTE_ERROR;
}
return RTE_OK;
}

int transaction_process () {
    return RTE_OK;
}

int user_begin() {
    return RTE_OK;
}

void user_make_header(char *buffer) {
    int i;
    struct user_data_header *data = (struct user_data_header *)buffer;
}

```

### D.3 user\_slave.C

```

*****
/* user_slave.C                      Audit: 05/30/96 */
*****

static char *resid="$Id: user_slave.C,v 1.1 1999/02/22 06:31:06 channui Exp $";

*****
/**      TPC FILE FOR ALL USERS      ***/
*****

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/time.h>
#include "rte_slave.h"
#include "user_tpc.h"

/* This MUST match the corresponding one in client's inout.h file! */
#define TRIGGER "021"
#define NOSLEEP
// Increased EXPECT_TIMEOUT from 600000 - oz 10/20/97
#define EXPECT_TIMEOUT 600000
#define KEYWAIT_FUDGE 5000

extern SHM_Slave *shm;
extern TableEntrySlave *shmentry;
extern DriverStatus *status;
extern echo_trace(char *);
extern echo_trace();
extern char *expect_save;

const char *SQL_TPERRNO_MESSAGE = "tperrno";
const char *SQL_RTN_MESSAGE = "rtn:";
const char *SQL_FATAL_MESSAGE = "SQL Fatal Error";
const char *ROLLBACK_MESSAGE = "Item number is not valid";

int WHSEID; /* warehouse number for each users */

*****
/* The "uniform()" function has range of the absolute value of the
   * difference between the min. and the max values upto 2147483647. */
*****
/*-----*/
/* NURand */
/*-----*/
/* A: 255 for C_LAST, 1023 for C_ID, 8191 for OL_I_ID */
/* x: 0 for C_LAST, 1 for C_ID and OL_I_ID */
/* y: 999 for C_LAST, 3000 for C_ID, 100000 for OL_I_ID */
/*-----*/
long
NURand(int A, int x, int y, long cval)
{
    return (((long) uniform((long) 0, (long) A) | (long) uniform((long) x, (long) y)) + cval) % (y - x + 1) + x;
}
/*-----*/
/* getname */

```

```

/*-----*/
/* generates a random number from 0 to 999 inclusive */
/* a random name is generated by associating a random */
/* string with each digit of the generated number */
/* three strings are concatenated to generate lastname */
/*-----*/
char *
getname()
{
    char *last_name_parts[] =
    {
        "BAR",
        "OUGHT",
        "ABLE",
        "PRI",
        "PRES",
        "ESE",
        "ANTI",
        "CALLY",
        "ATION",
        "EING"
    };
    static char lastname[128];
    int random_num;

    #if 1
        random_num = NURand(255, 0, 999, shmglobal->lastc);
    #else
        random_num = NURand(255, 0, 999, LASTC);
    #endif
    strcpy(lastname, last_name_parts[random_num / 100]);
    random_num %= 100;
    strcat(lastname, last_name_parts[random_num / 10]);
    random_num %= 10;
    strcat(lastname, last_name_parts[random_num]);
    return (lastname);
}

typedef struct gen_tran_s {
    int invalid;
    void *data;
    long len;
    long keywait;
    long type;
    char *menu;
    char *request;
} gen_tran_t;

int generic_transaction( gen_tran_t *data ) {
    char buffer[2048];
    static int consecutive_errs = 0;
    int rc;
    set_typing_delay(0);
    iprint(IPRINT_TRACE, "> generic_transaction sleep (%d)\n", data->type);
    #ifndef NOSLEEP
        if (shmglobal->test_state == 0)
            transaction_sleep_do();
    #endif

    #ifndef EXPECT_TIMEOUT
        int timeout = EXPECT_TIMEOUT;
    #else
        int timeout = 0;
    #endif

    // Start the transaction (MENU)
    iprint(IPRINT_TRACE, "> generic_transaction start (%d)\n", data->type);
    transaction_start(data->type, data->len, data->data);

    iprint(IPRINT_TRACE, "> transmit data->menu\n");
    transmit(data->menu);
    echo_trace ("Waiting for Menu (DELIVERY)");
    if (expect(TRIGGER, timeout) == ERROR) {
        iprint (IPRINT_ERROR, "Slave %d: Failed to receive %s screen\n",
                shmentry->num, data->menu);
        return (ERROR);
    }
    #ifndef NOSLEEP
        usleep(shmglobal->emulex_menu[data->type]*1000000.0+0.9);
    #endif

    // Send our request (KEYING)
    transaction_mark(WHERE_NOW);
    echo_trace ("Keying");

    #ifndef NOSLEEP
        usleep(data->keywait*1000000+KEYWAIT_FUDGE); // Keying delay
    #endif

    // Wait for response (RESPONSE)
    transaction_mark(WHERE_NOW);

    iprint(IPRINT_TRACE, "> transmit data->request\n");
    transmit(data->request);

    echo_trace ("Wait for Response");
    if (expect(TRIGGER, timeout) == ERROR) {
        iprint (IPRINT_ERROR, "Slave %d: Failed to receive %s response\n",
                shmentry->num, data->menu);
        return (ERROR);
    }
}

```

```

}
#endif NOSLEEP
usleep(shmglobal->emulex_response[data->type]*1000000.0+0.9);
#endif

// Look for errors and set our think time (THINK)
transaction_mark(WHERE_NOW);
if (expect_after_match ("ERROR: ")) {
    data->invalid = 1;
    iprint (IPRINT_ERROR, "Slave %d: %s found %s\n",
           shmentry->num, data->menu, "ERROR.");
    // Very dangerous, keep going rather than exiting...
    return RTE_ERROR;
    // Check for consecutive errors and if there are more than
    // 4 of them exit - allow for transient errors to make
    // tuning and testing easier -oz
    // In either case the transaction is marked as invalid and
    // will be reported as an error by the analyze program.
    // if (consecutive_errs++ > 4)
    //     return RTE_ERROR;
} else {
    consecutive_errs = 0;
}
echo_trace ("Thinking");
transaction_sleep_set(neg_exp_4(shmglobal->think[data->type])*1000.0);
iprint(IPRINT_TRACE, "< generic_transaction finish\n");
return (RTE_OK);
}

/*****
***      Delivery Transaction      ***
*****/
int
Delivery()
{
    static struct delivery_struct delivery, delivery_new;
    int rc;
    char *ptr;
    char buffer[256];
    gen_tran_t tran;

    tran.invalid = 0;
    tran.data = &delivery;
    tran.len = sizeof(delivery);
    tran.keywait = 2;
    tran.type = DELIVERY;
    tran.menu = "4";
    tran.request = buffer;

    // Set up all data for new transactions
    delivery_new.carrier = uniform(1, 10); // carrier # 1 to 10

    // Now create the actual request
    ptr = buffer;
    ptr += sprintf(ptr, "%d\n", delivery_new.carrier);

    // Go do the transaction
    rc = generic_transaction(&tran);
    delivery = delivery_new;
    delivery.invalid = tran.invalid;

    return (rc);
}

/*****
***      New Order Transaction      ***
*****/
int NewOrder() {
    static struct neword_struct neword, neword_new;
    int i, rc, whses, low_whse=1;
    char buffer[2048];
    char *ptr;
    const char *ptr2;
    gen_tran_t tran;

    tran.invalid = 0;
    tran.data = &neword;
    tran.len = sizeof(neword);
    tran.keywait = 18;
    tran.type = NEWORDER;
    tran.menu = "1";
    tran.request = buffer;

    neword_new.rollback=0;

    /**** SECTION TO DETERMINE ROLLBACK TRANSACTION FOR 1% OF NEW ORDERS ****/
    neword_new.did = uniform(1, 10); // district number
    neword_new.cid = NURand(1023, 1, 3000, CUSTC); // customer # 1 to 3000
    neword_new.nloop = uniform(5, 15); // number of items to order (5-15)
    neword_new.olremote=0; // find total number of remote order-lines

    whses = shmglobal->max_warehouses;

    for (i = 0; i < neword_new.nloop; i++) {
        // Warehouse Number
        neword_new.item[i].olswid = WHSEID;
        if (whses > 1 && (uniform(0.0, 100.0) < 1.0)) {
            /* for 1% of items (if * uniform()=0) */
            /* Generate a uniform whse number that's different from WHSEID */
            neword_new.item[i].olswid =
                (long) uniform((long) low_whse, (long) whses-1);
            if (neword_new.item[i].olswid >= WHSEID)
                neword_new.item[i].olswid++;
            neword_new.olremote++; // find total number of remote order-lines
        }
        // Item number 1-100000
        neword_new.item[i].oliid = NURand(8191, 1, 100000, ITEMC);
        // Quantity 1-10
        neword_new.item[i].olquantity = uniform(1, 10);
    }
    // We occasionally force a transaction to have invalid data to force a
    // rollback
    if (uniform(1, 5000) <= 50)
        neword_new.item[neword_new.nloop-1].oliid = 999999;

    neword_new.oremote = (neword_new.olremote > 0);

    // Now create the actual request
    ptr = buffer;
    ptr += sprintf(ptr, "%d\t%d", neword_new.did, neword_new.cid);
    for (i = 0; i < neword_new.nloop; i++) {
        ptr += sprintf(ptr, "%d\t%d\t%d",
                      neword_new.item[i].olswid,
                      neword_new.item[i].oliid,
                      neword_new.item[i].olquantity);
    }
    ptr += sprintf(ptr, "\n");

    // Go do the transaction
    rc = generic_transaction(&tran);
    neword = neword_new;
    neword.invalid = tran.invalid;

    // Check for a rollback
    if (expect_after_match (ROLLBACK_MESSAGE)) {
        neword.rollback=1;
        echo_trace ("Found rollback!\n");
    }

    // Grab the orderID from the
    if (!(ptr2 = expect_after_match ("033[6;15H]"))) {
        echo_trace ("Didn't find order-id for neworder");
        iprint (IPRINT_ERROR, "Neworder didn't have Order-ID");
        neword.oid = -1;
    } else {
        neword.oid = atoi(ptr2+8);
    }

    // This is really not useful since we aren't going to be sending individual
    // keystrokes anymore
    if (shmentry->flags & TES_FLAG_KEYSTROKE_TIME) {
        log_data(RTE_ITEM_KEYSTROKE_TIMES,
                keystroke_length*sizeof(int), keystroke_times);
    }

    return (rc);
}

/*****
***      Order Status Transaction      ***
*****/
int OrderStatus() {
    static struct ordstat_struct ordstat, ordstat_new;
    char buffer[2048];
    int rc;
    char *ptr;
    gen_tran_t tran;

    tran.invalid = 0;
    tran.data = &ordstat;
    tran.len = sizeof(ordstat);
    tran.keywait = 2;
    tran.type = ORDSTAT;
    tran.menu = "3";
    tran.request = buffer;

    // Set up all data for new transactions
    ordstat_new.did = uniform(1, 10); // district number 1 to 10 */
    if (uniform(1, 100) <= 60) /* for 60% of transactions */
        char *tmp = getname();
        strcpy(ordstat_new.clast, tmp); // by customer last name */
        if (ordstat_new.clast[0] < 'A' || ordstat_new.clast[0] > 'Z') {
            iprint (IPRINT_ERROR,
                   "ASSERTION: OrderStatus getname() returns invalid name! %s\n",
                   ordstat_new.clast);
            return RTE_ERROR;
        }
        ordstat_new.byname = 1;
        ordstat_new.cid = 0;
    } else {
        ordstat_new.cid = NURand(1023, 1, 3000, CUSTC); // cust. # 1 to 3000 */
        ordstat_new.byname = 0;
        ordstat_new.clast[0] = (char) NULL;
    }

    // Now create the actual request
    ptr = buffer;
}

```

```

ptr += sprintf(ptr, "%d\t", ordstat_new.did);
if (ordstat_new.byname) {
    ptr += sprintf(ptr, "t%s\n", ordstat_new.clast);
} else {
    ptr += sprintf(ptr, "%d\n", ordstat_new.cid);
}

// Go do the transaction
rc = generic_transaction(&tran);
ordstat = ordstat_new;
ordstat.invalid = tran.invalid;

return (rc);
}

*****
***      Payment Transaction      ***
*****

int
Payment()
{
    static struct payment_struct payment, payment_new;
    int    dollars, cents, rc, whses, low_whse = 1;
    char   buffer[2048];
    char   *ptr;
    gen_tran_t   tran;

    tran.invalid = 0;
    tran.data = &payment;
    tran.len = sizeof(payment);
    tran.keywait = 3;
    tran.type = PAYMENT;
    tran.menu = "2";
    tran.request = buffer;

    payment_new.did = uniform(1, 10); /* district number 1 to 10 */
    if (uniform(1, 100) <= 60) /* for 60% of transactions */
        strcpy(payment_new.clast, getname(), 17); // by customer last name
    if (payment_new.clast[0] < 'A' || payment_new.clast[0] > 'Z') {
        fprintf (IPRINT_ERROR,
                "ASSERTION: payment_new getname() returns invalid name! '%s'\n",
                payment_new.clast);
        return RTE_ERROR;
    }
    payment_new.byname = 1;
    payment_new.cid = 0;
} else {
    payment_new.cid = NURand(1023, 1, 3000, CUSTC); /* cust. # 1 to 3000 */
    payment_new.byname = 0;
    payment_new.clast[0] = (char) NULL;
}

whses = shmglobal->max_warehouses;

if (whses < 2 || uniform(1, 100) <= 85) /* for 85 % of transactions */
    payment_new.cwid = WHSEID;
    payment_new.cdid = payment_new.did;
    payment_new.remote = 0;
} else {
    /* for 15 % of transactions */
    payment_new.cwid = (long) uniform((long)low_whse, (long) whses-1);
    if (payment_new.cwid >= WHSEID)
        payment_new.cwid++;

    payment_new.remote = 1;
    payment_new.cdid = uniform(1, 10); /* district 1 to 10 */
}

dollars = uniform(1, 5000); /* dollar amt = 1 to 5000 */
if (dollars == 5000)
    cents = 0;
else
    cents = uniform(0, 99);

payment_new.amount = ((double) dollars) + ((double) cents) / 100.0;

// Now create the actual request
ptr = buffer;
ptr += sprintf(ptr, "%d\t", payment_new.did);
if (payment_new.byname) {
    ptr += sprintf(ptr, "t%s\t", payment_new.clast);
} else {
    ptr += sprintf(ptr, "%d\t\t", payment_new.cid);
}
ptr += sprintf(ptr, "%d\t%d\t", payment_new.cwid, payment_new.cdid);
ptr += sprintf(ptr, "%d.%02.2d\n", dollars, cents);

// Go do the transaction
rc = generic_transaction(&tran);
payment = payment_new;
payment.invalid = tran.invalid;

return (rc);
}

*****
***      Stock Level Transaction      ***
*****

int
StockLevel()

```

```

{
    static struct stocklev_struct stocklevel, stocklevel_new;
    char   buffer[2048];
    int    rc;
    char   *ptr;
    gen_tran_t   tran;

    tran.invalid = 0;
    tran.data = &stocklevel;
    tran.len = sizeof(stocklevel);
    tran.keywait = 2;
    tran.type = STOCKLEV;
    tran.menu = "5";
    tran.request = buffer;

    stocklevel_new.invalid = 0;
    stocklevel_new.threshold = uniform(10, 20); /* uniform no. between 10 and * 20 */

    // Now create the actual request
    ptr = buffer;
    ptr += sprintf(ptr, "%d\n", stocklevel_new.threshold);

    // Go do the transaction
    rc = generic_transaction(&tran);
    stocklevel = stocklevel_new;
    stocklevel.invalid = tran.invalid;

    return (rc);
}

*****
*** MAIN() ***
*****

int
user_transaction()
{
    char   logout[32];
    double ntask;
    int    resp;
    static int task = 0;

    if (shmentry->flags & TES_FLAG_KEYSTROKE_TIME) {
        int rc;
        /* Wait for specified period of time */
        sleep (shmglobal->keystroke_sleep);
        /* Quit after one transaction */
        shm->lock(shmentry->pid);
        shmentry->flags |= TES_FLAG_DIE;
        shm->unlock(shmentry->pid);
        rc = NewOrder();
        fprintf (IPRINT_INFO, "Slave %d: Keystroke timing setting die flag\n", shmentry->num);
        return rc;
    }

    #if 1
    switch (shmglobal->test_state) {
        case 0: // Normal
            break;
        case 1: // pause
            sleep (1);
            return RTE_OK;
        case 2: // warmup
            switch(task++) {
                case 0: return Delivery();
                case 1: return OrderStatus();
                case 2: return Payment();
                case 3: return StockLevel();
                case 4: task = 0; return NewOrder();
            }
    }

    /******
    *** CHOOSE ONE OF THE TRANSACTIONS ***
    *****/
    ntask = (double) uniform(0.0, 100.0);
    if (ntask <= shmglobal->chances[DELIVERY]) {
        return Delivery();
    }
    ntask -= shmglobal->chances[DELIVERY];
    if (ntask <= shmglobal->chances[ORDSTAT]) {
        return OrderStatus();
    }
    ntask -= shmglobal->chances[ORDSTAT];
    if (ntask <= shmglobal->chances[PAYMENT]) {
        return Payment();
    }
    ntask -= shmglobal->chances[PAYMENT];
    if (ntask <= shmglobal->chances[STOCKLEV]) {
        return StockLevel();
    }
    return NewOrder();
}

#else
// this code should be shared between all of the users on a slave
// int the best case it should be shared between all of the slaves,
// but that would be too costly.
// for now it is done on a per user basis. If this thing is ever
// modified to be threaded then it will probably go to the per-process
// basis. Although with shared memory, it would be possible to go to
// per-slave. Actually, before this code is put into use it must be

```

```

// fixed up to share across processes. Right now it will take, on average,
// 22 minutes for one user to just key in the 100 entries.

// use a card deck with no replacement to fulfill the requirements
{
    int deck[100], count=-1, i, size=1, tmp;
    // lock deck
    if (count < 0) {
        // deck is empty fill it up
        count = 0;
        for (i = 0; i < 43 * size; i++) {
            deck[count++] = Payment;
        }
        for (i = 0; i < 4 * size; i++) {
            deck[count++] = StockLevel;
        }
        for (i = 0; i < 4 * size; i++) {
            deck[count++] = OrderStatus;
        }
        for (i = 0; i < 4 * size; i++) {
            deck[count++] = Delivery;
        }
        for (; count < 100 * size; i++) {
            deck[count++] = NewOrder;
        }
        // randomize the deck
        for (i = 0; i < 100 * size; i++) {
            int tmp;
            int pick = uniform(i+1, 100);
            tmp = deck[i];
            deck[i] = deck[pick];
            deck[pick] = tmp;
        }
        tmp = deck[count--];
        // unlock deck
        switch(tmp) {
            case Delivery: return Delivery();
            case OrderStatus: return OrderStatus();
            case Payment: return Payment();
            case StockLevel: return StockLevel();
            case NewOrder: return NewOrder();
        }
    }
}
#endif

#if 0
if (resp != RTE_OK) { /* logoff if response is not correct */
    strcpy(logout, "9\n"); /* menu option 9 */
    transmit(logout);
    resp = expect("tpcc_cstux_inf:");
    return (ERROR);
} else
    return (RTE_OK);
#endif

/* end of main */

int user_parameter_change(void) {
    #if 0
    int i;
    fprintf(IPRINT_TRACE, "Slave %d: total_users = %d\n", shmentry->num);
    fprintf(IPRINT_TRACE, "Slave %d: chances = ", shmentry->num);
    for (i = 0; i < MAX_TRAN_TYPE; i++)
        fprintf(IPRINT_TRACE, "%6.2f", shmglobal->chances[i]);
    fprintf(IPRINT_TRACE, "\nSlave %d: think = ", shmentry->num);
    for (i = 0; i < MAX_TRAN_TYPE; i++)
        fprintf(IPRINT_TRACE, "%6.2f", shmglobal->think[i]);
    fprintf(IPRINT_TRACE, "\n");
    #endif
    return RTE_OK;
}

int user_login(char *user, char *password, void *data) {
    UserLocal *localdata = (UserLocal *)data;
    int rc;
    int timeout_value = shmglobal->login_timeout;
    char buffer[32];
    set_typing_delay(0);

    rc = expect (TRIGGER, timeout_value);
    if (rc == RTE_ERROR) {
        fprintf (IPRINT_ERROR, "Slave %d: didn't find Warehouse prompt\n", shmentry->num);
    }
    sprintf(buffer, "%d%d\n", localdata->Warehouse, localdata->District);
    transmit(buffer);
    fprintf (IPRINT_TRACE, "Slave %d: Warehouse=%d, District=%d, pid=%d\n", shmentry->num,
        localdata->Warehouse, localdata->District, getpid());

    rc = expect (TRIGGER, timeout_value);
    if (rc != RTE_OK) {
        fprintf (IPRINT_ERROR, "Slave %d: Failed logging in\n", shmentry->num);
        return RTE_ERROR;
    }
    return RTE_OK;
}

int user_init () {
    extern int expect_save_active;
    WHSEID = shmlocal->Warehouse;
}

```

```

status->max_transmit = shmglobal->keystroke_packet_size;
expect_save_active = 1;
return RTE_OK;
}

int user_logout () {
    transmit("9");
    fprintf (IPRINT_TRACE, "Slave %d: Warehouse=%d, District=%d\n", shmentry->num,
        shmlocal->Warehouse, shmlocal->District);
    return RTE_OK;
}

int user_cleanup () {
    transaction_sleep_do();
    transaction_start(0, 0, NULL); // Just something to clear out the buffer...
    return RTE_OK;
}

int user_spawn_ok () {
    int rc, hb;
    hb = ((UserGlobal *) (shm->global_data))->host_busy;
    rc = hb ? RTE_ERROR : RTE_OK;
    return rc;
}

```

## D.4 user tpcc.h

```

/*****
 * user_tpcc.h          Audit: 05/30/96 */
 *****/

/* $Id: user_tpcc.h,v 1.1 1999/02/22 06:31:06 channui Exp $ */

#ifndef USER_TPCC_H
#define USER_TPCC_H
/*****
 *** run-time constant for customer last name from 0 to 255,    ***/
 *** run-time constant for customer id from 0 to 1023,        ***/
 *** run-time constant for item id from 0 to 8191.            ***/
 *****/
#define LASTC 117 /*
 * Change for 3.1 */
#define LASTC 193
#define CUSTC 319
#define ITEM 3849

/*****
 *** response type
 *****/
#define OK 1 /*
 * #define ERROR -1 */

/*****
 *** transaction type
 *****/
#define NEWORDER 1
#define PAYMENT 2
#define ORDSTAT 3
#define DELIVERY 4
#define STOCKLEV 5

/*****
 *** transaction structures
 *****/
struct neword_struct {
    char    invalid; /* transaction completed successfully */
    long    did;
    long    cid;
    long    oid; /* Order-ID returned from client */
    long    nloop; /* number of order line, avg = 15 */
    char    oremote; /* 1 for remote order, 10% */
    long    olremote; /* number of remote order line, 1% */
    char    rollback; /* actually saw rollback text on screen */
    struct items_struct {
        long    olswid;
        long    olid;
        long    olquantity;
    } item[15];
};

struct payment_struct {
    char    invalid; /* transaction completed successfully */
    long    did;
    long    cid;
    long    cwid;
    long    cdid;
    char    clast[17];
    double    amount;
    char    byname; /* 1 for by last name, 0 for by id */
    char    remote; /* 1 for remote warehouse, 0 otherwise */
};

```

```

struct ordstat_struct {
char      invalid;      /* transaction completed successfully */
long did;
long cid;
char clast[17];
char byname;           /* 1 for by last name, 0 for by id */
};

struct delivery_struct {
char      invalid;      /* transaction completed successfully */
char carrier;
};

struct stocklev_struct {
char      invalid;      /* transaction completed successfully */
long threshold;
};

struct generic_struct {
char      invalid;      /* transaction completed successfully */
};

union transaction_info {
char      invalid;
struct generic_struct generic;
struct neword_struct neword;
struct payment_struct payment;
struct ordstat_struct ordstat;
struct delivery_struct delivery;
struct stocklev_struct stocklev;
};

struct UserGlobal {
int total_users;
int max_warehouses;
int keystroke_sleep;
int login_timeout;
int keystroke_packet_size;
int lastc;
int test_state;
int host_busy;
double chances[MAX_TRAN_TYPE];
double think[MAX_TRAN_TYPE];
double emulex_response[MAX_TRAN_TYPE];
double emulex_menu [MAX_TRAN_TYPE];
};

struct UserLocal {
int Warehouse;
int District;
};

struct user_data_header {
};

extern UserGlobal *shmglobal;
extern UserLocal *shmlocal;

#endif

```

# Appendix E: Third Party Quote

Oct-20-00 13:49	NetCruiser Tech. Inc	908 222 9992				
	<b>SALES ORDER</b>	P. 01 Page 1 of 1				
NetCruiser Technologies, Inc. 3005 Hadley Road Unit #3 South Plainfield, NJ 07080 (908) 222-8811		ORDER NUMBER: 0000124 ORDER DATE: 10/11/00				
		SALESPERSON: 0002 CUSTOMER NO: 90-A0USER0				
<b>SOLD TO:</b> Yahoo Order (USA)/DBA: CoreReady		<b>SHIP TO:</b> Doug Yakesch Austin, TX 78758				
CONFIRM TO:						
CUSTOMER P.O.	SHIP VIA UPS-G	F.O.B.	TERMS VISA Card			
ITEM NUMBER	UNIT	ORDERED	SHIPPED	BACK ORDER	PRICE	AMOUNT
CT1008DP	EACH	1	0	0	18.00	18.00
8 Port Desktop Hub/Plastic		WISE: 000				
(Fax: 512-836-1852)						
Price Valid Through December 31, 2000 <i>P.D.</i>						
<i>Gr 10</i> <i>11</i>						
						Net Order: 18.00 Less Discount: 0.00 Freight: 0.00 Sales Tax: 0.00 <b>Order Total: 18.00</b>
OCT 20 '00 13:26		908 222 9992		PAGE. 01		

