
IBM System x3950

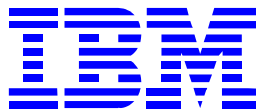
Using

DB2 9 Enterprise Edition

and

SUSE Linux Enterprise Server 10 for Intel EM64T

TPC BenchmarkTM C
Full Disclosure Report



First Edition

December 15, 2006

Special Notices

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

IBM System p

IBM System x

AIX

IBM

DB2

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

TPC Benchmark, TPC-C, and tpmC are trademarks of the Transaction Processing Performance Council

Microsoft Windows 2000 server and COM+ are registered trademarks of Microsoft Corporation

Linux is a registered trademark of Linus Torvalds

SUSE is a registered trademark of Novell, Inc.

First Edition: December 15, 2006

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.

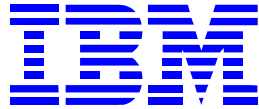
Request for additional copies of this document should be sent to the following address:

TPC Benchmark Administrator
IBM Commercial Performance
Mail Stop 9571
11501 Burnet Road
Austin, TX 78758
FAX Number (512) 838-1852

© Copyright International Business Machines Corporation, 2006. 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 System x3950
DB2® 9**

TPC-C Rev. 5.7

**Report Date:
December 15, 2006**

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date	
\$1,754,100 USD	331,087	\$5.30 USD	Feb 14, 2007	
Database Processors/Cores/Threads	Database Manager	Operating System	Other Software	No. Users
4/8/16 3.5 GHz Intel Xeon 7150N	DB2 9	SLES 10	Microsoft Visual C++ Microsoft COM+	280,320

Gigabit Switch

<p><u>16 Clients</u> IBM® System x3500 2x Intel 5160 DC 3.0GHz 2GB Memory 36GB Internal Drive 2 Gigabit Ethernet</p>	<p><u>IBM® System x3950</u> 4xIntel Xeon 7150N (3.5GHz) 128GB Memory 1x DS4000 Single Port 4GB FC 1x36GB Internal SAS Drives 6 x DS4000 Dual Port 4GB FC 2 x Intel Pro/1000 GT Ethernet</p>	<p><u>Storage</u> 6 x IBM® DS4800 Storage Servers 63x IBM® EXP810 DiskEnclosures 1008 x 36.4GB 15K RPM Drives 1 x DS3400 DC Storage Controller 7 x 73.4GB 15K RPM SAS Drives</p>
--	---	--

System Components	Each of the 16 Clients		Server	
	Quantity	Description	Quantity	Description
Processors/Cores/Threads	2/4/4	3.0GHz 2MB L2 Xeon 5160 Dual Core	4/8/16	3.5 GHz Intel Xeon 7150N
Memory	4	512 MB	32	4 GB
Disk Controllers	1	SAS	2 6 1 6 1	Integrated SAS 4Gb dual-port FC Adapters 4Gb single-port FC Adapters IBM DS4800 Controllers for Data IBM DS3400 Controller for Log
Disk Drives	1	36 GB	1008 1 7	36.4GB 15K RPM FC 36GB 10K RPM SAS 73.4GB 15K RPM SAS
Total Storage		576 GB		36,820 GB
Terminals	1	System Console	1	System Console

IBM Corporation	IBM System x3950 c/s			TPC-C Revision 5.7			
	DB2 9			Report Date: Dec. 15, 2006			
Description	Part Number	Third Party Brand	Pricing	Unit Price	Quantity	Extended Price	3-Yr. Maint. Price
Server Hardware							
IBM System x3950 (2 x Intel Xeon Processor 7150N 3.5GHz 667MHz, 2 x 1 MB L2 Cache and 16MB L3 Cache)	8878-5RU	IBM	1	25,199	1	25,199	
Intel Xeon Processor 7150N 3.5GHz/2x1MB L2, 16MB L3	43V4553	IBM	1	5,999	2	11,998	
x3950 E (0 Processors, 0 Memory, 2 Memory Cards)	8879-1RU	IBM	1	6,999	1	6,999	
Scalability Cable 2.3.M	13M7414	IBM	1	299	2	598	
8GB (2x4GB) PC2-3200 CL3 2RX4 ECC DDR2 SDRAM RDIMM	30R5145	IBM	1	5,569	16	89,104	
Active Memory 4-Slot Memory Expansion Card	41Y5000	IBM	1	499	2	998	
IBM 36.4GB 10K 2.5 inch SAS Drive	40K1051	IBM	1	279	1	279	
Intel Pro/1000 GT Single Port Server Adapter	39Y6105	IBM	1	139	2	278	
Display	49387NU	IBM	1	149	1	149	
IBM Preferred Pro USB Keyboard	40K9584	IBM	1	29	1	29	
IBM 3-Button Optical Mouse - Black - USB	40K9201	IBM	1	19	1	19	
ServicePac for 3-Year 24x7x4 Support (x3950)	96P2688	IBM	1	3,390	1		3,390
ServicePac for 3-Year 24x7x4 Support (x3950 E)	10N3059	IBM	1	1,695	1		1,695
ServicePac for 3-Year 24x7x4 Support (Display)	30L9183	IBM	1	90	1		90
Subtotal						135,650	5,175
Server Storage							
IBM DS4000 FC 4Gbps - PCI-X Dual Port HBA	39M5895	IBM	1	2,485	6	14,910	
IBM TotalStorage DS4800 Midrange Disk Subsystem	1815-82H	IBM	1	65,245	6	391,470	
IBM Short Wave SFP Module (4 Pack)	22R0483	IBM	1	550	54	29,700	
IBM 1m LC-LC Fibre Channel Cable	39M5696	IBM	1	79	144	11,376	
IBM 5m LC-LC Fibre Channel Cable	39M5697	IBM	1	129	12	1,548	
IBM TotalStorage DS4000 EXP810 Storage Exp. Unit	1812-81H	IBM	1	6,000	72	432,000	
36.4GB 15K 2Gbps FC Drive	06P5772	IBM	1	1,115	1008	1,123,920	
IBM System Storage DS3400 Dual Controller	1726-42X	IBM	1	10,224	1	10,224	
IBM Short Wave SFP Module	19K1271	IBM	1	499	1	499	
73.4GB 15K SAS Hot-Swap HDD	40K1043	IBM	1	399	7	2,793	
IBM DS4000 FC 4Gbps - PCI-X Single-Port HBA	39M5894	IBM	1	1,485	1	1,485	
1M Fibre Optic Cable	39M5696	IBM	1	79	1	79	
IBM S2 42U Standard Rack	93074RX	IBM	1	1,489	6	8,934	
ServicePac for 3-Year 24x7x4 Support (EXP810)	41L2768	IBM	1	760	72		54,720
ServicePac for 3-Year 24x7x4 Support (DS4800)	96P2062	IBM	1	1,087	6		6,522
ServicePac for 3-Year 24x7x4 Support (DS3400)	41L2768	IBM	1	760	1		760
ServicePac for 3-Year 24x7x4 Support (Rack)	41L2760	IBM	1	300	6		1,800
Subtotal						2,028,938	63,802
Server Software							
DB2 ESE 9		IBM	2	262.65	400	105,060	
Extended Systems - SW License and Maintenance 12 Months							
SW Maintenance Renewal - 1 Year		IBM	2	12.51	800		10,008
SUSE Linux Enterprise Server 10 for x86, AMD64, and Intel EM64T with Priority Support for 3 Years	4584142100	SUSE	3	3,748	1	3,748	
Subtotal						108,808	10,008
Client Hardware							
Linksys ProConnect KVM Switch - 2 ports (2 spares)	430446		6	40	3	120	
IBM System x3500 (Dual-Core Intel Xeon 5160 3.00GHz, 2x512MB RAM)	7977-92U	IBM	1	2,639	16	42,224	
3.00GHz/1333MHz/2x2MB L2 Xeon Processor 5160 Upgrade	40K1236	IBM	1	1,279	16	20,464	
1GB (2x512MB) PC2-5300 CL5 ECC DDR2 FBD 667MHz	39M5782	IBM	1	259	16	4,144	
36.4GB 15K SAS Hot-Swap Drive	40K1042	IBM	1	249	16	3,984	
ServicePac for 3-Year 24x7x4 Support (x3550)	96P2250	IBM	1	586	16		9,376
Subtotal						70,936	9,376
Client Software							
Microsoft Windows Server 2000 with COM+	C11-00821	Microsoft	4a	738	16	11,808	
Microsoft Visual C++ Professional 6.0	254-00170	Microsoft	4	109	1	109	
Microsoft Problem Resolution Services		Microsoft	4	245	1		245
Subtotal						11,917	245
Network Components							
D-LINK DGS-1024D 24-Port 10/100/1000 Switch (2 spares)	DGS1024D		5	235	3	705	
Ethernet Cable (2 spares)	CC5E-B14B		7	3	20	60	
Subtotal						765	
Total						2,357,014	88,606
Large Purchase Discount (See Note 1.)	29.89%		1				691,520
Pricing: 1 - IBM - 1-888-SHOP-IBM, ext. 5821; 2 - IBM; 3 - Novell; 4 - Microsoft, 4a = 8% discount; 5 - compuplus.com; 6 - CDW.com; 7 - newegg.com				Three-Year Cost of Ownership USD:		\$1,754,100	
Note 1: Discount based on IBM Direct guidance applies to all line items where Pricing=1. Pricing is for this system or one of similar size.				tpmC:		331,087	
Note 2: Pricing for DB2 9 is based on Value Units (VUs) as shown in the price quote in Appendix D.				\$ USD/tpmC:		\$5.30	
Note 3: Fifteen x335 were substituted for the priced clients.							
Audited by Francois Raab, InfoSizing, Inc. (www.sizing.com)							
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 stated prices are not available according to these terms, please inform the TPC at pricing @ tpc.org.							

Numerical Quantities Summary for the IBM System x3950

MQTH, computed Maximum Qualified Throughput: 331,087 tpmC

<u>Response Times (in seconds)</u>	<u>90th %</u>	<u>Average</u>	<u>Maximum</u>
New Order	2.11	1.17	5.35
Payment	2.20	1.19	6.10
Order-Status	1.95	1.15	4.40
Delivery (interactive)	1.30	0.64	4.40
Delivery (deferred)	0.30	0.19	0.81
Stock-Level	2.09	1.30	5.00
Menu	1.42	0.65	5.21

Response time delay added for emulated components was 0.1 seconds

<u>Transaction Mix, in percent of total transactions</u>	<u>Percent</u>
New Order	44.90%
Payment	43.03%
Order-Status	4.01%
Delivery	4.01%
Stock-Level	4.02%

<u>Keying/Think Times (in seconds)</u>	<u>Min.</u>	<u>Average</u>	<u>Max.</u>
New Order	18.00/0.01	18.01/12.02	18.08/120.20
Payment	3.00/0.01	3.01/12.02	3.05/120.20
Order-Status	2.00/0.01	2.01/10.01	2.05/100.10
Delivery	2.00/0.01	2.01/5.02	2.03/50.20
Stock-Level	2.00/0.01	2.01/5.02	2.05/50.20

Test Duration

Ramp-up Time	2 hours 19 minutes
Measurement interval	2 hours 00 minutes
Transactions during measurement interval (all types)	88469266
Ramp-down time	17 minutes

Checkpoints

Number of checkpoints	N/A
Checkpoint interval	N/A

Table of Content

Preface.....	8
0 General Items	9
0.1. Application Code Disclosure	9
0.2. Benchmark Sponsor	9
0.3. Parameter Settings.....	9
0.4. Configuration Diagrams.....	9
1 Clause 1: Logical Data Base Design Related Items	11
1.1. Table Definitions.....	11
1.2. Database Organization	11
1.3. Insert and/or Delete Operations.....	11
1.4. Horizontal or Vertical Partitioning.....	11
2 Clause 2: Transaction & Terminal Profiles Related Items	12
2.1. Verification for the Random Number Generator.....	12
2.2. Input/Output Screens.....	12
2.3. Priced Terminal Features	12
2.4. Presentation Managers	12
2.5. Home and Remote Order-lines.....	12
2.6. New-Order Rollback Transactions.....	12
2.7. Number of Items per Order	13
2.8. Home and Remote Payment Transactions.....	13
2.9. Non-Primary Key Transactions.....	13
2.10. Skipped Delivery Transactions	13
2.11. Mix of Transaction Types	14
2.12. Queuing Mechanism of Delivery	14
3 Clause 3: Transaction and System Properties	15
3.1. Atomicity Requirements	15
3.2. Consistency Requirements	15
3.3. Isolation Requirements.....	16
3.4. Durability Requirements	16
4 Clause 4: Scaling and Data Base Population Related Items.....	18
4.1. Cardinality of Tables.....	18
4.2. Distribution of Tables and Logs.....	18
4.3. Data Base Model Implemented.....	19
4.4. Partitions/Replications Mapping	19
4.5. 60-Day Space Calculations	22
5 Clause 5: Performance Metrics and Response Time Related Items	23
5.1. Response Times	23
5.2. Keying and Think Times.....	23
5.3. Response Time Frequency Distribution	24
5.4. Performance Curve for Response Time versus Throughput	26
5.5. Think Time Frequency Distribution.....	27
5.6. Throughput versus Elapsed Time.....	27
5.7. Steady State Determination.....	28
5.8. Work Performed During Steady State.....	28
5.9. Measurement Interval.....	29
6 Clause 6: SUT, Driver, and Communication Definition Related Items	30
6.1. RTE Availability	30
6.2. Functionality and Performance of Emulated Components.....	30
6.3. Network Bandwidth	30
6.4. Operator Intervention	30
7 Clause 7: Pricing Related Items	31
7.1. Hardware and Programs Used.....	31
7.2. Three Year Cost of System Configuration.....	31
7.3. Availability Dates	31
7.4. Statement of tpmC and Price/Performance	31

8	Clause 9: Audit Related Items.....	32
9	Appendix A: Client Server Code.....	35
9.1.	Client/Terminal Handler Code.....	35
9.2.	Transaction Code.....	46
10	Appendix B: Tunable Parameters.....	78
10.1.	Database Parameters.....	78
10.2.	Transaction Monitor Parameters.....	79
10.3.	Linux Parameters.....	81
11	Appendix C: Database Setup Code.....	87
11.1.	Database Creation Scripts.....	87
11.2.	Data Generation.....	215
12	Appendix D: Pricing.....	227
	Abstract	

This report documents the full disclosure information required by the TPC Benchmark™ C Standard Specification Revision 5.7 dated April, 2006, for measurements on the IBM System x3950. The software used on the IBM System x3950 includes SUSE Linux Enterprise Server 10 for Intel EM64T operating system and DB2 9 data server. Microsoft COM+ is used as the transaction manager.

IBM System x3950

Company Name	System Name	Data Base Software	Operating System Software
IBM Corporation	IBM System x3950	DB2 9	SUSE Linux Enterprise Server 10 for Intel EM64T

Total System Cost	TPC-C Throughput	Price/Performance
<ul style="list-style-type: none"> • Hardware • Software • 3 Years Maintenance 	Sustained maximum throughput of system running TPC-C expressed in transactions per minute	Total system cost/tpmC
\$1,754,100 USD	331,087	\$5.30 USD

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 5.7 in April 2006.

This is the full disclosure report for benchmark testing of the IBM System x3950 and DB2 9 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:

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

0 General Items

0.1. 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 application code for the five TPC Benchmark™ C transactions.

0.2. 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.**

0.3. 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.*

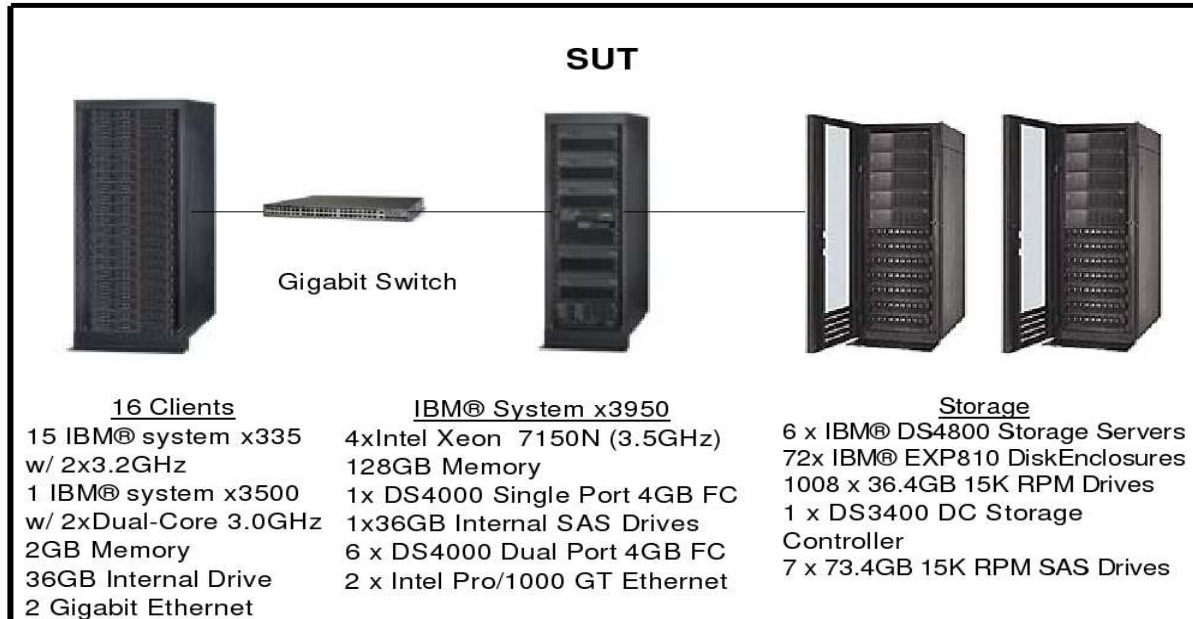
Appendix B contains the system, data base, and application parameters changed from their default values used in these TPC Benchmark™ C tests.

0.4. Configuration Diagrams

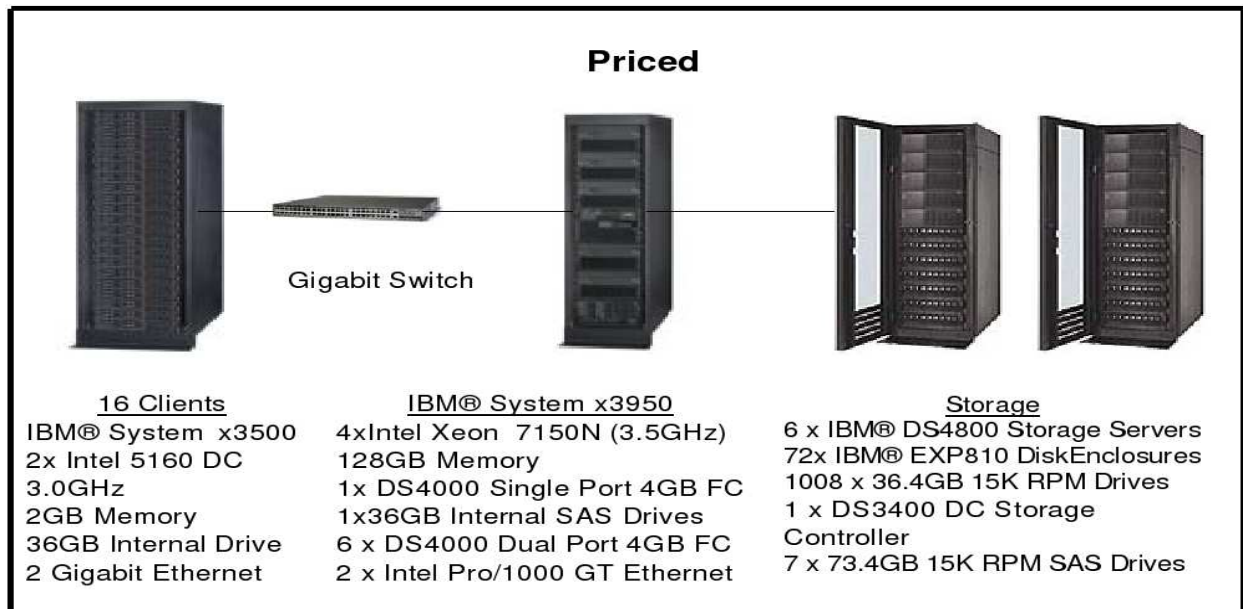
Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:

- *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)*

IBM System x3950 Benchmark Configuration



IBM System x3950 Priced Configuration



The measured configuration consisted of 1 x3500 client with two 3.0GHz Intel 5160 Dual Core processors and 15 x335 clients each with two 3.2GHz Intel Xeon DP processors. However, 16 x3500 clients were priced. Based on the response times and throughput values collected from each of the clients during the run and submitted to the auditor, the auditor determined that this upgrade does not have a material effect on the reported performance. All other components, including the Fibre Channel disk subsystem, adapters, HBAs, and the system under test, were priced as measured. For the priced configuration, see the Executive Summary.

1 Clause 1: Logical Data Base Design Related Items

1.1. 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.

1.2. Database Organization

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

Physical space was allocated to DB2 on the server disks according to the details provided in Appendix C.

1.3. 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 DB2 and priced as static space.

The insert and delete functions were verified by the auditor. In addition, the auditor verified that the primary key for each database table could be updated outside the range of its initial partition.

1.4. 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.

WAREHOUSE, DISTRICT, STOCK, CUSTOMER, HISTORY, ORDERS, ORDERLINE, and NEWORDER were horizontally partitioned into multiple tables.

Each table partition contains data associated with a range of 584 warehouses.

For each partitioned table, a view was created over all table partitions to provide full transparency of data manipulation.

No tables were replicated.

2 Clause 2: Transaction & Terminal Profiles Related Items

2.1. Verification for the Random Number Generator

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

The `srandom()`, `getpid()` and `gettimeofday()` functions are used to produce unique random seeds for each driver. The drivers use these seeds to seed the `srand()`, `srandom()` 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 seeds for each user were captured and verified by the auditor to be unique. In addition, the contents of the database were systematically searched and randomly sampled by the auditor for patterns that would indicate the random number generator had affected any kind of a discernible pattern; none were found.

2.2. Input/Output Screens

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

The screen layouts are now presented in HTML 1.0 web pages. Clauses 2.4.3, 2.5.3, 2.6.3, 2.7.3, and 2.8.3 of the TPC-C specifications were used as guidelines for html character placement.

2.3. 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 eServer x226 systems, are commercially available and support all of the requirements in Clause 2.2.2.4.

2.4. Presentation Managers

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

The 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.

2.5. Home and Remote Order-lines

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

Table 2-1 shows the percentage of home and remote transactions that occurred during the measurement period for the New-Order transactions.

2.6. 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 2-1 shows the percentage of New-Order transactions that were rolled back due to an illegal item being entered.

2.7. Number of Items per Order

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

Table 2-1 show the average number of items ordered per New-Order transaction.

2.8. Home and Remote Payment Transactions

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

Table 2-1 shows the percentage of home and remote transactions that occurred during the measurement period for the Payment transactions.

2.9. 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 2-1 shows the percentage of non-primary key accesses to the data base by the Payment and Order-Status transactions.

2.10. 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 2-1 shows the percentage of Delivery transactions missed due to a shortage of supply of rows in the NEW-ORDER table.

2.11. Mix of Transaction Types

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

Table 2-1 shows the mix percentage for each of the transaction types executed by the SUT.

New Order	IBM System x3950
Percentage of Home order lines	99.00%
Percentage of Remote order lines	1.00%
Percentage of Rolled Back Transactions	1.00%
Average Number of Items per order	10
Payment	
Percentage of Home transactions	85.00%
Percentage of Remote transactions	15.00%
Non-Primary Key Access	
Percentage of Payment using C_LAST	60.00%
Percentage of Order-Status using C_LAST	59.98%
Delivery	
Delivery transactions skipped	0
Transaction Mix	
New-Order	44.90%
Payment	43.03%
Order-Status	4.01%
Delivery	4.01%
Stock-Level	4.02%

Table 2-1: Numerical Quantities for Transaction and Terminal Profiles

2.12. Queuing Mechanism of Delivery

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

The Delivery transaction was submitted to an ISAPI queue that is separate from the COM+ queue that the other transactions used. This queue is serviced by a variable amount of threads that are separate from the worker threads inside the web server. Web server threads are able to complete the on-line part of the Delivery transaction and immediately return successful queuing responses to the drivers. The threads servicing the queue are responsible for completing the deferred part of the transaction asynchronously.

3 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.

3.1. 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.

3.1.1. 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, BALANCE_1, was retrieved from the CUSTOMER table for a random Customer, District and Warehouse combination.
2. The Payment transaction was executed and committed for the Customer, District, and Warehouse combination used in step 1.
3. The balance, BALANCE_2, was retrieved again for the Customer, District, and Warehouse combination used in step 1 and step 2. It was verified that BALANCE_1 was greater than BALANCE_2 by the amount of the Payment transaction.

3.1.2. 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 implemented with a Perl script that allowed the transaction to be rolled back rather than committed.
2. The balance, BALANCE_3, was retrieved from the Customer table for the same Customer, District, and Warehouse combination used in the completed Payment transaction Atomicity test.
3. The Payment transaction was executed for the Customer, District and Warehouse used in step 2. Rather than commit the transaction, the transaction was rolled back.
4. The balance, BALANCE_4 was retrieved again for the Customer, District, and Warehouse combination used in step 2. It was verified that BALANCE_4 was equal to BALANCE_3, demonstrating that there were no remaining effects of the rolled back Payment transaction.

3.2. 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.

The specification defines 12 consistency conditions of which the following four are required to be explicitly demonstrated:

1. The sum of balances (d_ytd) for all Districts within a specific Warehouse is equal to the balance (w_ytd) of that Warehouse.
2. For each District within a Warehouse, the next available Order ID (d_next_o_id) minus one is equal to the most recent Order ID [max(o_id)] for the Order table associated with the preceding District and Warehouse.

Additionally, that same relationship exists for the most recent Order ID [max(o_id)] for the New Order table associated with the same District and Warehouse. Those relationships can be illustrated as follows:

$$d_next_o_id - 1 = \max(o_id) = \max(no_o_id)$$

where (d_w_id = o_w_id = no_w_id) and (d_id = o_d_id = no_d_id)

3. For each District within a Warehouse, the value of the most recent Order ID [max(no_o_id)] minus the first Order ID [min(no_o_id)] plus one, for the New Order table associated with the District and Warehouse equals the number of rows in that New Order table. That relationship can be illustrated as follows:

$$\max(no_o_id) - \min(no_o_id) + 1 = \text{number of rows in New Order for the Warehouse/District}$$

4. For each District within a Warehouse, the sum of Order Line counts [sum(o_ol_cnt)] for the Order table associated with the District equals the number of rows in the Order Line table associated with the same District. That relationship can be illustrated as follows:

$$\text{sum}(o_ol_cnt) = \text{number of rows in the Order Line table for the Warehouse/District}$$

An RTE driven run was executed against a freshly loaded database. After the run the 4 consistency conditions defined above were tested using a script to issue queries to the database. All queries showed that the database was still in a consistent state.

3.3. 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.

The benchmark specification defines nine tests to demonstrate the property of transaction isolation. The tests, described in Clauses 3.4.2.1 – 3.4.2.9 were all successfully executed using a series of scripts. Case A was observed during the execution of Isolation Tests 7-9.

3.4. 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

- *Permanent irrecoverable failure of any single durable medium containing TPC-C database tables or recovery log data (this test includes failure of all or part of memory)*
- *Instantaneous interruption (system crash/system hang) in processing that requires system reboot to recover*
- *Failure of all or part of memory (loss of contents)*

Failure of Log Disk, Log Fast Write Cache, Instantaneous Interruption and Memory Failure:

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

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 full load test was started and allowed to run for over 10 minutes.
3. One of the disks containing the transaction log was removed. Since the log was implemented as a RAID-5 array, DB2 continued to process the transactions successfully.
4. The test continued for at least another 5 minutes.
5. A storage controller holding one copy of the mirrored write cache for the log was removed from the storage subsystem. The contents of the write cache mirrors became out-of-sync.

6. The system was subsequently powered off, which removed power from all system components, including memory.
7. The storage controller from step 5 was reinserted into the storage subsystem. The controller detected the cache out-of-sync condition and synchronized with the write cache mirror in the other controller.
8. The disk from step 3 was replaced.
9. The system was powered back on and DB2 was allowed to recover.
10. Step 1 was performed returning the value for SUM_2. It was verified that SUM_2 was greater than SUM_1 plus the completed New_Order transactions recorded by the RTE. The additional transactions found in the database were attributed to in-flight activity at the time of the failure.
11. Consistency condition 3 was verified.

Failure of Durable Medium Containing TPC-C Database Tables:

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

1. The contents of the database were backed up in full.
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 with 12.5% of the full load.
4. A disk containing the TPC-C tables was removed, causing DB2 to report numerous errors.
5. The system was subsequently shutdown.
6. The disk was reinserted.
7. The system was powered back on.
8. The full database was restored from the backup copy in step 1.
9. DB2 was restarted and the transactions in the log were applied to the database.
10. Step 2 was performed returning SUM_2. It was verified that SUM_2 was equal to SUM_1 plus the number of completed New_Order transactions recorded by the RTE.
11. Consistency condition 3 was verified.

4 Clause 4: Scaling and Data Base Population Related Items

4.1. 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 4-1 portrays the TPC Benchmark™ C defined tables and the number of rows for each table as they were built initially.

Table Name	Number of Rows
Warehouse	28,032
District	280,320
Customer	840,960,000
History	840,960,000
Order	840,960,000
New Order	252,288,000
Order Line	8,409,748,445
Stock	2,803,200,000
Item	100,000

Table 4-1: Initial Cardinality of Tables

4.2. Distribution of Tables and Logs

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

There is one Logical Disk (LD) for the logs.

There is one storage adapter for the log LD.

The log LD is configured as a RAID5 (6+p) disk array, with 7 physical disks. Each physical disk has a capacity of 73.4 GB. The total Raid 5 capacity available is 440 GB.

There are 48 Logical Disks (LD) for the tables.

There are 6 storage adapters for the tables. Each adapter is assigned 8 LDs.

Each LD is configured as a RAID0 disk array, with 21 physical disks.

There are a total of 1008 data disks and each physical disk has a capacity of 36.4 GB.

Each LD is partitioned identically. Each LD contains 14 partitions, 12 of which are used for DB2 Containers. Partitions are laid out on an LD as follows:

Partition#	Blocks	Container Usage
1	7968	Warehouse
2	8001	District
3	7969	Item
4	Extended partition	
5	23270058	Stock

6	16868164	Customer
7	1606440	Customer Index
8	2409662	History
9	1606434	Orders
10	1606406	Order Index
11	26507192	Order Line
12	803178	New OrdersA
13	803164	New OrdersB
14	659990299	Not used for data tables

Tablespaces are laid out to use LDs as follows:

Warehouse	48 tablespaces each using 1 LD
District	48 tablespaces each using 1 LD
Item	One tablespace using 48 LDs
Stock	48 tablespaces each using 1 LD
Customer	48 tablespaces each using 1 LD
Customer Index	48 tablespaces each using 1 LD
History	48 tablespaces each using 1 LD
Orders	48 tablespaces each using 1 LD
Order Index	48 tablespaces each using 1 LD
Order Line	48 tablespaces each using 1 LD
New OrdersA	48 tablespaces each using 1 LD
New OrdersB	48 tablespaces each using 1 LD

4.3. 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 DB2 9. DB2 is a relational DBMS. DB2 remote stored procedures and embedded SQL statements were used. The DB2 stored procedures were invoked via SQL CALL statements. Both the client application and stored procedures were written in embedded C code.

4.4. Partitions/Replications Mapping

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

The specifics of the distribution of partitioned and non-partitioned tables across the physical media can be found in tables 4-2.

Data Distribution Logical Disks (LDs)			
PARTITION	Storage Adapter	RAW Device	Assigned Tablespace
LD1 – LD8 Partition 1	FC1	raw1 –raw8	ts_wh_01-ts_wh_08
LD9 – LD16 Partition 1	FC2	raw9 –raw16	ts_wh_09-ts_wh_16
LD17 – LD24 Partition 1	FC3	raw17 –raw24	ts_wh_17-ts_wh_24
LD25 – LD32 Partition 1	FC4	raw25 –raw32	ts_wh_25-ts_wh_32
LD33 – LD40 Partition 1	FC5	raw33 –raw40	ts_wh_33-ts_wh_40
LD41 – LD48 Partition 1	FC6	raw41 –raw48	ts_wh_41-ts_wh_48
LD1 – LD8 Partition 2	FC1	raw101 –raw108	ts_dis_01-ts_dis_08
LD9 – LD16 Partition 2	FC2	raw109 –raw116	ts_dis_09-ts_dis_16
LD17 – LD24 Partition 2	FC3	raw117 –raw124	ts_dis_17-ts_dis_24

LD25 – LD32 Partition 2	FC4	raw125 –raw132	ts_dis_25-ts_dis_32
LD33 – LD40 Partition 2	FC5	raw133–raw140	ts_dis_33-ts_dis_40
LD41 – LD48 Partition 2	FC6	raw141 –raw148	ts_dis_41_ts_dis_48
LD1 – LD8 Partition 3	FC1	raw201 –raw208	ts_item
LD9 – LD16 Partition 3	FC2	raw209 –raw216	ts_item
LD17 – LD24 Partition 3	FC3	raw217 –raw224	ts_item
LD25 – LD32 Partition 3	FC4	raw225 –raw232	ts_item
LD33 – LD40 Partition 3	FC5	raw233–raw240	ts_item
LD41 – LD48 Partition 3	FC6	raw241 –raw248	ts_item
LD1 – LD8 Partition 5	FC1	raw301 –raw308	ts_stock_01-ts_stock_08
LD9 – LD16 Partition 5	FC2	raw309 –raw316	ts_stock_09-ts_stock_16
LD17 – LD24 Partition 5	FC3	raw317 –raw324	ts_stock_17-ts_stock_24
LD25 – LD32 Partition 5	FC4	raw325 –raw332	ts_stock_25-ts_stock_32
LD33 – LD40 Partition 5	FC5	raw333–raw340	ts_stock_33-ts_stock_40
LD41 – LD48 Partition 5	FC6	raw341 –raw348	ts_stock_41-ts_stock_48
LD1 – LD8 Partition 6	FC1	raw401 –raw408	ts_customer_01-ts_customer_08
LD9 – LD16 Partition 6	FC2	raw409 –raw416	ts_customer_09-ts_customer_16
LD17 – LD24 Partition 6	FC3	raw417 –raw424	ts_customer_17-ts_customer_24
LD25 – LD32 Partition 6	FC4	raw425 –raw432	ts_customer_25-ts_customer_32
LD33 – LD40 Partition 6	FC5	raw433–raw440	ts_customer_33-ts_customer_40
LD41 – LD48 Partition 6	FC6	raw441 –raw448	ts_customer_41-ts_customer_48
LD1 – LD8 Partition 7	FC1	raw501 –raw508	is_customer_01-is_customer_08
LD9 – LD16 Partition 7	FC2	raw509 –raw516	is_customer_09-is_customer_16
LD17 – LD24 Partition 7	FC3	raw517 –raw524	is_customer_17-is_customer_24
LD25 – LD32 Partition 7	FC4	raw525 –raw532	is_customer_25-is_customer_32
LD33 – LD40 Partition 7	FC5	raw533–raw540	is_customer_33-is_customer_40
LD41 – LD48 Partition 7	FC6	raw541 –raw548	is_customer_41-is_customer_48
LD1 – LD8 Partition 8	FC1	raw601 –raw608	ts_history_01-ts_history_08
LD9 – LD16 Partition 8	FC2	raw609 –raw616	ts_history_09-ts_history_16
LD17 – LD24 Partition 8	FC3	raw617 –raw624	ts_history_17-ts_history_24
LD25 – LD32 Partition 8	FC4	raw625 –raw632	ts_history_25-ts_history_32
LD33 – LD40 Partition 8	FC5	raw633–raw640	ts_history_33-ts_history_40
LD41 – LD48 Partition 8	FC6	raw641 –raw648	ts_history_41-ts_history_48
LD1 – LD8 Partition 9	FC1	raw701 –raw708	ts_order_01-ts_order_08

LD9 – LD16 Partition 9	FC2	raw709 –raw716	ts_order_09-ts_order_16
LD17 – LD24 Partition 9	FC3	raw717 –raw724	ts_order_17-ts_order_24
LD25 – LD32 Partition 9	FC4	raw725 –raw732	ts_order_25-ts_order_32
LD33 – LD40 Partition 9	FC5	raw733–raw740	ts_order_33-ts_order_40
LD41 – LD48 Partition 9	FC6	raw741 –raw748	ts_order_41-ts_order_48
LD1 – LD8 Partition 10	FC1	raw801 –raw808	is_order_01-is_order_08
LD9 – LD16 Partition 10	FC2	raw809 –raw816	is_order_09-is_order_16
LD17 – LD24 Partition 10	FC3	raw817 –raw824	is_order_17-is_order_24
LD25 – LD32 Partition 10	FC4	raw825 –raw832	is_order_25-is_order_32
LD33 – LD40 Partition 10	FC5	raw833–raw840	is_order_33-is_order_40
LD41 – LD48 Partition 10	FC6	raw841 –raw848	is_order_41-is_order_48
LD1 – LD8 Partition 11	FC1	raw901 –raw908	ts_orderline_01-ts_orderline_08
LD9 – LD16 Partition 11	FC2	raw909 –raw916	ts_orderline_09-ts_orderline_16
LD17 – LD24 Partition 11	FC3	raw917 –raw924	ts_orderline_17-ts_orderline_24
LD25 – LD32 Partition 11	FC4	raw925 –raw932	ts_orderline_25-ts_orderline_32
LD33 – LD40 Partition 11	FC5	raw933–raw940	ts_orderline_33-ts_orderline_40
LD41 – LD48 Partition 11	FC6	raw941 –raw948	ts_orderline_41-ts_orderline_48
LD1 – LD8 Partition 12	FC1	raw1001 –raw1008	ts_newordA_01-ts_newordA_08
LD9 – LD16 Partition 12	FC2	raw1009 –raw1016	ts_newordA_09-ts_newordA_16
LD17 – LD24 Partition 12	FC3	raw1017 –raw1024	ts_newordA_17-ts_newordA_24
LD25 – LD32 Partition 12	FC4	raw1025 –raw1032	ts_newordA_25-ts_newordA_32
LD33 – LD40 Partition 12	FC5	raw1033–raw1040	ts_newordA_33-ts_newordA_40
LD41 – LD48 Partition 12	FC6	raw1041 –raw1048	ts_newordA_41-ts_newordA_48
LD1 – LD8 Partition 13	FC1	raw1141 –raw1148	ts_newordB_01-ts_newordB_08
LD9 – LD16 Partition 13	FC2	raw1149 –raw1156	ts_newordB_09-ts_newordB_16
LD17 – LD24 Partition 13	FC3	raw1157 –raw1164	ts_newordB_17-ts_newordB_24
LD25 – LD32 Partition 13	FC4	raw1165 –raw1172	ts_newordB_25-ts_newordB_32
LD33 – LD40 Partition 13	FC5	raw1173 –raw1180	ts_newordB_33-ts_newordB_40
LD41 – LD48 Partition 13	FC6	raw1141 –raw1148	ts_newordB_41-ts_newordB_48

Table 4-2: IBM System x3950 Data Distribution Benchmark Configuration

4.5. 60-Day Space Calculations

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

60-Day Space Computation

All data sizes in MB unless otherwise stated

Warehouses	28,032
Measured TpmC	331,087

Table	Rows	Table	Index	5% Space	Total Space
Warehouse	28,032	30	0	2	32
District	280,320	84	0	4	88
Item	100,000	10	0	1	11
Stock	2,803,200,000	912,720	0	45,636	958,356
Customer	840,960,000	657,216	40,896	34,906	733,018
New-Order	252,288,000	9,888	0	494	10,382
Orders	840,960,000	32,585	23,904	0	56,489
Order-Line	8,409,600,000	563,605	0	0	563,605
History	840,960,000	52,608	0	0	52,608
Additional Overhead		485,058			485,058

Free Space	422,894		
Dynamic Space	648,798		
Static Space	2,210,848		
Daily Growth	122,608		
Daily Spread	238,982		
Data Storage Requirement			
60 Days (MB)	23,906,239		
60 Days (GB)	23,346		
Log Storage Requirement			
8 Hours (GB)	359.20		

		<u>30 Minute log Computations</u>	
		Log Written (KB)	23,540,286
		New-Order Txns	9,932,610
		Log Written per New-Order (KB)	2.37

Disk Sizing

Disk Type	Formatted		SUT		Priced	
	Capacity (GB)	# of Disks	Capacity (GB)	# of Disks	Capacity (GB)	# of Disks
DB FastT 36.4GB	36.40	1,008	36,691	1,008	36,691	1,008
LOG DS3400 RAID5	73.40	6	440	6	440	6
OS SCSI 36GB	36.40	1	36	1	36	1

Total Capacity						37,168
-----------------------	--	--	--	--	--	--------

5 Clause 5: Performance Metrics and Response Time Related Items

5.1. 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 5-1 lists the response times and the ninetieth percentiles for each of the transaction types for the measured system.

5.2. Keying and Think Times

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

Table 5-1 lists the TPC-C keying and think times for the measured system.

Response Times	New Order	Payment	Order Status	Delivery (int./def.)	Stock Level	Menus
90 %	2.11	2.20	1.95	1.30/0.30	2.09	1.42
Average	1.17	1.19	1.15	0.64/0.19	1.30	0.65
Maximum	5.35	6.10	5.00	4.40/0.81	5.00	5.21
Think Times						
Minimum	0.01	0.01	0.01	0.01	0.01	N/A
Average	12.02	12.02	10.01	5.02	5.02	N/A
Maximum	120.20	120.20	100.10	50.20	50.20	N/A
Keying Times						
Minimum	18.00	3.00	2.00	2.00	2.00	N/A
Average	18.01	3.01	2.01	2.01	2.01	N/A
Maximum	18.08	3.05	2.05	2.03	2.05	N/A

Table 5-1: Think and Keying Times

5.3. Response Time Frequency Distribution

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

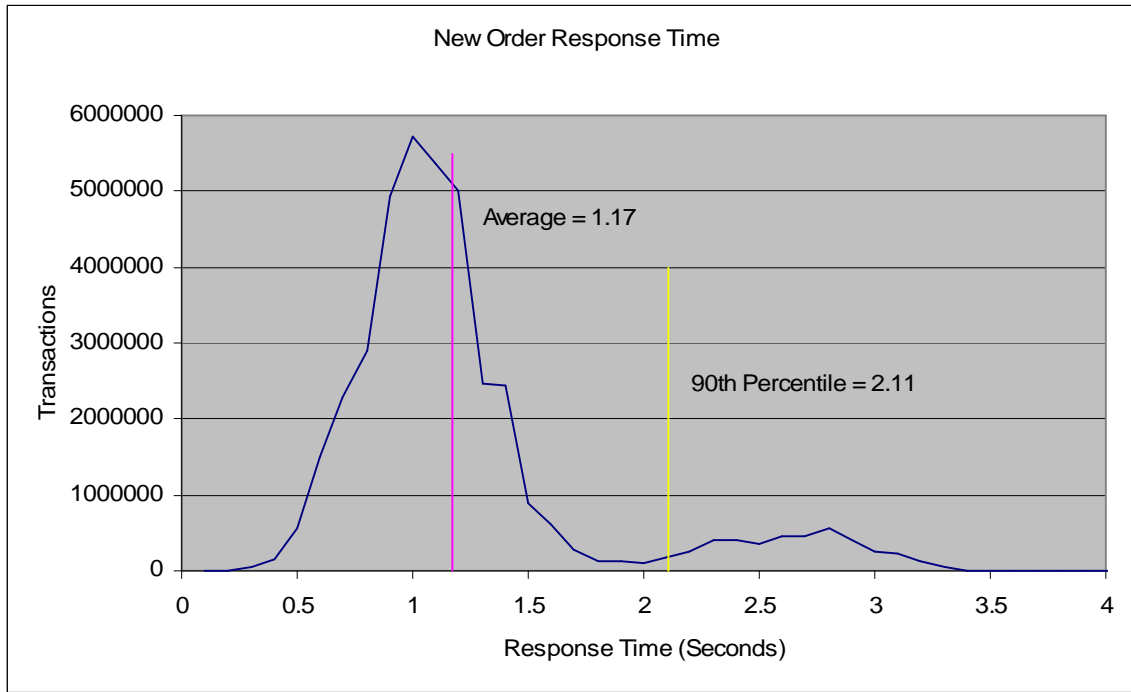


Figure 5-1: New-Order Response Time Distribution

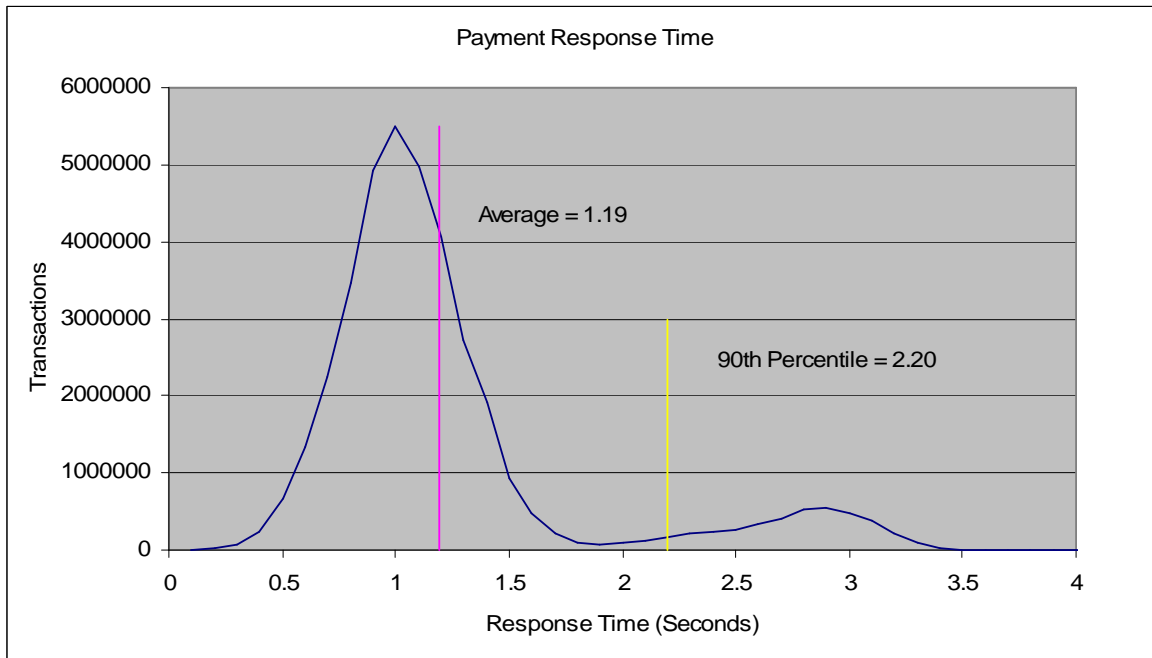


Figure 5-2: Payment Response Time Distribution

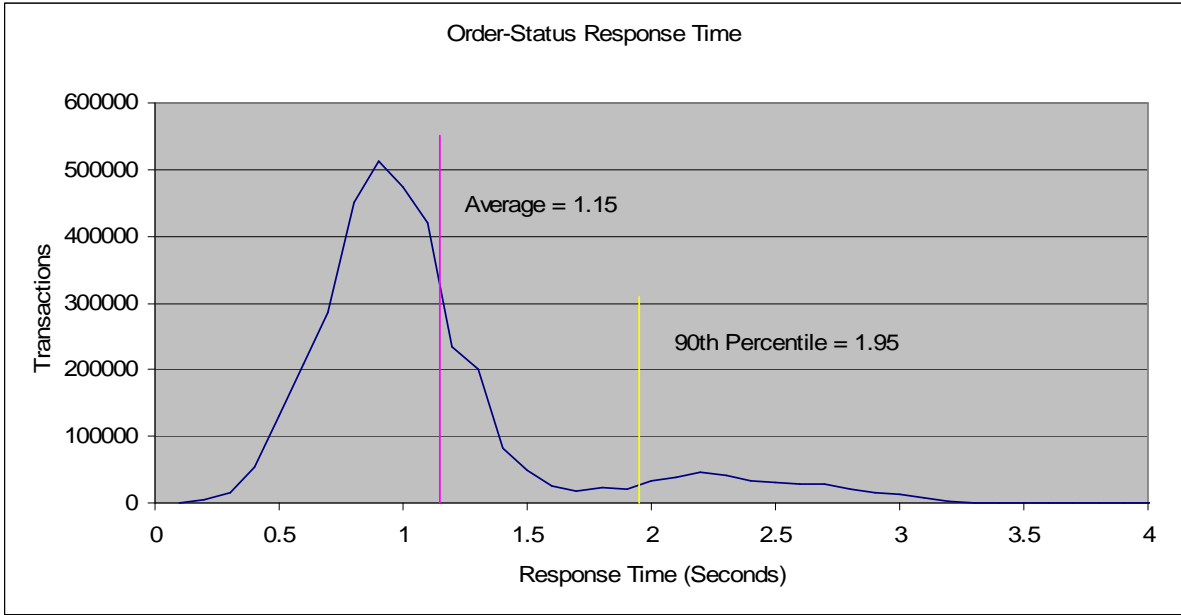


Figure 5-3: Order-Status Response Time Distribution

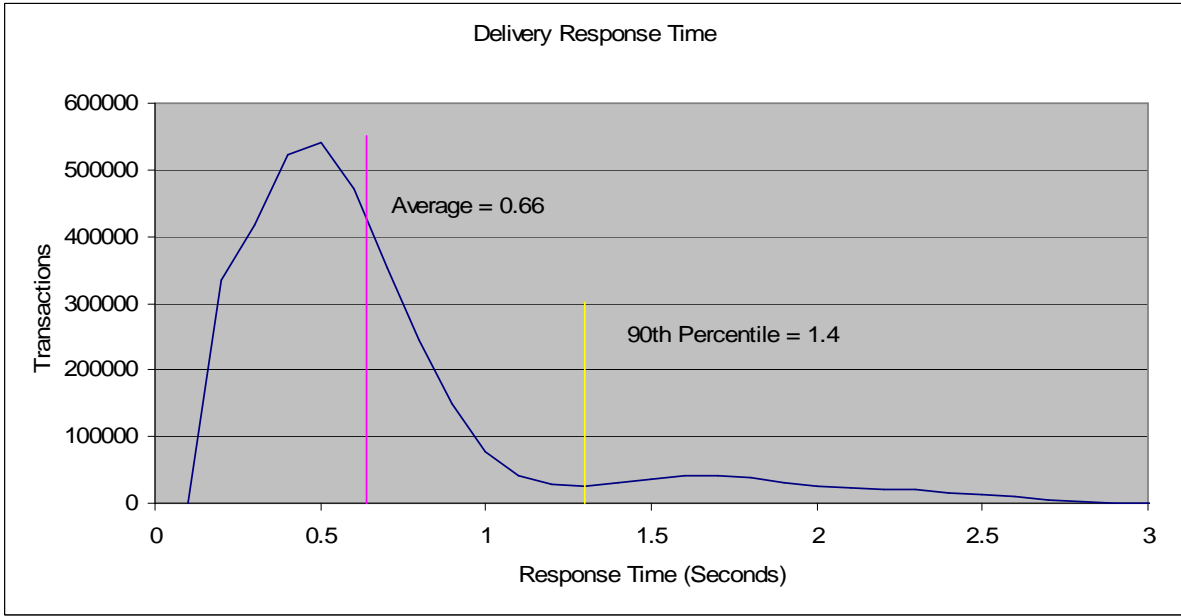


Figure 5-4: Delivery (Interactive) Response Time Distribution

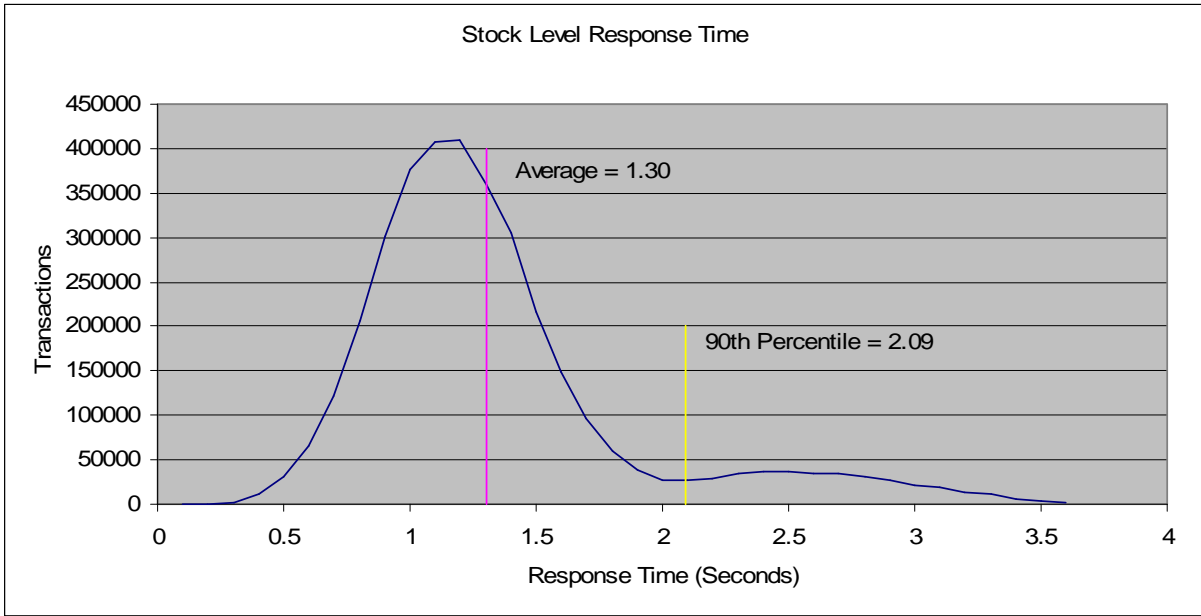


Figure 5-5: Stock Level Response Time Distribution

5.4. Performance Curve for Response Time versus Throughput

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

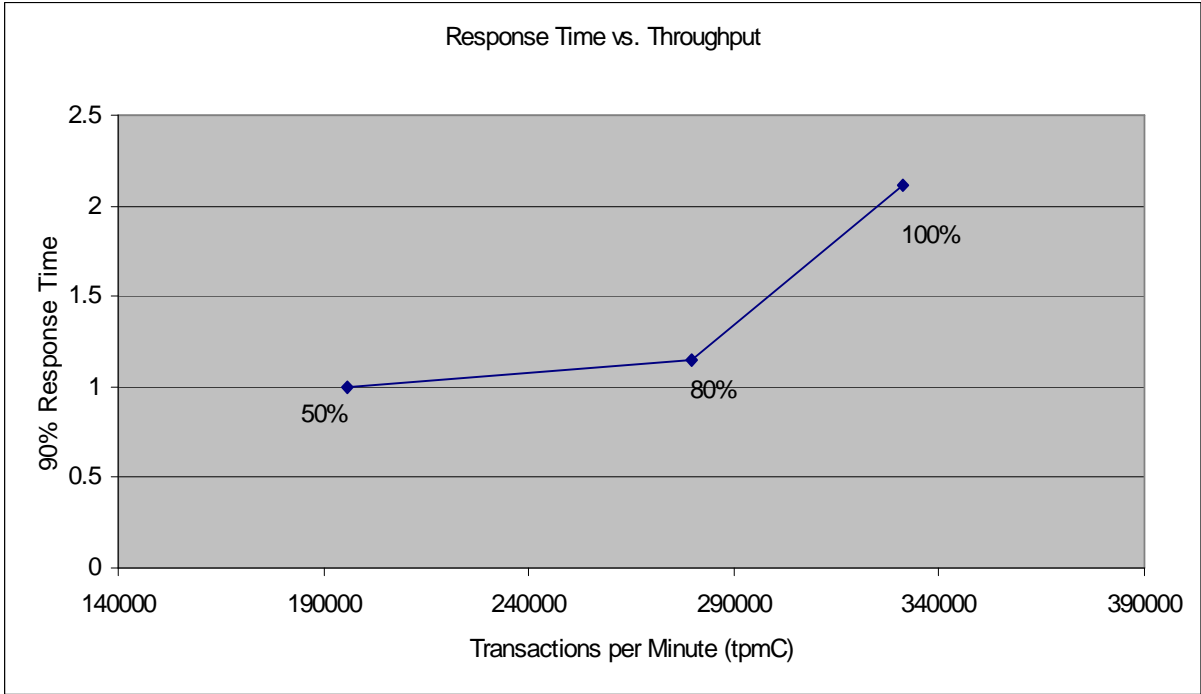


Figure 5-6: New-Order Response Time vs. Throughput

5.5. Think Time Frequency Distribution

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

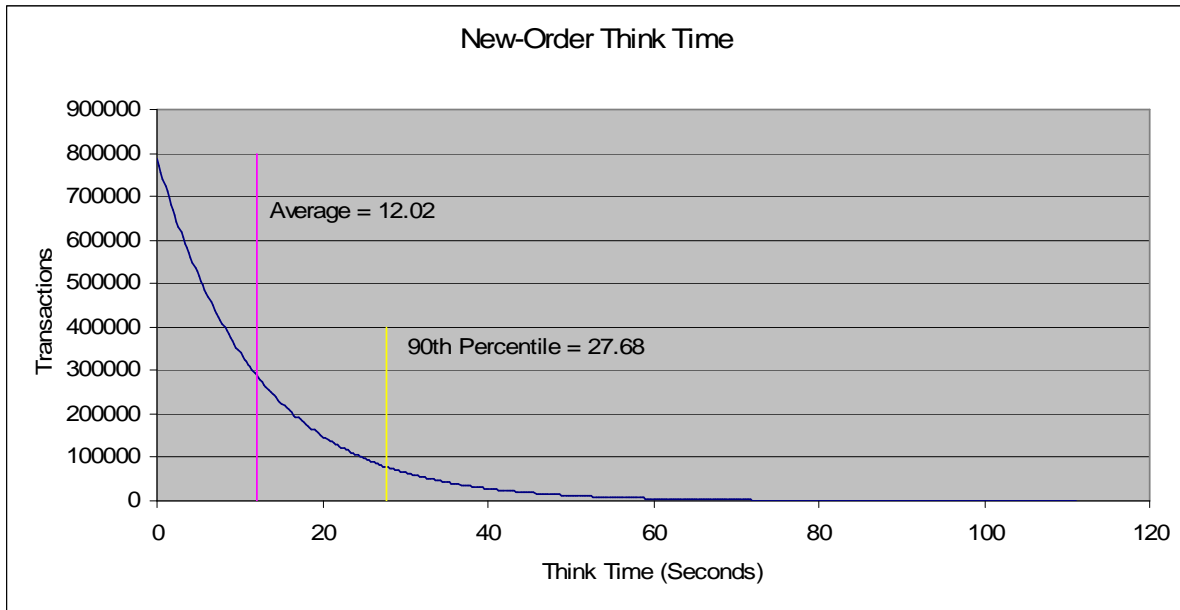


Figure 5-7: New-Order Think Time Distribution

5.6. Throughput versus Elapsed Time

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

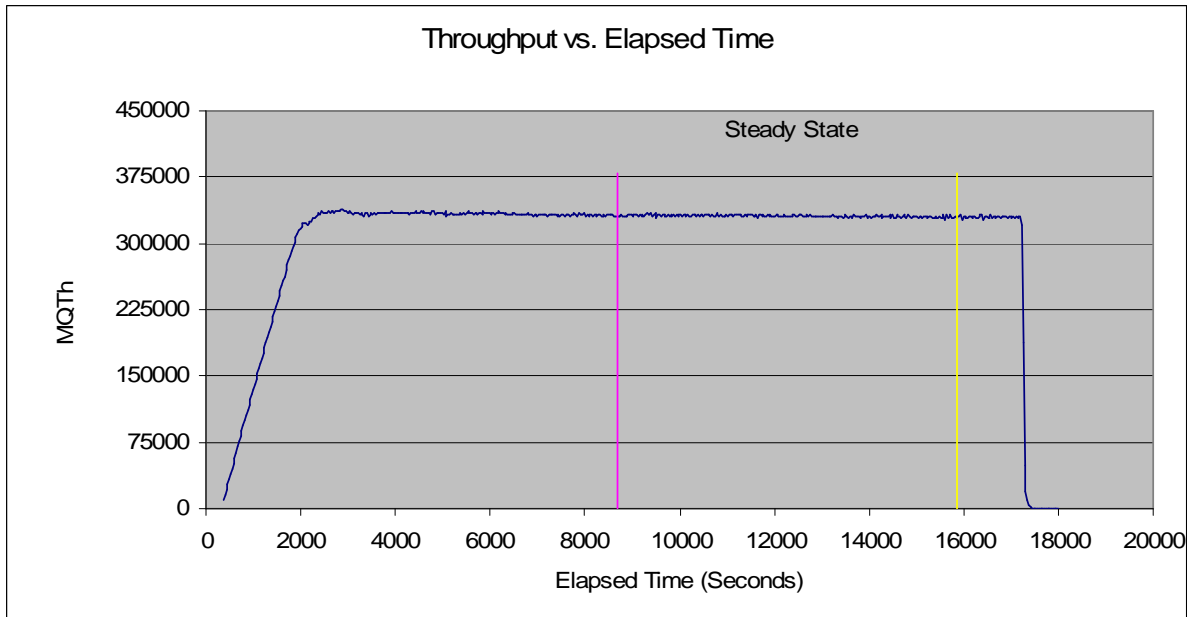


Figure 5-8: New-Order Throughput vs. Elapsed Time

5.7. 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 user ramp-up phase is clearly visible on the graph above. Refer to the Numerical Quantities Summary pages for the rampup time. Figure 5-8 New-Order throughput versus Elapsed Time graph shows that the system maintained a steady state during the measurement interval

5.8. 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.

A 2-hour measurement interval was used to guarantee that all work normally performed during an 8-hour sustained test is included in the reported throughput.

5.8.1. Transaction Flow

Each of the 4 (non-delivery) transactions is serviced by 2 individual programs, Internet Information System 5.1 (IIS) and a Microsoft COM+ 1.0 Queued Component Server, used as the transaction manager (COM+). Both programs are running on the client system:

- The initial HTML 1.0 request is serviced by an ISAPI custom-written handler running on Internet Information System 5.1(IIS). IIS is responsible for handling all HTML requests. The web server communicates to the COM+ server through a Microsoft COM+ API interface.
- COM+ communicates with the server system over Ethernet and handles all database operations, using DB2 embedded SQL calls.

When the COM+ server boots up, it creates a configurable amount of connections to the server (listed in application settings).

COM+ routes the transaction and balances the load according to the options defined in the Component Services GUI for the COM+ server application and settings in the Windows 2000 registry. The configuration file and registry variables are listed in Appendix B.2.

At the beginning, each TPC-C user sends a pair of HTML 1.0 requests submitting its unique warehouse and district to the IIS ISAPI handler. Upon successful validation of user's login, IIS displays an HTML form which encapsulates the TPC-C transaction menu.

The transaction flow is described below:

- The TPC-C user requests the transaction type's HTML form and proceeds to generate (fill in) a GET request with the required files for the transaction.
- IIS accepts the filled in GET request, parses, and validates all values entered by the user.
- It then proceeds to transmit those values to the COM+ server through an transaction type specific COM+ api interface.
- The COM+ Pool Manager receives the request and first decides if there is a connection object in the pool available to service it.
 - If so, the connection is used to send the transaction request to the Server.
 - If no connection is available, the request will enter a COM+ internal queue and will be serviced by the next available connection.
- Once the connection is available to be used, a COM+ pool thread receives the transaction and calls a TPC-C back end DB2 client api to execute all database operations related to the transaction type. (All the transaction information entered on the HTML form is available in a data structure provided by the ISAPI caller).
- The transaction is committed and the DB2 back end client returns control back to the COM pool thread.
- COM pool thread returns control to the ISAPI caller.

(All transaction results are inside the data structure that the ISAPI caller provided to the COM+ api in the parameter list).
- ISAPI caller returns control to the "screen application" by doing a PUT request.

5.8.2. Database Transaction

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

Using embedded SQL calls, the TPC-C back-end program interacts with DB2 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.

DB2 Server proceeds to update the database as follows:

When DB2 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, DB2 Server will make space by flushing some modified pages to disk. Modified pages are also written to disk as part of the “Soft” checkpoint to ensure that no updates remain unflushed for longer than the allowed time. 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.

5.8.3. Checkpoints

DB2 uses a write-ahead-logging protocol to guarantee recovery. This protocol uses “Soft” checkpoint to write least-recently-used database pages to disk independent of transaction commit. However, enough log information to redo/undo the change to a database pages is committed to disk before the database page itself is written. This protocol therefore renders checkpoint unnecessary for DB2. For a more detailed description of the general principles of the write-ahead-logging protocol, see the IBM research paper, “ARIES: A Transaction Recovery Method Supporting Fine Granularity Locking and Partial Rollbacks Using Write-Ahead Logging,” by C. Mohan, Database Technology Institute, IBM Almaden Research Center.

([http:// portal.acm.org/citation.cfm?id=128770&coll=portal&dl=ACM&CFID=10343790&CFTOKEN=42047146](http://portal.acm.org/citation.cfm?id=128770&coll=portal&dl=ACM&CFID=10343790&CFTOKEN=42047146))

5.9. Measurement Interval

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

A 2-hour measurement interval was used. No connections were lost during the run.

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

6.1. 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. 28,032 warehouses were configured. A rampup time of two hours nineteen minutes was specified, along with a run time of two hours, a rampdown wait of two minutes, and a rampdown time of seventeen minutes.

6.2. 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.

No components were emulated.

6.3. Network Bandwidth

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

The network between the clients and the database server was configured as 1000 MegaBits per second Full Duplex.

6.4. 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.

7 Clause 7: Pricing Related Items

7.1. 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 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.

7.2. Three Year Cost of System Configuration

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

The pricing details for this disclosure is contained in the executive summary pages. All 3rd party quotations are included at the end of this report in Appendix D. All prices are based on IBM US list prices.

Discount are based on US list prices and for similar quantities and configurations. A discount of 28.89% has been applied to specified IBM hardware, and services based on the total value and quantities of the components of the configuration.

7.3. 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 components of the SUT will be available on Feb 14, 2007.

7.4. Statement of tpmC and Price/Performance

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

System	tpmC	3-year System Cost	\$/tpmC	Availability Date
IBM System x3950	331,087	\$1,754,100 USD	\$5.30 USD	Feb 14, 2007

Please refer to the price list on the Executive Summary page for details.

8 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:

Sponsors:

Raymond J. Venditti
 IBM Linux Performance
 11501 Burnet Road
 Austin, TX 78758

Berni Schiefer
 IBM DB2 Performance
 8200 Warden Avenue
 Markham, Ontario L6G1C7

Celia Schreiber
 System x Performance
 3039 Cornwallis Road
 Reasearch Triangle Park, NC 27709

December 14, 2006

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

Platform: IBM System x™ 3950 c/s
 Operating system: SLES 10
 Database Manager: DB2 9
 Transaction Manager: Microsoft COM+

The results were:

CPU's Speed	Memory	Disks	NewOrder 90% Response Time	tpmC
Server: IBM System x3950				
4 x Xeon 7150N Dual-Core (3.5 GHz)	128 GB main (16 MB L3 cache per Socket)	1 x 36 GB SAS int. 1008 x 36.4 GB FC 7 x 73.4 GB SAS int.	2.11 Sec.	331,087.79
16 Clients: IBM System x3500 (Specification for each)				
2 x Intel Xeon Dual-Core (3.0 GHz)	2 GB main (2 MB L2 cache per Core)	1 x 36 GB	n/a	n/a

In my opinion, these performance results were produced in compliance with the TPC requirements for the current revision (5.7) of the benchmark.

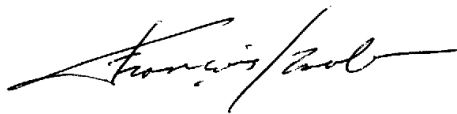
The following verification items were given special attention:

- The transactions were correctly implemented.
- The database records were the proper size.
- The database was properly scaled and populated.
- The ACID properties were met.
- Input data was generated according to the specified percentages.
- The transaction cycle times included the required browser delay, keying and think times.
- The reported response times were correctly measured.
- At least 90% of all delivery transactions met the 80 Second completion time limit.
- All 90% response times were under the specified maximums.
- The measurement interval was representative of steady state conditions.
- The reported measurement interval was 2 hours.
- Write-ahead-logging was active during the measurement interval.
- The 60 day storage requirement was correctly computed.
- The system pricing was verified for major components and maintenance.

Additional Audit Notes:

The measured configuration included fifteen IBM x335 client systems and one IBM x3500 client system. The IBM x335 systems were substituted, one for one, with IBM x3500 systems in the priced configuration. The TPC requirements for substitution of priced clients were verified and met.

Respectfully Yours,

A handwritten signature in black ink, appearing to read "François Raab", written in a cursive style.

François Raab, President

9 Appendix A: Client Server Code

9.1. Client/Terminal Handler Code

Makefile.config

```
#####
#####
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use, duplication or
## disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#####
#
# Makefile.config - NT/Win2000 Makefile Configuration
#
# Make Configuration (MSVC)
MAKE=nmake.exe
#
# Compiler Configuration (MSVC).
# CFLAGS_DEBUG may be set to "-Zi -Od", "-DDEBUGIT" "-Zi -Od -DDEBUGIT" or left
# blank
CC=cl.exe
CFLAGS_OS=-DSQLWINT -MT -DWIN32 -J -Zp8 -DREG_KIT_METHOD
CFLAGS_OUT=/Fo
CFLAGS_DEBUG=
#
# Linker Configuration (MSVC)
LD_EXEC=link.exe
LD_STORP=link.exe
LDFLAGS_EXEC=
LDFLAGS_SHLIB=DLL
LDFLAGS_STORP=$(LDFLAGS_SHLIB) /DEF:rpctpc.def
LDFLAGS_LIB=/LIBPATH:$(TPCC_SQLLIB)/lib /LIBPATH:"C:\Program Files\Microsoft
Visual Studio\VC98\lib" db2api.lib winmm.lib
LDFLAGS_OUT=/OUT:
#
# Library Configuration
AR=lib.exe
ARFLAGS=
ARFLAGS_LIB=
ARFLAGS_OUT=/OUT:
#
# OS Commands
ERASE=del /F
ERASEDIR=rmdir /S
MOVE=MOVE
COPY=COPY
#
# OS File Extensions & Path Separator
OBJEXT=.obj
LIBEXT=.lib
```

```
SHLIBEXT=.dll
BINEXT=.exe
SLASH=\
CMDSEP=&
```

Src.Cli/Makefile

```
#####
#####
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use, duplication or
## disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
# =====
#
# Makefile - Makefile for Src.Cli (RTE/Driver Interface)
#
!include $(TPCC_ROOT)/Makefile.config
#
#####
# Preprocessor Compiler and Linker Flags
#
#####
BND_OPTS = GRANT PUBLIC \
            MESSAGES $*.bnd.msg
PRP_OPTS = BINDFILE \
            ISOLATION RR \
            EXPLAIN ALL \
            MESSAGES $*.prep.msg \
            LEVEL $(TPCC_VERSION) \
            NOLINEMACRO
INCLUDES = -I$(TPCC_SQLLIB)/include -I$(TPCC_ROOT)/include
CFLAGS = $(CFLAGS_OS) $(INCLUDES) $(CFLAGS_DEBUG) \
          $(UOPTS) -D$(DB2EDITION) -D$(DB2VERSION) -D$(TPCC_SPTYPE)
OBJS = $(TPCC_ROOT)/Src.Common/tpccdbg$(OBJEXT) \
        $(TPCC_ROOT)/Src.Common/tpccctx$(OBJEXT) \
        tpcccli$(OBJEXT)
LIBS = tpcccli$(LIBEXT)
#
#####
# User Targets
#
#####
all: connect $(OBJS) plan $(LIBS) disconnect
      $(AR) $(ARFLAGS) $(ARFLAGS_OUT)tpcccli$(LIBEXT) $(OBJS) $(ARFLAGS_LIB)
      @echo "-----"
      @echo "Please copy lval.h, db2tpcc.h, and tpcccli$(LIBEXT) to"
```

```
@echo "a place where they can be #included and linked with the"
@echo "RTE/driver code."
@echo "-----"
```

```
clean:
- $(ERASE) *.msg *.bnd *.plan *$(OBJEXT) *$(LIBEXT) tpcccli.c
#
#####
# Helper Targets
#
#####
connect:
- db2 connect to $(TPCC_DBNAME)
disconnect:
- db2 connect reset
- db2 terminate
plan:
- db2exfmt -d $(TPCC_DBNAME) -e $(TPCC_SCHEMA) -s $(TPCC_SCHEMA) -w -1 -n
TPCCCLI -g -# 0 -o TPCCCLI.exfmt.plan
- db2expln -d $(TPCC_DBNAME) -c $(TPCC_SCHEMA) -p TPCCCLI -s 0 -g -o
TPCCCLI.expln.plan
rebind: connect
      db2 bind tpcccli.bnd $(BND_OPTS) QUERYOPT 7
#
#####
# Build Rules
#
#####
.SUFFIXES:
.SUFFIXES: $(OBJEXT) .c .sqc
tpcccli.c:
@echo "Prepping $*.sqc"
- db2 prep $*.sqc $(PRP_OPTS) ISOLATION RR
@echo "Binding $*.bnd"
db2 bind $*.bnd $(BND_OPTS) QUERYOPT 7
#
#####
# Dependencies
#
#####
# Client Library:
tpcccli$(LIBEXT): $(OBJS)
#
# Source
tpcccli$(OBJEXT): tpcccli.c
#
# Headers
tpcccli.c: $(TPCC_ROOT)/include/db2tpcc.h $(TPCC_ROOT)/include/lval.h
#
Src.Cli/tpcccli.sqc
/*****
** Licensed Materials - Property of IBM
**
```

```

** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****

```

```

/*
 * tpccli.sqc - Client/Server code for TPCC
 */

```

```

#include <stdlib.h>
#include <errno.h>
#include "db2tpcc.h"
#include "tpccapp.h"
#include "tpccdbg.h"

```

```

#include "sqlca.h"
#include "sql.h"

```

```

// -----
// New Order CLIENT
// -----

```

```

static int itemComparison ( const void * a , const void * b )
{
    struct in_items_struct * one = (struct in_items_struct *) a ;
    struct in_items_struct * two = (struct in_items_struct *) b ;

```

```

    if ( one->s_OL_I_ID != two->s_OL_I_ID )
    {
        return ( one->s_OL_I_ID - two->s_OL_I_ID ) ;
    }
    else
    {
        return ( one->s_OL_SUPPLY_W_ID - two->s_OL_SUPPLY_W_ID ) ;
    }
}

```

```

int neword_sql ( struct in_neword_struct * in_neword
                , struct out_neword_struct * neword )

```

```

{
    struct sqlca sqlca ;

    EXEC SQL BEGIN DECLARE SECTION;

```

```

    struct vc_new_in
    {
        short len;
        char data[ 262 ] ;
    } * pHostvarInput ;

```

```

    struct vc_new_out
    {
        short len;
        char data[ 682 ] ;
    } * pHostvarOutput ;

```

```

    EXEC SQL END DECLARE SECTION;

```

```

    int clientRc = TRAN_OK ;

```

```

    int itemIndex = 0 ;

```

```

    in_neword->s_all_local = 1 ;

```

```

    for ( itemIndex = 0 ;
          itemIndex < 15 && in_neword->in_item[ itemIndex ].s_OL_I_ID !=
UNUSED_ITEM_ID ;
          itemIndex++
        )
    {
        if ( in_neword->in_item[ itemIndex ].s_OL_SUPPLY_W_ID != in_neword->s_W_ID )
        {
            in_neword->s_all_local = 0 ;
        }
    }

```

```

    in_neword->s_O_OL_CNT = itemIndex ;

```

```

    qsort( in_neword->in_item, in_neword->s_O_OL_CNT
          , sizeof ( in_neword->in_item[ 0 ] )
          , itemComparison
          ) ;

```

```

    pHostvarInput = (struct vc_new_in *) in_neword ;
    pHostvarInput->len = sizeof(struct in_neword_struct) - SPGENERAL_ADJUST ;

```

```

    pHostvarOutput = (struct vc_new_out *) neword;
    pHostvarOutput->len = sizeof(struct out_neword_struct) - SPGENERAL_ADJUST ;

```

```

#ifdef DEBUGIT
    new_debug(neword, in_neword, "Client before SP call");
#endif /* DEBUGIT */

```

```

#ifdef SWAP_ENDIAN
    for (itemIndex=0; itemIndex<in_neword->s_O_OL_CNT; itemIndex++)
    {
        SWAP_BYTE(in_neword->in_item[ itemIndex ].s_OL_I_ID);
        SWAP_BYTE(in_neword->in_item[ itemIndex ].s_OL_SUPPLY_W_ID);
        SWAP_BYTE(in_neword->in_item[ itemIndex ].s_OL_QUANTITY);
    }
    SWAP_BYTE(in_neword->s_C_ID);
    SWAP_BYTE(in_neword->s_W_ID);
    SWAP_BYTE(in_neword->s_D_ID);
    SWAP_BYTE(in_neword->s_O_OL_CNT);
    SWAP_BYTE(in_neword->s_O_OL_LOCAL);
    SWAP_BYTE(in_neword->s_all_local);
    SWAP_BYTE(in_neword->duplicate_items);
#endif //SWAP_ENDIAN

```

```

    EXEC SQL CALL news ( :pHostvarInput, :pHostvarOutput );

```

```

#ifdef SWAP_ENDIAN
    SWAP_BYTE(in_neword->s_C_ID);
    SWAP_BYTE(in_neword->s_W_ID);
    SWAP_BYTE(in_neword->s_D_ID);
    SWAP_BYTE(in_neword->s_O_OL_CNT);
    SWAP_BYTE(in_neword->s_O_OL_LOCAL);
    SWAP_BYTE(in_neword->duplicate_items);
    for (itemIndex=0; itemIndex<in_neword->s_O_OL_CNT; itemIndex++)
    {
        SWAP_BYTE(in_neword->in_item[ itemIndex ].s_OL_I_ID);
        SWAP_BYTE(in_neword->in_item[ itemIndex ].s_OL_SUPPLY_W_ID);
        SWAP_BYTE(in_neword->in_item[ itemIndex ].s_OL_QUANTITY);
    }

```

```

    SWAP_BYTE(neword->s_W_TAX);
    SWAP_BYTE(neword->s_D_TAX);
    SWAP_BYTE(neword->s_C_DISCOUNT);
    SWAP_BYTE(neword->s_total_amount);
    SWAP_BYTE(neword->s_O_ID);
    SWAP_BYTE(neword->s_O_OL_CNT);
    SWAP_BYTE(neword->s_transtatus);
    SWAP_BYTE(neword->deadlocks);

```

```

    for (itemIndex=0; itemIndex<in_neword->s_O_OL_CNT; itemIndex++)
    {

```

```

        SWAP_BYTE(neword->item[ itemIndex ].s_I_PRICE);
        SWAP_BYTE(neword->item[ itemIndex ].s_OL_AMOUNT);
        SWAP_BYTE(neword->item[ itemIndex ].s_S_QUANTITY);
    }
#endif //SWAP_ENDIAN

```

```

    if ( sqlca.sqlcode == 0 )
    {
        float wtax = neword->s_W_TAX ;
        float dtax = neword->s_D_TAX ;
        float cdisc = neword->s_C_DISCOUNT ;
        float factor = (1.0 - cdisc) * (1.0 + wtax + dtax) ;

```

```

        // Compute order total

```

```

        neword->s_total_amount = 0 ;

```

```

        for ( itemIndex = 0 ;
              itemIndex < in_neword->s_O_OL_CNT ; // from input , not output
              itemIndex++
            )
        {
            if ( neword->item[ itemIndex ].s_I_PRICE > 0 ) // A zero price signifies a bad item
            {
                neword->item[ itemIndex ].s_OL_AMOUNT = neword-
>item[ itemIndex ].s_I_PRICE *
                in_neword->in_item[ itemIndex ].s_OL_QUANTITY ; //
reference input value

```

```

                neword->s_total_amount += neword->item[ itemIndex ].s_OL_AMOUNT ;
            }
        }

```

```

        neword->s_total_amount *= factor;
    }
    else
    {
        sqlerror( NEWORD_SQL, "NEW", __FILE__, __LINE__, &sqlca ;
        neword->s_transtatus = FATAL_SQLERROR ;
        clientRc = FATAL_SQLERROR ;
    }
}

```

```

#ifdef DEBUGIT
    new_debug(neword, in_neword, "Client after SP call");
#endif /* DEBUGIT */

```

```

    if (neword->s_transtatus <= FATAL_SQLERROR)
    {
        new_debug(neword, in_neword, "NEW failed");
        clientRc = FATAL_SQLERROR ;
    }

```

```

    if (neword->s_transtatus == INVALID_ITEM)
    {
        clientRc = INVALID_ITEM ;
    }

```

```

    return ( clientRc ) ;
}

```

```

// -----
// Payment CLIENT
// -----

```

```

int payment_sql ( struct in_payment_struct * in_payment
                 , struct out_payment_struct * payment )

```

```

{
    struct sqlca sqlca ;

```

```

int clientRc = TRAN_OK ;

EXEC SQL BEGIN DECLARE SECTION;

// Inputs

float h_amount ;
sqlint32 in_c_id ;

struct s_data_type { short len ; char data[ 16 ] ; } c_last_input ;

sqlint32 w_id ;
sqlint32 c_w_id ;
short d_id ;
short c_d_id ;

// Outputs

sqlint32 c_id ;

double c_credit_lim ;
float c_discount ;
double c_balance ;

char w_street_1 [ 20 ], w_street_2 [ 20 ] ;
char w_city [ 20 ], w_state [ 2 ], w_zip [ 9 ] ;

char d_street_1 [ 20 ], d_street_2 [ 20 ], d_city [ 20 ] ;
char d_state [ 2 ], d_zip [ 9 ], c_first [ 16 ] ;

char c_last [ 16 ] ;

char c_middle [ 2 ], c_street_1 [ 20 ] ;
char c_street_2 [ 20 ], c_city [ 20 ], c_state [ 2 ] ;
char c_zip [ 9 ], c_phone [ 16 ] ;

char c_credit [ 2 ] ;

char c_since [ 27 ] ;

char c_data [ 200 ] ;
short c_data_indicator = 0 ;

char h_date [ 27 ] ;

struct c_data_prefix_c_last_type { short len ; char data[ 28 ] ; } c_data_prefix_c_last ;
struct c_data_prefix_c_id_type { short len ; char data[ 34 ] ; } c_data_prefix_c_id ;

EXEC SQL END DECLARE SECTION;

// Input redirects

#define h_amount in_payment->s_H_AMOUNT
#define in_c_id in_payment->s_C_ID

#define w_id in_payment->s_W_ID
#define d_id in_payment->s_D_ID

#define c_d_id in_payment->s_C_D_ID
#define c_w_id in_payment->s_C_W_ID

// Output redirects

#define c_credit_lim payment->s_C_CREDIT_LIM
#define c_discount payment->s_C_DISCOUNT
#define c_balance payment->s_C_BALANCE

#define c_id payment->s_C_ID

```

```

#define c_last payment->s_C_LAST

#define c_first payment->s_C_FIRST
#define c_middle payment->s_C_MIDDLE
#define c_street_1 payment->s_C_STREET_1
#define c_street_2 payment->s_C_STREET_2
#define c_city payment->s_C_CITY
#define c_state payment->s_C_STATE
#define c_zip payment->s_C_ZIP
#define c_phone payment->s_C_PHONE
#define c_credit payment->s_C_CREDIT
#define c_since payment->s_C_SINCE_time
#define c_data payment->s_C_DATA

#define w_street_1 payment->s_W_STREET_1
#define w_street_2 payment->s_W_STREET_2
#define w_city payment->s_W_CITY
#define w_state payment->s_W_STATE
#define w_zip payment->s_W_ZIP

#define d_street_1 payment->s_D_STREET_1
#define d_street_2 payment->s_D_STREET_2
#define d_city payment->s_D_CITY
#define d_state payment->s_D_STATE
#define d_zip payment->s_D_ZIP

#define h_date payment->s_H_DATE_time

payment->deadlocks = -1 ;
payment->s_transtatus = TRAN_OK ;

if ( c_w_id == 0 ) { c_w_id = w_id ; }
if ( c_d_id == 0 ) { c_d_id = d_id ; }

#ifdef DEBUGIT
pay_debug(payment, in_payment, "Client before SQL call");
#endif /* DEBUGIT */

// Create c_data_prefix strings and copy some elements from
// in -> out struct outside of retry_tran loop

if ( in_c_id == 0 )
{
c_data_prefix_c_last.len = sprintf( c_data_prefix_c_last.data, " %2.2d %6.6d %2.2d
%6.6d %06.2f", c_d_id, c_w_id, d_id, w_id, h_amount ) ;

// Setup the input c_last varchar
c_last_input.len = strlen( in_payment->s_C_LAST ) ;
memcpy( c_last_input.data, in_payment->s_C_LAST, c_last_input.len ) ;

// Copy to the output structure
memcpy( payment->s_C_LAST, in_payment->s_C_LAST, sizeof( payment-
>s_C_LAST ) ) ;

} else {

// Copy c_id to the output structure
c_id = in_c_id ;

c_data_prefix_c_id.len = sprintf( c_data_prefix_c_id.data, " %5.5d %2.2d %6.6d %2.2d
%6.6d %06.2f", c_id, c_d_id, c_w_id, d_id, w_id, h_amount ) ;

}

retry_tran:

payment->deadlocks ++ ;

```

```

if ( in_c_id == 0 )
{
EXEC SQL BEGIN COMPOUND NOT ATOMIC STATIC

SELECT W_STREET_1, W_STREET_2, W_CITY, W_STATE, W_ZIP
, D_STREET_1, D_STREET_2, D_CITY, D_STATE, D_ZIP
, C_ID, C_FIRST, C_MIDDLE, C_STREET_1, C_STREET_2
, C_CITY, C_STATE, C_ZIP, C_PHONE, C_SINCE, C_CREDIT,
C_CREDIT_LIM
, C_DISCOUNT, C_BALANCE, C_DATA, H_DATE

INTO :w_street_1, :w_street_2, :w_city, :w_state, :w_zip
, :d_street_1, :d_street_2, :d_city, :d_state, :d_zip
, :c_id, :c_first, :c_middle, :c_street_1, :c_street_2, :c_city, :c_state
, :c_zip, :c_phone, :c_since, :c_credit, :c_credit_lim
, :c_discount, :c_balance, :c_data :c_data_indicator, :h_date
FROM TABLE ( PAY_C_LAST( :w_id
, :d_id
, :c_w_id
, :c_d_id
, :c_last_input
, CAST(:h_amount AS DECIMAL(6,2))
, :c_data_prefix_c_last
)

) AS T ( W_STREET_1, W_STREET_2, W_CITY, W_STATE, W_ZIP
, D_STREET_1, D_STREET_2, D_CITY, D_STATE, D_ZIP
, C_ID, C_FIRST, C_MIDDLE, C_STREET_1, C_STREET_2
, C_CITY, C_STATE, C_ZIP, C_PHONE, C_SINCE, C_CREDIT,
C_CREDIT_LIM
, C_DISCOUNT, C_BALANCE, C_DATA, H_DATE
)
;

COMMIT ;

END COMPOUND ;

}
else
{
EXEC SQL BEGIN COMPOUND NOT ATOMIC STATIC

SELECT W_STREET_1, W_STREET_2, W_CITY, W_STATE, W_ZIP
, D_STREET_1, D_STREET_2, D_CITY, D_STATE, D_ZIP
, C_LAST, C_FIRST, C_MIDDLE, C_STREET_1, C_STREET_2
, C_CITY, C_STATE, C_ZIP, C_PHONE, C_SINCE, C_CREDIT,
C_CREDIT_LIM
, C_DISCOUNT, C_BALANCE, C_DATA, H_DATE

INTO :w_street_1, :w_street_2, :w_city, :w_state, :w_zip
, :d_street_1, :d_street_2, :d_city, :d_state, :d_zip
, :c_last, :c_first, :c_middle, :c_street_1, :c_street_2, :c_city, :c_state
, :c_zip, :c_phone, :c_since, :c_credit, :c_credit_lim
, :c_discount, :c_balance, :c_data :c_data_indicator, :h_date
FROM TABLE ( PAY_C_ID( :w_id
, :d_id
, :c_w_id
, :c_d_id
, :in_c_id
, CAST(:h_amount AS DECIMAL(6,2))
, :c_data_prefix_c_id
)

) AS T ( W_STREET_1, W_STREET_2, W_CITY, W_STATE, W_ZIP
, D_STREET_1, D_STREET_2, D_CITY, D_STATE, D_ZIP
, C_LAST, C_FIRST, C_MIDDLE, C_STREET_1, C_STREET_2
, C_CITY, C_STATE, C_ZIP, C_PHONE, C_SINCE, C_CREDIT,
C_CREDIT_LIM
, C_DISCOUNT, C_BALANCE, C_DATA, H_DATE

```

```

);
;
COMMIT;
END COMPOUND;
}
#endif DEBUGIT
pay_debug(payment, in_payment, "Client after SQL call");
#endif /* DEBUGIT */

if ( sqlca.sqlcode != 0 )
{
DLCHK( retry_tran );

sqlerror( PAYMENT_SQL, "PAY", __FILE__, __LINE__, &sqlca );
payment->s_transtatus = FATAL_SQLERROR;
clientRc = FATAL_SQLERROR;

pay_debug( payment, in_payment, "PAY failed" );

EXEC SQL ROLLBACK WORK;

if ( sqlca.sqlcode != 0 )
{
sqlerror( PAYMENT_SQL, "ROLLBACK FAILED", __FILE__, __LINE__, &sqlca );
}
}

return ( clientRc );
}

// -----
// Order Status CLIENT
// -----

int ordstat_sql ( struct in_ordstat_struct * in_ordstat
, struct out_ordstat_struct * ordstat )
{
struct sqlca sqlca;

EXEC SQL BEGIN DECLARE SECTION;

struct vc_ord_in
{
short len;
char data[ 42 ];
} * in_ord;

struct vc_ord_out
{
short len;
char data[ 822 ];
} * out_ord;

EXEC SQL END DECLARE SECTION;

int clientRc = TRAN_OK;
int itemIndex = 0;

in_ord = (struct vc_ord_in *) in_ordstat;
in_ord->len = sizeof(struct in_ordstat_struct) - SPGENERAL_ADJUST;

out_ord = (struct vc_ord_out *) ordstat;
out_ord->len = sizeof(struct out_ordstat_struct) - SPGENERAL_ADJUST;

#endif DEBUGIT
ord_debug(ordstat, in_ordstat, "Client before SP call");

```

```

#endif /* DEBUGIT */

#endif SWAP_ENDIAN
SWAP_BYTE(in_ordstat->s_C_ID);
SWAP_BYTE(in_ordstat->s_W_ID);
SWAP_BYTE(in_ordstat->s_D_ID);
#endif //SWAP_ENDIAN

EXEC SQL CALL ords ( :*"in_ord, :*"out_ord );

#endif SWAP_ENDIAN
SWAP_BYTE(in_ordstat->s_C_ID);
SWAP_BYTE(in_ordstat->s_W_ID);
SWAP_BYTE(in_ordstat->s_D_ID);

SWAP_BYTE(ordstat->s_C_BALANCE);
SWAP_BYTE(ordstat->s_C_ID);
SWAP_BYTE(ordstat->s_O_ID);
SWAP_BYTE(ordstat->s_O_CARRIER_ID);
SWAP_BYTE(ordstat->s_ol_cnt);
SWAP_BYTE(ordstat->s_transtatus);
SWAP_BYTE(ordstat->deadlocks);
for (itemIndex=0; itemIndex<ordstat->s_ol_cnt; itemIndex++)
{
SWAP_BYTE(ordstat->item[ itemIndex ].s_OL_AMOUNT);
SWAP_BYTE(ordstat->item[ itemIndex ].s_OL_I_ID);
SWAP_BYTE(ordstat->item[ itemIndex ].s_OL_SUPPLY_W_ID);
SWAP_BYTE(ordstat->item[ itemIndex ].s_OL_QUANTITY);
}
#endif //SWAP_ENDIAN

if ( sqlca.sqlcode == 0 )
{
// Propagate the field we already knew into the output structure
// 60% of the time, we already new c_last (input c_id is 0)

if ( in_ordstat->s_C_ID == 0 )
{
memcpy( ordstat->s_C_LAST, in_ordstat->s_C_LAST, sizeof( ordstat-
>s_C_LAST ) );
}
else
{
ordstat->s_C_ID = in_ordstat->s_C_ID;
}
}
else
{
sqlerror( ORDSTAT_SQL, "ORD", __FILE__, __LINE__, &sqlca );
ordstat->s_transtatus = FATAL_SQLERROR;
clientRc = FATAL_SQLERROR;
}
}

#endif DEBUGIT
ord_debug(ordstat, in_ordstat, "Client after SP call");
#endif /* DEBUGIT */

if ( ordstat->s_transtatus <= FATAL_SQLERROR )
{
ord_debug(ordstat, in_ordstat, "ORD failed");
clientRc = FATAL_SQLERROR;
}

return ( clientRc );
}

// -----
// Delivery CLIENT
// -----

```

```

int delivery_sql ( struct in_delivery_struct * in_delivery
, struct out_delivery_struct * delivery )
{
struct sqlca sqlca;

EXEC SQL BEGIN DECLARE SECTION;

struct vc_del_in
{
short len;
char data[ 14 ];
} * in_del;

struct vc_del_out
{
short len;
char data[ 50 ];
} * out_del;

EXEC SQL END DECLARE SECTION;

int clientRc = TRAN_OK;
int orderIndex = 0;

in_del = (struct vc_del_in *) in_delivery;
in_del->len = sizeof(struct in_delivery_struct) - SPGENERAL_ADJUST;

out_del = (struct vc_del_out *) delivery;
out_del->len = sizeof(struct out_delivery_struct) - SPGENERAL_ADJUST;

#endif DEBUGIT
del_debug(delivery, in_delivery, "Client before SP call");
#endif /* DEBUGIT */

#endif SWAP_ENDIAN
SWAP_BYTE(in_delivery->s_W_ID);
SWAP_BYTE(in_delivery->s_O_CARRIER_ID);
#endif //SWAP_ENDIAN

EXEC SQL CALL dels ( :*"in_del, :*"out_del );

#endif SWAP_ENDIAN
SWAP_BYTE(in_delivery->s_W_ID);
SWAP_BYTE(in_delivery->s_O_CARRIER_ID);

for (orderIndex=0; orderIndex<10; orderIndex++) {
SWAP_BYTE(delivery->s_O_ID[ orderIndex ]);
}

SWAP_BYTE(delivery->s_transtatus);
SWAP_BYTE(delivery->deadlocks);
#endif //SWAP_ENDIAN

#endif DEBUGIT
del_debug(delivery, in_delivery, "Client after SP call");
#endif /* DEBUGIT */

if ( sqlca.sqlcode != 0 )
{
sqlerror( DELIVERY_SQL, "DEL", __FILE__, __LINE__, &sqlca );
delivery->s_transtatus = FATAL_SQLERROR;
clientRc = FATAL_SQLERROR;
}

if ( delivery->s_transtatus <= FATAL_SQLERROR )
{
del_debug(delivery, in_delivery, "DEL failed");
clientRc = FATAL_SQLERROR;
}

return ( clientRc );
}

```

```

}
// -----
// Stock CLIENT
// -----

#undef w_id
#undef d_id

int stocklev_sql ( struct in_stocklev_struct * in_stocklev
, struct out_stocklev_struct * stocklev )
{
    struct sqlca sqlca ;

    int clientRc = TRAN_OK ;

    EXEC SQL BEGIN DECLARE SECTION;

        // input

        sqlint32  threshold ;

        // output

        sqlint32  low_stock ;

    EXEC SQL END DECLARE SECTION;

    #define w_id  in_stocklev->s_W_ID
    #define d_id  in_stocklev->s_D_ID
    #define threshold in_stocklev->s_threshold
    #define low_stock stocklev->s_low_stock

    stocklev->deadlocks = -1 ;
    stocklev->s_transtatus = TRAN_OK ;

#ifdef DEBUGIT
    stk_debug(stocklev, in_stocklev, "Client before SQL call");
#endif /* DEBUGIT */

    retry_tran:

        stocklev->deadlocks ++ ;

    EXEC SQL BEGIN COMPOUND NOT ATOMIC STATIC

        SELECT COUNT( S_I_ID ) INTO :low_stock

            FROM ( SELECT DISTINCT S_I_ID

                FROM ORDER_LINE , STOCK , DISTRICT

                WHERE D_W_ID = :w_id
                AND D_ID = :d_id
                AND OL_O_ID < d_next_o_id
                AND OL_O_ID >= ( d_next_o_id - 20 )
                AND OL_W_ID = D_W_ID
                AND OL_D_ID = D_ID
                AND S_I_ID = OL_I_ID
                AND S_W_ID = OL_W_ID
                AND S_QUANTITY < :threshold

            ) OLS

        WITH CS
        ;

    COMMIT ;

    END COMPOUND ;

```

```

#ifdef DEBUGIT
    stk_debug(stocklev, in_stocklev, "Client after SQL call");
#endif /* DEBUGIT */

    if ( sqlca.sqlcode != 0 )
    {
        DLCHK( retry_tran ) ;

        sqlerror( STOCKLEV_SQL , "STK" , __FILE__ , __LINE__ , &sqlca);
        stocklev->s_transtatus = FATAL_SQLERROR ;
        clientRc = FATAL_SQLERROR ;

        stk_debug( stocklev, in_stocklev, "STK failed" ) ;

        EXEC SQL ROLLBACK WORK ;

        if ( sqlca.sqlcode != 0 )
        {
            sqlerror( STOCKLEV_SQL, "ROLLBACK FAILED", __FILE__, __LINE__, &sqlca ) ;
        }
    }

    return ( clientRc ) ;
}

Src.Common/Makefile

#####
####
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp. 1996 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use, duplication or
## disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#####
#####
#
# Makefile - Makefile for Src.Common
#

include $(TPCC_ROOT)/Makefile.config

#
#####
# Preprocessor, Compiler and Linker Flags
#
#####
BND_OPTS = GRANT PUBLIC \
    MESSAGES $.bnd.msg
PRP_OPTS = BINDFILE \
    OPTLEVEL 1 \
    ISOLATION RR \
    MESSAGES $.prep.msg \
    LEVEL $(TPCC_VERSION) \
    NOLINEMACRO

INCLUDES = -I$(TPCC_SQLLIB)$(SLASH)include -I$(TPCC_ROOT)$(SLASH)include

CFLAGS = $(CFLAGS_OS) $(CFLAGS_DEBUG) $(INCLUDES) \
    -DSQLA_NOLINES -D$(DB2EDITION) -D$(DB2VERSION) \
    -D$(TPCC_SPTYPE)

```

```

UTIL_OBJ = tpcmisc$(OBJEXT) tpcdbg$(OBJEXT)
UTIL_OBJ_DB2 = tpcctx$(OBJEXT)

#
#####
# User Targets
#
#####

all:  dbgen connect $(UTIL_OBJ_DB2) disconnect

dbgen: $(UTIL_OBJ)

clean:
    - $(ERASE) *$(OBJEXT) *.bnd *.msg tpcctx.c

#
#####
# Helper Targets
#
#####

connect:
    - db2 connect to $(TPCC_DBNAME)

disconnect:
    - db2 connect reset
    - db2 terminate

rebind: connect
    db2 bind tpcctx.bnd $(BND_OPTS)

#
#####
# Build Rules
#
#####

.SUFFIXES:
.SUFFIXES: $(OBJEXT) .c .sqc

.sqc.c:
    @echo "Prepping $.sqc"
    -db2 prep $.sqc $(PRP_OPTS)
    @echo "Binding $.bnd"
    db2 bind $.bnd $(BND_OPTS)

#
#####
# Dependencies
#
#####

# Source
tpcdbg$(OBJEXT):  tpcdbg.c
tpcctx$(OBJEXT):  tpcctx.c
tpcmisc$(OBJEXT): tpcmisc.c

# Headers
tpcdbg.c: $(TPCC_ROOT)/include/db2tpcc.h

Src.Common/tpcctx.sqc

/*****
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2005

```

```

** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****

/*
 *
 * tpcctx.sqc - TPCC context code
 *
 */

#include <string.h>
#include <sqlutil.h>
#include "db2tpcc.h"
#include "tpccdbg.h"

int connect_to_TM(char *in_dbname);
int connect_to_TM_auth(char *in_dbname, char *in_username, char *in_password);
int disconnect_from_TM(void);

int connect_to_TM(char *in_dbname)
{
    return connect_to_TM_auth(in_dbname, "", "");
}

int connect_to_TM_auth(char *in_dbname, char *in_username, char *in_password)
{
    SQL_STRUCTURE sqlca sqlca;
    int ConnectSQLCODE = 0;

    EXEC SQL BEGIN DECLARE SECTION;
    char dbname[9];
    char username[129];
    char password[15];
    EXEC SQL END DECLARE SECTION;

    /* Copy 9 characters - 8 for dbname, 1 for NULL */
    strncpy(dbname,in_dbname,9);
    if (strcmp(in_username,"") == 0)
    {
        EXEC SQL CONNECT TO :dbname IN SHARE MODE;
    } else {
        strncpy(username,in_username,128);
        strncpy(password,in_password,14);
        EXEC SQL CONNECT TO :dbname IN SHARE MODE USER :username
        USING :password;
    }

    ConnectSQLCODE = SQLCODE;
    if (ConnectSQLCODE != 0)
    {
        sqlerror( CLIENT_SQL, "CONNECT", __FILE__, __LINE__, &sqlca);

        return ConnectSQLCODE;
    }

    return 0;
}

int disconnect_from_TM(void)
{
    SQL_STRUCTURE sqlca sqlca;
    int DisconnectSQLCODE = 0;

    EXEC SQL CONNECT RESET;

    DisconnectSQLCODE = SQLCODE;
    if (DisconnectSQLCODE != 0) {
        sqlerror( CLIENT_SQL, "DISCONNECT", __FILE__, __LINE__, &sqlca);
    }
}

```

```

}

if (DisconnectSQLCODE) {
    return DisconnectSQLCODE;
}
return 0;
}

```

Src.Common/tpccdbg.c

```

*****
** Licensed Materials - Property of IBM
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****

/*
 * tccdbg.c - Debugging Routines
 *
 */

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

#include "sqlca.h"
#include "sql.h"
#include "db2tpcc.h"
#include "tpccdbg.h"

#define DEBUG_FILENAME_SZ 128
#define DEBUG_PATH_SIZE 128

void del_print();
void new_print();
void ord_print();
void pay_print();
void stk_print();

void current_tmstamp(char *buf);

static int debugInit = 0;
static char debugPath[DEBUG_PATH_SIZE] = "";

/*-----*/
/* InitializeDebug */
/*-----*/
inline void InitializeDebug(void) {
    if (debugInit == 0) {
        char *p = getenv("TPCC_DEBUGDIR");
        if (p) {
            strncpy(debugPath, p, DEBUG_PATH_SIZE);
        } else {
            strcpy(debugPath, "C:\\temp");
        }
        strcat(debugPath, "\\");
    }
    debugInit = 1;
}

```

```

/*-----*/
/* sqlerror */
/*-----*/
void sqlerror(int tranType, char *msg, char *file, int line, SQL_STRUCTURE sqlca *psqlca)
{
    FILE *err_fp = NULL;
    char err_fn[DEBUG_PATH_SIZE + DEBUG_FILENAME_SZ];
    char tranName[16];
    int j,k;
    char timeStamp[27];
    char errStr[512] = "";

    InitializeDebug();
    strncpy(err_fn, debugPath, DEBUG_PATH_SIZE);
    current_tmstamp(&timeStamp[0]);
    timeStamp[19] = (char)NULL;

    switch(tranType)
    {
        case NEWORD_SQL:
            // sprintf(err_fn, "%d.err.out", getpid());
            strcat(err_fn, "new.err.out");
            strcpy(tranName, "NEW_ORDER");
            break;

        case DELIVERY_SQL:
            // sprintf(err_fn, "%d.err.out", getpid());
            strcat(err_fn, "del.err.out");
            strcpy(tranName, "DELIVERY");
            break;

        case PAYMENT_SQL:
            // sprintf(err_fn, "%d.err.out", getpid());
            strcat(err_fn, "pay.err.out");
            strcpy(tranName, "PAYMENT");
            break;

        case ORDSTAT_SQL:
            // sprintf(err_fn, "%d.err.out", getpid());
            strcat(err_fn, "ord.err.out");
            strcpy(tranName, "ORDER_STAT");
            break;

        case STOCKLEV_SQL:
            //sprintf(err_fn, "%d.err.out", getpid());
            strcat(err_fn, "stk.err.out");
            strcpy(tranName, "STOCK_LVL");
            break;

        case 0:
            strcat(err_fn, "cli.err.out");
            strcpy(tranName, "CLIENT");
            break;

        default:
            return;
    }

    /* Generate Formatted Error Message */
    sqlaintp(errStr, 512, 78, psqlca);

    if ((err_fp = fopen(err_fn, "a+")) == NULL)
    {
        return;
    }

    fprintf(err_fp, "-----\n");
    fprintf(err_fp, "Transaction: %s (%s)\n", tranName, msg);
}

```



```

fprintf(err_fp, "FILE %s (%u)\n", file, line);
fprintf(err_fp, "SQLCODE %d ", psqlca->sqlcode);
fprintf(err_fp, "PID %d ", getpid());
fprintf(err_fp, "TIME %s\n", timeStamp);
fprintf(err_fp, "-----\n");
fprintf(err_fp, "%s", errStr);
fprintf(err_fp, "-----\n");

if (psqlca->sqlerrmc[0] != ' ' || psqlca->sqlerrmc[1] != ' ')
{
    fprintf(err_fp, "slerrmc: ");

    for(j = 0; j < 5; j++)
    {
        for(k = 0; k < 16; k++) {
            int pos = j * 16 + k;
            if (pos < 70) fprintf(err_fp, "%02x ", psqlca->sqlerrmc[pos]);
            else fprintf(err_fp, " ");
        }
        fprintf(err_fp, " |");
        for(k = 0; k < 16; k++) {
            int pos = j * 16 + k;
            char c = ' ';
            if (pos < 70) {
                c = psqlca->sqlerrmc[pos];
                if (!strcmp(c, " "));
            }
            fprintf(err_fp, "%c", c);
        }
        fprintf(err_fp, "\n");
        if (j < 4) fprintf(err_fp, " ");
    }

    fprintf(err_fp, "sqlerrp: ");
    for(j = 0; j < 8; j++)
        fprintf(err_fp, "%c", psqlca->sqlerrp[j]);
    fprintf(err_fp, "\n");

    fprintf(err_fp, "sqlerrd: ");
    for(j = 0; j < 6; j++)
        fprintf(err_fp, "%c", psqlca->sqlerrd[j]);
    fprintf(err_fp, "\n");

    if (psqlca->sqlwarn[0] != ' ')
    {
        fprintf(err_fp, "sqlwarn: ");
        for(j = 0; j < 8; j++)
            fprintf(err_fp, "%c", psqlca->sqlwarn[j]);
        fprintf(err_fp, "\n");
    }

    fprintf(err_fp, "\n");

    fclose(err_fp);
}

/*-----*/
/* del_debug */
/*-----*/
void del_debug (struct out_delivery_struct *delivery_ptr,
                struct in_delivery_struct *in_delivery,
                char *msg)
{
    char debug_fn[DEBUG_PATH_SIZE + DEBUG_FILENAME_SZ];

    InitializeDebug();
    strncpy(debug_fn, debugPath, DEBUG_PATH_SIZE);
    strcat(debug_fn, "new.debug.out");
    new_print(newword_ptr, in_newword, debug_fn, msg);
}

/*-----*/
/* new_print */
/*-----*/
void new_print (struct out_newword_struct *newword_ptr,
                struct in_newword_struct *in_newword,
                char *filename,

```

```

                char *msg);
}

/*-----*/
/* del_print */
/*-----*/
void del_print (struct out_delivery_struct *delivery_ptr,
                struct in_delivery_struct *in_delivery,
                char *filename,
                char *msg)
{
    FILE *debug_fp;
    char timeStamp[27];
    int j;

    current_tmstamp(&timeStamp[0]);
    timeStamp[19] = (char)NULL;

    if ((debug_fp = fopen(filename, "a+")) == NULL)
    {
        return;
    }

    fprintf(debug_fp, "Delivery debug information follows %s (%s)\n", timeStamp, msg);
    fprintf(debug_fp, "PID %d ", getpid());
    fprintf(debug_fp, "-----\n");

    fprintf(debug_fp, "in_delivery_struct {\n");
    fprintf(debug_fp, "ts_W_ID = %d (%X)\n",
            in_delivery->s_W_ID, in_delivery->s_W_ID);
    fprintf(debug_fp, "ts_O_CARRIER_ID = %d (%X)\n",
            in_delivery->s_O_CARRIER_ID, in_delivery->s_O_CARRIER_ID);
    fprintf(debug_fp, ")\n\n");

    fprintf(debug_fp, "out_delivery_struct {\n");
    fprintf(debug_fp, "ts_transtatus = %d (%X)\n",
            delivery_ptr->s_transtatus, delivery_ptr->s_transtatus);
    fprintf(debug_fp, "tdeadlocks = %d (%X)\n",
            delivery_ptr->deadlocks, delivery_ptr->deadlocks);

    for (j = 0; j < 10; j++) {
        fprintf(debug_fp, "\tts_O_ID[%d] = %d\n",
                j, delivery_ptr->s_O_ID[j]);
    }
    fprintf(debug_fp, "\t)\n\n");
    fclose(debug_fp);
}

/*-----*/
/* new_debug */
/*-----*/
void new_debug (struct out_newword_struct *newword_ptr,
                struct in_newword_struct *in_newword,
                char *msg)
{
    char debug_fn[DEBUG_PATH_SIZE + DEBUG_FILENAME_SZ];

    InitializeDebug();
    strncpy(debug_fn, debugPath, DEBUG_PATH_SIZE);
    strcat(debug_fn, "new.debug.out");
    new_print(newword_ptr, in_newword, debug_fn, msg);
}

/*-----*/
/* new_print */
/*-----*/
void new_print (struct out_newword_struct *newword_ptr,
                struct in_newword_struct *in_newword,
                char *filename,

```

```

                char *msg)
{
    FILE *debug_fp;
    char timeStamp[27];
    int j, items;

    current_tmstamp(&timeStamp[0]);
    timeStamp[19] = (char)NULL;

    if ((debug_fp = fopen(filename, "a+")) == NULL)
    {
        return;
    }

    fprintf(debug_fp, "New order debug information follows %s (%s)\n", timeStamp, msg);
    fprintf(debug_fp, "PID %d ", getpid());
    fprintf(debug_fp, "-----\n");

    fprintf(debug_fp, "in_newword_struct {\n");

    fprintf(debug_fp, "\tts_C_ID = %d (%X)\n",
            in_newword->s_C_ID, in_newword->s_C_ID);
    fprintf(debug_fp, "\tts_W_ID = %d (%X)\n",
            in_newword->s_W_ID, in_newword->s_W_ID);
    fprintf(debug_fp, "\tts_D_ID = %d (%X)\n",
            in_newword->s_D_ID, in_newword->s_D_ID);
    fprintf(debug_fp, "\tts_O_OL_CNT = %d (%X)\n",
            in_newword->s_O_OL_CNT, in_newword->s_O_OL_CNT);
    fprintf(debug_fp, "\tts_all_local = %d (%X)\n",
            in_newword->s_all_local, in_newword->s_all_local);
    // fprintf(debug_fp, "\tts_transtatus = %d (%X)\n",
    // in_newword->s_transtatus, in_newword->s_transtatus);
    // fprintf(debug_fp, "\tduplicate_items= %d (%X)\n",
    // in_newword->duplicate_items, in_newword->duplicate_items);

    fprintf(debug_fp, "\titems {\n");
    items = in_newword->s_O_OL_CNT;
    for (j=0; j<items; j++) {
        if(j != 0)
            fprintf(debug_fp, "\n");
        fprintf(debug_fp, "\tts_OL_I_ID[%d] = %d (%X)\n",
                j, in_newword->in_item[j].s_OL_I_ID, in_newword->in_item[j].s_OL_I_ID);
        fprintf(debug_fp, "\tts_OL_SUPPLY_W_ID[%d] = %d (%X)\n",
                j, in_newword->in_item[j].s_OL_SUPPLY_W_ID, in_newword->
                in_item[j].s_OL_SUPPLY_W_ID);
        fprintf(debug_fp, "\tts_OL_QUANTITY[%d] = %d (%X)\n",
                j, in_newword->in_item[j].s_OL_QUANTITY, in_newword->
                in_item[j].s_OL_QUANTITY);
    }
    fprintf(debug_fp, "\t)\n\n");

    fprintf(debug_fp, "out_newword_struct {\n");
    fprintf(debug_fp, "\tts_C_LAST = %s\n",
            newword_ptr->s_C_LAST);
    fprintf(debug_fp, "\tts_C_CREDIT = %s\n",
            newword_ptr->s_C_CREDIT);
    fprintf(debug_fp, "\tts_W_TAX = %04.4f\n",
            newword_ptr->s_W_TAX);
    fprintf(debug_fp, "\tts_D_TAX = %04.4f\n",
            newword_ptr->s_D_TAX);
    fprintf(debug_fp, "\tts_C_DISCOUNT = %04.4f\n",
            newword_ptr->s_C_DISCOUNT);
    fprintf(debug_fp, "\tts_O_ID = %d (%X)\n",
            newword_ptr->s_O_ID, newword_ptr->s_O_ID);
    fprintf(debug_fp, "\tts_O_OL_CNT = %d (%X)\n",
            newword_ptr->s_O_OL_CNT, newword_ptr->s_O_OL_CNT);
    fprintf(debug_fp, "\tts_O_ENTRY_D = %s\n",
            newword_ptr->s_O_ENTRY_D_time);
    fprintf(debug_fp, "\tts_total_amount = %02.f\n",
            newword_ptr->s_total_amount);
}

```



```

    payment_ptr->s_D_STREET_2);
fprintf(debug_fp,"ts_D_CITY   = %s\n",
    payment_ptr->s_D_CITY);
fprintf(debug_fp,"ts_D_STATE  = %s\n",
    payment_ptr->s_D_STATE);
fprintf(debug_fp,"ts_D_ZIP   = %s\n",
    payment_ptr->s_D_ZIP);
fprintf(debug_fp,"ts_C_FIRST  = %s\n",
    payment_ptr->s_C_FIRST);
fprintf(debug_fp,"ts_C_MIDDLE = %s\n",
    payment_ptr->s_C_MIDDLE);
fprintf(debug_fp,"ts_C_LAST   = %s\n",
    payment_ptr->s_C_LAST);
fprintf(debug_fp,"ts_C_STREET_1 = %s\n",
    payment_ptr->s_C_STREET_1);
fprintf(debug_fp,"ts_C_STREET_2 = %s\n",
    payment_ptr->s_C_STREET_2);
fprintf(debug_fp,"ts_C_CITY   = %s\n",
    payment_ptr->s_C_CITY);
fprintf(debug_fp,"ts_C_STATE  = %s\n",
    payment_ptr->s_C_STATE);
fprintf(debug_fp,"ts_C_ZIP   = %s\n",
    payment_ptr->s_C_ZIP);
fprintf(debug_fp,"ts_C_PHONE  = %s\n",
    payment_ptr->s_C_PHONE);
fprintf(debug_fp,"ts_C_SINCE  = %s\n",
    payment_ptr->s_C_SINCE_time);
fprintf(debug_fp,"ts_C_CREDIT = %s\n",
    payment_ptr->s_C_CREDIT);
fprintf(debug_fp,"ts_C_DATA   = %s\n",
    payment_ptr->s_C_DATA);
fprintf(debug_fp,"ts_transtatus = %d (%X)\n",
    payment_ptr->s_transtatus,payment_ptr->s_transtatus);
fprintf(debug_fp,"tdeadlocks  = %d (%X)\n",
    payment_ptr->deadlocks,payment_ptr->deadlocks);
fprintf(debug_fp,"\n)\n\n");
fclose(debug_fp);
}

/*-----*/
/* stk_debug */
/*-----*/
void stk_debug (struct out_stocklev_struct *stocklev,
    struct in_stocklev_struct *in_stocklev,
    char *msg)
{
    char debug_fn[DEBUG_PATH_SIZE + DEBUG_FILENAME_SZ];

    InitializeDebug();
    strncpy(debug_fn, debugPath, DEBUG_PATH_SIZE);
    strcat(debug_fn, "stk.debug.out");
    stk_print(stocklev, in_stocklev, debug_fn, msg);
}

/*-----*/
/* stk_print */
/*-----*/
void stk_print (struct out_stocklev_struct *stocklev,
    struct in_stocklev_struct *in_stocklev,
    char *filename,
    char *msg)
{
    FILE *debug_fp;
    char timeStamp[27];

    current_tmstamp(&timeStamp[0]);
    timeStamp[19] = (char)NULL;

    if ((debug_fp = fopen(filename, "a+")) == NULL)

```

```

{
    return;
}

fprintf(debug_fp,"Stock level debug information follows %s (%s)\n", timeStamp, msg);
fprintf(debug_fp, " PID %d ", getpid());
fprintf(debug_fp,"n=====n");

fprintf(debug_fp,"in_stocklev_struct {n");
fprintf(debug_fp,"ts_W_ID   = %d (%X)\n",
    in_stocklev->s_W_ID, in_stocklev->s_W_ID);
fprintf(debug_fp,"ts_D_ID   = %d (%X)\n",
    in_stocklev->s_D_ID, in_stocklev->s_D_ID);
fprintf(debug_fp,"ts_threshold = %d (%X)\n",
    in_stocklev->s_threshold, in_stocklev->s_threshold);
fprintf(debug_fp,"}\n\n");

fprintf(debug_fp,"out_stocklev_struct {n");
fprintf(debug_fp,"ts_transtatus = %d (%X)\n",
    stocklev->s_transtatus, stocklev->s_transtatus);
fprintf(debug_fp,"tdeadlocks  = %d (%X)\n",
    stocklev->deadlocks, stocklev->deadlocks);
fprintf(debug_fp,"ts_low_stock  = %d (%X)\n",
    stocklev->s_low_stock, stocklev->s_low_stock);
fprintf(debug_fp,"}\n\n");
fclose(debug_fp);
}

void current_tmstamp(char *buf)
{
    time_t t = time(NULL);
    strncpy(buf,ctime(&t),19);
}

include/db2tpcc.h
/*-----*/
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
/*-----*/

/*
 * db2tpcc.h - Macros and Miscellany
 */

#ifndef __DB2TPCC_H
#define __DB2TPCC_H

#include <sys/types.h>

#include "lval.h"

/* ----- */
/* Transaction Return Codes (s_transtatus) */
/* ----- */

#define INVALID_ITEM      100
#define TRAN_OK           0
#define FATAL_SQLERROR   -1
/* ----- */

```

```

/* Definition of Unused and Bad Items */
/* ----- */
/* Define unused item ID to be 0. This allows the SUT to determine the */
/* number of items in the order as required by 2.4.1.3 and 2.4.2.2 since */
/* the assumption that any item with OL_I_ID = 0 is unused will be true. */
/* This in turn requires that the value used for an invalid item is */
/* equal to ITEMS + 1. */
/* ----- */

#define INVALID_ITEM_ID (2 * ITEMS) + 1
#define UNUSED_ITEM_ID 0

#define MIN_WAREHOUSE 1
#define MAX_WAREHOUSE WAREHOUSES

/* ----- */
/* NURand Constants */
/* C_C_LAST_RUN and C_C_LAST_LOAD must adhere to clause 2.1.6. */
/* Analysis indicates that a C_LAST delta of 85 is optimal. */
/* ----- */

#define C_C_LAST_RUN 88
#define C_C_LAST_LOAD 173
#define C_C_ID 319
#define C_OL_I_ID 3849
#define A_C_LAST 255
#define A_C_ID 1023
#define A_OL_I_ID 8191

/* ----- */
/* Transaction Type Identifiers */
/* ----- */

#define CLIENT_SQL 0
#define NEWORD_SQL 1
#define PAYMENT_SQL 2
#define ORDSTAT_SQL 3
#define DELIVERY_SQL 4
#define STOCKLEV_SQL 5

#define SPGENERAL_PAD 3
#define SPGENERAL_ADJUST sizeof(int16_t)

struct in_neword_struct {
    int16_t len;
    int16_t pad[SPGENERAL_PAD];
    struct in_items_struct {
        int32_t s_OL_I_ID;
        int32_t s_OL_SUPPLY_W_ID;
        int16_t s_OL_QUANTITY;
        int16_t pad1[3];
    } in_item[15];
    int32_t s_C_ID;
    int32_t s_W_ID;
    int16_t s_D_ID;
    int16_t s_O_OL_CNT; /* init by SUT */
    int16_t s_all_local;
    int16_t duplicate_items;
};

struct out_neword_struct {
    int16_t len;
    int16_t pad[SPGENERAL_PAD];
    struct items_struct {
        float s_I_PRICE;
        float s_OL_AMOUNT;
        int16_t s_S_QUANTITY;
        int16_t pad2;
        char s_I_NAME[25];
        char s_brand_generic;
    } item[15];
};

```

```

float s_W_TAX;
float s_D_TAX;
float s_C_DISCOUNT;
float s_total_amount;
int32_t s_O_ID;
int16_t s_O_OL_CNT;
int16_t s_transtatus;
int16_t deadlocks;
char s_C_LAST[17];
char s_C_CREDIT[3];
char s_O_ENTRY_D_time[27];

```

```
};
```

```

struct in_payment_struct {
int16_t len;
int16_t pad[SPGENERAL_PAD];
float s_H_AMOUNT;
int32_t s_W_ID;
int32_t s_C_W_ID;
int32_t s_C_ID;
int16_t s_C_D_ID;
int16_t s_D_ID;
char s_C_LAST[17];
};

```

```

struct out_payment_struct {
int16_t len;
int16_t pad[SPGENERAL_PAD];
double s_C_CREDIT_LIM;
double s_C_BALANCE;
float s_C_DISCOUNT;
int32_t s_C_ID;
int16_t s_transtatus;
int16_t deadlocks;
char s_W_STREET_1[21];
char s_W_STREET_2[21];
char s_W_CITY[21];
char s_W_STATE[3];
char s_W_ZIP[10];
char s_D_STREET_1[21];
char s_D_STREET_2[21];
char s_D_CITY[21];
char s_D_STATE[3];
char s_D_ZIP[10];
char s_C_FIRST[17];
char s_C_MIDDLE[3];
char s_C_LAST[17];
char s_C_STREET_1[21];
char s_C_STREET_2[21];
char s_C_CITY[21];
char s_C_STATE[3];
char s_C_ZIP[10];
char s_C_PHONE[17];
char s_C_CREDIT[3];
char s_C_DATA[201];
char s_H_DATE_time[27];
char s_C_SINCE_time[27];
};

```

```

struct in_ordstat_struct {
int16_t len;
int16_t pad[SPGENERAL_PAD];
int32_t s_C_ID;
int32_t s_W_ID;
int16_t s_D_ID;
int16_t pad1[3];
char s_C_LAST[17];
};

```

```

struct out_ordstat_struct {
int16_t len;
int16_t pad[SPGENERAL_PAD];
double s_C_BALANCE;
int32_t s_C_ID;
int32_t s_O_ID;
int16_t s_O_CARRIER_ID;
int16_t s_ol_cnt;
int16_t pad1[2];
struct oitems_struct {
double s_OL_AMOUNT;
int32_t s_OL_I_ID;
int32_t s_OL_SUPPLY_W_ID;
int16_t s_OL_QUANTITY;
int16_t pad2;
char s_OL_DELIVERY_D_time[27];
} item[15];
int16_t s_transtatus;
int16_t deadlocks;
char s_C_FIRST[17];
char s_C_MIDDLE[3];
char s_C_LAST[17];
char s_O_ENTRY_D_time[27];
int16_t pad3[2];
};

```

```

struct in_delivery_struct {
int16_t len;
int16_t pad[SPGENERAL_PAD];
int32_t s_W_ID;
int16_t s_O_CARRIER_ID;
};

```

```

struct out_delivery_struct {
int16_t len;
int16_t pad[SPGENERAL_PAD];
int32_t s_O_ID[10];
int16_t s_transtatus;
int16_t deadlocks;
};

```

```

struct in_stocklev_struct {
int16_t len;
int16_t pad[SPGENERAL_PAD];
int32_t s_threshold;
int32_t s_W_ID;
int16_t s_D_ID;
};

```

```

struct out_stocklev_struct {
int16_t len;
int16_t pad[SPGENERAL_PAD];
int32_t s_low_stock;
int16_t s_transtatus;
int16_t deadlocks;
};

```

```

/* ***** */
/* Transaction Prototypes */
/* ***** */

```

```

#ifdef __cplusplus
extern "C" {
#endif

extern int neword_sql(struct in_neword_struct*, struct out_neword_struct*);
extern int payment_sql(struct in_payment_struct*, struct out_payment_struct*);
extern int ordstat_sql(struct in_ordstat_struct*, struct out_ordstat_struct*);
extern int delivery_sql(struct in_delivery_struct*, struct out_delivery_struct*);
extern int stocklev_sql(struct in_stocklev_struct*, struct out_stocklev_struct*);

```

```

#ifdef __cplusplus
}
#endif

```

```

/* ***** */
/* DB2 Connect/Disconnect & Thread Context Wrappers */
/* ***** */

```

```

#ifdef __cplusplus
extern "C" {
#endif

```

```

extern int connect_to_TM(char*);
extern int connect_to_TM_auth(char*, char*, char*);
extern int disconnect_from_TM(void);

```

```

#ifdef __cplusplus
}
#endif

```

```
#endif // __DB2TPCC_H
```

include/tpccapp.h

```

/* ***** */
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2004
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
/* ***** */

```

```

/*
 * tpccapp.h - Application Macros
 */

```

```

#ifdef __TPCCAPP_H
#define __TPCCAPP_H

```

```

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

```

```

#include "sqlenv.h"
#define daricall __stdcall

```

```

#include "sqlca.h"
#include "sqlcodes.h"

```

```

#ifdef SWAP_ENDIAN
#define SWAP_BYTE(Var) SwapEndian((void*)&Var, sizeof(Var))

```

```

/* ***** */
FUNCTION: SwapEndian
PURPOSE: Swap the byte order of a structure
EXAMPLE: int I=0x12345678; SWAP_BYTE(I); I => 0x78563412;
IMPLEMENTATION: Fold Addr in half, swap header & tail by XOR op
e.g.: *a = 0x12 [ Addr + 0];
      *b = 0x78 [ Add + 4 - 0 - 1 = Addr+3];
      *a ^= *b; // sets *a to 0x6A
      *b ^= *a; // sets *b to 0x12

```

```

        *a ^= *b; // sets *a to 0x78
Now *a => 0x78 && *b => 0x12
-----/

void SwapEndian(void *Addr, int nb)
{
    int i;
    for (i=0; i<nb/2; i++)
    {
        char *a = (char*)Addr+i;
        char *b = (char*)Addr+(nb-i-1);

        *a ^= *b;
        *b ^= *a;
        *a ^= *b;
    }
}
#endif //SWAP_ENDIAN

/-----/
/* SQLCODE Macros */
/-----/

#define DLCHK(a) \
if (sqlca.sqlcode == SQL_RC_E911) { goto a; }

/* ***** */
/* In NOT ATOMIC COMPOUND SQL, all statements will be executed, but not */
/* all will necessarily complete successfully. We can use sqlerrd(4) to */
/* determine how many statements succeeded, but this won't tell us what */
/* statements failed. In order to determine this, we need to look at */
/* sqlerrmc, which has the following structure: HHHXNNNSSSSXNNNSSSS... */
/* (See the docs for more details.) Since we're interested in the first */
/* failing statement, we can look at elements 5 and 6, which will contain */
/* the first two digits of NNN (which is right-padded with spaces). We */
/* need to look at the first two digits since some of our compound blocks */
/* have > 9 statements. We convert these digits from ASCII to an int and */
/* set 'last' to this value. */
/* ***** */

#define NACOMPCHK(last) \
if (sqlca.sqlcode != SQL_RC_E1339) { last = -1; } \
else { int a = ((sqlca.sqlerrmc[4] == 0x20) ? 0 : sqlca.sqlerrmc[4]-0x30); \
int b = ((sqlca.sqlerrmc[5] == 0x20) ? 0 : sqlca.sqlerrmc[5]-0x30); \
if (b == 0) { last = a; } else { last = a * 10 + b; } \
}

#endif // __TPCCAPP_H

```

include/lval.h

/* lval.h - generated automatically at 20060905.1052 */

```

#ifndef __LVAL_H
#define __LVAL_H
#define WAREHOUSES 28032
#define DISTRICTS_PER_WAREHOUSE 10
#define CUSTOMERS_PER_DISTRICT 3000
#define ITEMS 100000
#define STOCK_PER_WAREHOUSE 100000
#define MIN_ORDER 15
#define MAX_ORDER 5
#define NU_ORDERS_PER_DISTRICT 900
#endif // __LVAL_H

```

include/tpccdbg.h

```

/-----/
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
/-----/

/*
 * tpccdbg.h - Debugging Macros
 *
 */

#ifndef __TPCCDBG_H
#define __TPCCDBG_H

#ifdef __cplusplus
extern "C" {
#endif

extern void sqlerror (int tranType, char *msg, char *file, int line,
SQL_STRUCTURE sqlca *psqlca);

extern void new_debug (struct out_neword_struct *neword_ptr,
struct in_neword_struct *in_neword_ptr,
char *msg);
extern void pay_debug (struct out_payment_struct *payment_ptr,
struct in_payment_struct *in_payment_ptr,
char *msg);
extern void ord_debug (struct out_ordstat_struct *ordstat_ptr,
struct in_ordstat_struct *in_ordstat_ptr,
char *msg);
extern void del_debug (struct out_delivery_struct *delivery_ptr,
struct in_delivery_struct *in_delivery_ptr,
char *msg);
extern void stk_debug (struct out_stocklev_struct *stocklev_ptr,
struct in_stocklev_struct *in_stocklev_ptr,
char *msg);

extern void new_print (struct out_neword_struct *neword_ptr,
struct in_neword_struct *in_neword_ptr,
char *filename,
char *msg);
extern void pay_print (struct out_payment_struct *payment_ptr,
struct in_payment_struct *in_payment_ptr,
char *filename,
char *msg);
extern void ord_print (struct out_ordstat_struct *ordstat_ptr,
struct in_ordstat_struct *in_ordstat_ptr,
char *filename,
char *msg);
extern void del_print (struct out_delivery_struct *delivery_ptr,
struct in_delivery_struct *in_delivery_ptr,
char *filename,
char *msg);
extern void stk_print (struct out_stocklev_struct *stocklev_ptr,
struct in_stocklev_struct *in_stocklev_ptr,
char *filename,
char *msg);

#ifdef __cplusplus
}
#endif

#endif // __TPCCDBG_H

```

tpccenv.bat

```

@REM *****
@REM Licensed Materials - Property of IBM
@REM
@REM Governed under the terms of the International
@REM License Agreement for Non-Warranted Sample Code.
@REM
@REM (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
@REM All Rights Reserved.
@REM
@REM US Government Users Restricted Rights - Use, duplication or
@REM disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
@REM *****

@REM tpccenv.bat - Windows Environment Setup
@REM

@REM The Kit Version
set TPCC_VERSION=CK060815

@REM The DB2 Instance Name (for DB2)
set DB2INSTANCE=%USERNAME%

@REM The OS being used (i.e. "WINDOWS")
set PLATFORM=WINDOWS

@REM The type of make command and slash used by the OS
@REM (i.e. UNIX - "/", WINDOWS - "\")
@REM These are referenced all over the kit.
set SLASH=
set MAKE=nmake

@REM Specifies whether or not to use dari stored proc's for the TPC-C driver. Set to
either DARIVERISION or NONDARI;
@REM set TPCC_SPTYPE=NOSP
@REM set TPCC_SPTYPE=SPGENERAL2
set TPCC_SPTYPE=SPGENERAL
@REM set TPCC_SPTYPE=DARI2SQLDA

set DB2VERSION=v8

@REM The schema name is typically the SQL authorization ID (or username).
@REM This is required for runstats and EEE.
set TPCC_SCHEMA=%USERNAME%

@REM DB2 EE/EEE Configuration
set DB2EDITION=EE
@REM set DB2EDITION=EEE
set DB2NODE=0
@REM set to the number of nodes you have. Set to 1 for EE.
set DB2NODES=1

@REM TPCC General Configuration
@REM ** IMPORTANT NOTE **
@REM The kit is not guaranteed to work properly if TPCC_ROOT or TPCC_SQLLIB
@REM have spaces in them. If you absolutely must use paths with spaces,
@REM then the entire path must be surrounded by double quotes.
@REM For example: HOME="C:\Program Files\IBM"
set HOME=C:\home\tpcc
set TPCC_DBNAME=TPCC
set TPCC_ROOT=%HOME%\tpc-c.ibm
set TPCC_SQLLIB=C:\Progra-1\IBM\sqllib
set TPCC_RUNDATA=%HOME%\tpc-c.ibm\tpccdata

@REM TPCC Debug Configuration
@REM This is the path where all error and debug logs are placed.
@REM To get debugging from within the stored procedures, you must

```

```
@REM set DB2ENVLIST="TPCC_DEBUGDIR" in tpcc.config.
set TPCC_DEBUGDIR=c:\temp
```

```
@REM Specifies where stored procedures should be placed and if they should
@REM be fenced.
set TPCC_SPDIR=%TPCC_SQLLIB%\function
set TPCC_FENCED=NO
```

9.2. Transaction Code

Makefile.config

```
#####
####
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp. 1996 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use, duplication or
## disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#####

#
# Makefile.config - Linux 64-bit
#
#

# Make Configuration
MAKE=make

# Compiler Configuration.
# CFLAGS_DEBUG may be set to "-g", "-DDEBUGIT" "-g -DDEBUGIT" or left blank
CC=cc
CFLAGS_OS=-DSQLUNIX -DSQLLinux -O2 -fpic -m64
CFLAGS_OUT=-o
CFLAGS_DEBUG=

# Linker Configuration
LD_EXEC=gcc
LD_STORP=gcc
LDFLAGS_EXEC=
LDFLAGS_SHLIB=-shared
LDFLAGS_STORP=$(LDFLAGS_SHLIB)
LDFLAGS_LIB=-L$(TPCC_SQLLIB)/lib -ldb2 -m64
LDFLAGS_OUT=-o

# Library Configuration
AR=ar
ARFLAGS=-rv
ARFLAGS_LIB=
ARFLAGS_OUT=

# OS Commands
ERASE=rm -f
ERASEDIR=$(ERASE) -R
MOVE=mv
COPY=cp
```

```
# OS File Extensions & Path Separators
OBJEXT=.o
LIBEXT=.a
SHLIBEXT=.so
BINEXT=
SLASH=/
CMDSEP=;
```

Src.Common/Makefile

```
#####
####
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use, duplication or
## disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#####

#
# Makefile - Makefile for Src.Common
#
#

include $(TPCC_ROOT)/Makefile.config

#
#####
# Preprocessor, Compiler and Linker Flags
#
#####

BND_OPTS = GRANT PUBLIC \
  MESSAGES $*.bnd.msg
PRP_OPTS = BINDFILE \
  OPTLEVEL 1 \
  ISOLATION RR \
  MESSAGES $*.prep.msg \
  LEVEL $(TPCC_VERSION) \
  NOLINEMACRO

INCLUDE = -I$(TPCC_SQLLIB)/include -I$(TPCC_ROOT)/include

CFLAGS = $(CFLAGS_OS) $(CFLAGS_DEBUG) $(INCLUDE) \
  -DSQLA_NOLINES -D$(DB2EDITION) -D$(DB2VERSION) \
  -D$(TPCC_SPTYPE)

UTIL_OBJ_DBG = tpccdbg$(OBJEXT)
UTIL_OBJ_GEN = tpccmisc$(OBJEXT)
UTIL_OBJ_DB2 = tpccctx$(OBJEXT)

#
#####
# User Targets
#
#####

all: $(UTIL_OBJ_DBG) $(UTIL_OBJ_GEN) connect $(UTIL_OBJ_DB2) disconnect

dbgen: $(UTIL_OBJ_GEN)

clean:
  - $(ERASE) *$(OBJEXT) *.bnd *.msg tpccctx.c
```

```
#
#####
# Helper Targets
#
#####

connect:
  - db2 connect to $(TPCC_DBNAME)

disconnect:
  - db2 connect reset
  - db2 terminate

rebind: connect
  db2 bind tpccctx.bnd $(BND_OPTS)

#
#####
# Build Rules
#
#####

SUFFIXES:
.SUFFIXES: $(OBJEXT) .c .sqc

.sqc.c:
  @echo "Prepping $*.sqc"
  -db2 prep $*.sqc $(PRP_OPTS)
  @echo "Binding $*.bnd"
  db2 bind $*.bnd $(BND_OPTS)

#
#####
# Dependencies
#
#####

# Source
tpccdbg$(OBJEXT): tpccdbg.c
tpccctx$(OBJEXT): tpccctx.c
tpccmisc$(OBJEXT): tpccmisc.c

# Headers
tpccdbg.c: $(TPCC_ROOT)/include/db2tpcc.h

Src.Common/tpccctx.sqc

/*****
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****/

/*
 *
 * tpccctx.sqc - TPCC context code
 *
 */

#include <string.h>
#include <sqlutil.h>
#include "db2tpcc.h"
```

```

#include "tpccdbg.h"

int connect_to_TM(char *in_dbname);
int connect_to_TM_auth(char *in_dbname, char *in_username, char *in_password);
int disconnect_from_TM(void);

int connect_to_TM(char *in_dbname)
{
    return connect_to_TM_auth(in_dbname, "", "");
}

int connect_to_TM_auth(char *in_dbname, char *in_username, char *in_password)
{
    SQL_STRUCTURE sqlca sqlca;
    int ConnectSQLCODE = 0;

    EXEC SQL BEGIN DECLARE SECTION;
    char dbname[9];
    char username[129];
    char password[15];
    EXEC SQL END DECLARE SECTION;

    /* Copy 9 characters - 8 for dbname, 1 for NULL */
    strncpy(dbname,in_dbname,9);
    if (strcmp(in_username,"") == 0)
    {
        EXEC SQL CONNECT TO :dbname IN SHARE MODE;
    } else {
        strncpy(username,in_username,128);
        strncpy(password,in_password,14);
        EXEC SQL CONNECT TO :dbname IN SHARE MODE USER :username
        USING :password;
    }

    ConnectSQLCODE = SQLCODE;
    if (ConnectSQLCODE != 0)
    {
        sqlerror( CLIENT_SQL, "CONNECT", __FILE__, __LINE__, &sqlca);

        return ConnectSQLCODE;
    }

    return 0;
}

int disconnect_from_TM(void)
{
    SQL_STRUCTURE sqlca sqlca;
    int DisconnectSQLCODE = 0;

    EXEC SQL CONNECT RESET;

    DisconnectSQLCODE = SQLCODE;
    if (DisconnectSQLCODE != 0) {
        sqlerror( CLIENT_SQL, "DISCONNECT", __FILE__, __LINE__, &sqlca);
    }

    if (DisconnectSQLCODE) {
        return DisconnectSQLCODE;
    }
    return 0;
}

```

Src.Common/tpccdbg.c

```

/*****
** Licensed Materials - Property of IBM
**

```

```

** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****/

/*
 * tcdbg.c - Debugging Routines
 */

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

#include "sqlca.h"
#include "sql.h"
#include "db2tpcc.h"
#include "tpccdbg.h"

#define DEBUG_FILENAME_SZ 128
#define DEBUG_PATH_SIZE 128

void del_print();
void new_print();
void ord_print();
void pay_print();
void stk_print();

void current_tmstamp(char *buf);

static int debugInit = 0;
static char debugPath[DEBUG_PATH_SIZE] = "";

/*-----*/
/* InitializeDebug */
/*-----*/
__inline void InitializeDebug(void) {
    if (debugInit == 0) {
        char *p = getenv("TPCC_DEBUGDIR");
        if (p) {
            strncpy(debugPath, p, DEBUG_PATH_SIZE);
        } else {
            strcpy(debugPath, "/tmp");
        }
        strcat(debugPath, "/");
    }
    debugInit = 1;
}

/*-----*/
/* sqlerror */
/*-----*/
void sqlerror(int tranType, char *msg, char *file, int line, SQL_STRUCTURE sqlca *psqlca)
{
    FILE *err_fp = NULL;
    char err_fn[DEBUG_PATH_SIZE + DEBUG_FILENAME_SZ];
    char tranName[16];
    int j,k;
    char timeStamp[27];
    char errStr[512] = "";

    InitializeDebug();
    strncpy(err_fn, debugPath, DEBUG_PATH_SIZE);

```

```

    current_tmstamp(&timeStamp[0]);
    timeStamp[19] = (char)NULL;

    switch(tranType)
    {
        case NEWORD_SQL:
            // sprintf(err_fn, "%d.err.out", getpid());
            strcat(err_fn, "new.err.out");
            strcpy(tranName, "NEW_ORDER");
            break;

        case DELIVERY_SQL:
            // sprintf(err_fn, "%d.err.out", getpid());
            strcat(err_fn, "del.err.out");
            strcpy(tranName, "DELIVERY");
            break;

        case PAYMENT_SQL:
            // sprintf(err_fn, "%d.err.out", getpid());
            strcat(err_fn, "pay.err.out");
            strcpy(tranName, "PAYMENT");
            break;

        case ORDSTAT_SQL:
            // sprintf(err_fn, "%d.err.out", getpid());
            strcat(err_fn, "ord.err.out");
            strcpy(tranName, "ORDER_STAT");
            break;

        case STOCKLEV_SQL:
            //sprintf(err_fn, "%d.err.out", getpid());
            strcat(err_fn, "stk.err.out");
            strcpy(tranName, "STOCK_LVL");
            break;

        case 0:
            strcat(err_fn, "cli.err.out");
            strcpy(tranName, "CLIENT");
            break;

        default:
            return;
    }

    /* Generate Formatted Error Message */
    sqlaintp(errStr, 512, 78, psqlca);

    if ((err_fp = fopen(err_fn, "a+")) == NULL)
    {
        return;
    }

    fprintf(err_fp, "-----\n");
    fprintf(err_fp, "Transaction: %s (%s)\n", tranName, msg);
    fprintf(err_fp, "FILE %s (%u)\n", file, line);
    fprintf(err_fp, "SQLCODE %d", psqlca->sqlcode);
    fprintf(err_fp, "PID %d", getpid());
    fprintf(err_fp, "TIME %s\n", timeStamp);
    fprintf(err_fp, "-----\n");
    fprintf(err_fp, "%s", errStr);
    fprintf(err_fp, "-----\n");

    if (psqlca->sqlerrmc[0] != ' ' || psqlca->sqlerrmc[1] != ' ')
    {
        fprintf(err_fp, "slerrmc: ");

        for(j = 0; j < 5; j++)
        {
            for(k = 0; k < 16; k++) {

```

```

int pos = j * 16 + k;
if (pos < 70) fprintf(err_fp, "%02x ", psqlca->sqlerrmc[pos]);
else fprintf(err_fp, " ");
}
fprintf(err_fp, " |");
for(k = 0; k < 16; k++) {
int pos = j * 16 + k;
char c = ' ';
if (pos < 70) {
c = psqlca->sqlerrmc[pos];
if (!sprintf(c) c = ' ');
}
fprintf(err_fp, "%c", c);
}
fprintf(err_fp, "\n");
if (j < 4) fprintf(err_fp, " ");
}

fprintf(err_fp, "sqlerrp: ");
for(j = 0; j < 8; j++)
fprintf(err_fp, "%c", psqlca->sqlerrp[j]);
fprintf(err_fp, "\n");

fprintf(err_fp, "sqlerrd: ");
for(j = 0; j < 6; j++)
fprintf(err_fp, "%d", psqlca->sqlerrd[j]);
fprintf(err_fp, "\n");

if (psqlca->sqlwarn[0] != ' ')
{
fprintf(err_fp, "sqlwarn: ");
for(j = 0; j < 8; j++)
fprintf(err_fp, "%c ", psqlca->sqlwarn[j]);
fprintf(err_fp, "\n");
}

fprintf(err_fp, "\n");

fclose(err_fp);

/*-----*/
/* del_debug */
/*-----*/
void del_debug (struct out_delivery_struct *delivery_ptr,
struct in_delivery_struct *in_delivery,
char *msg)
{
char debug_fn[DEBUG_PATH_SIZE + DEBUG_FILENAME_SZ];

InitializeDebug();
strncpy(debug_fn, debugPath, DEBUG_PATH_SIZE);
strcat(debug_fn, "del.debug.out");
del_print(delivery_ptr, in_delivery, debug_fn, msg);
}

/*-----*/
/* del_print */
/*-----*/
void del_print (struct out_delivery_struct *delivery_ptr,
struct in_delivery_struct *in_delivery,
char *filename,
char *msg)
{
FILE *debug_fp;
char timeStamp[27];
int j;

```

```

current_tmstamp(&timeStamp[0]);
timeStamp[19] = (char)NULL;

if ((debug_fp = fopen(filename, "a+")) == NULL)
{
return;
}

fprintf(debug_fp, "Delivery debug information follows %s (%s)\n", timeStamp, msg);
fprintf(debug_fp, " PID %d ", getpid());
fprintf(debug_fp, "\n=====");

fprintf(debug_fp, "in_delivery_struct {\n");
fprintf(debug_fp, "ts_W_ID = %d (%X)\n",
in_delivery->s_W_ID, in_delivery->s_W_ID);
fprintf(debug_fp, "ts_O_CARRIER_ID = %d (%X)\n",
in_delivery->s_O_CARRIER_ID, in_delivery->s_O_CARRIER_ID);
fprintf(debug_fp, ")\n");

fprintf(debug_fp, "out_delivery_struct {\n");
fprintf(debug_fp, "ts_transtatus = %d (%X)\n",
delivery_ptr->s_transtatus, delivery_ptr->s_transtatus);
fprintf(debug_fp, "tdeadlocks = %d (%X)\n",
delivery_ptr->deadlocks, delivery_ptr->deadlocks);

for (j = 0; j < 10; j++) {
fprintf(debug_fp, "\tts_O_ID[%d] = %d\n",
j, delivery_ptr->s_O_ID[j]);
}
fprintf(debug_fp, "t)\n)\n");
fclose(debug_fp);

/*-----*/
/* new_debug */
/*-----*/
void new_debug (struct out_neword_struct *neword_ptr,
struct in_neword_struct *in_neword,
char *msg)
{
char debug_fn[DEBUG_PATH_SIZE + DEBUG_FILENAME_SZ];

InitializeDebug();
strncpy(debug_fn, debugPath, DEBUG_PATH_SIZE);
strcat(debug_fn, "new.debug.out");
new_print(neword_ptr, in_neword, debug_fn, msg);
}

/*-----*/
/* new_print */
/*-----*/
void new_print (struct out_neword_struct *neword_ptr,
struct in_neword_struct *in_neword,
char *filename,
char *msg)
{
FILE *debug_fp;
char timeStamp[27];
int j, items;

current_tmstamp(&timeStamp[0]);
timeStamp[19] = (char)NULL;

if ((debug_fp = fopen(filename, "a+")) == NULL)
{
return;
}

fprintf(debug_fp, "New order debug information follows %s (%s)\n", timeStamp, msg);

```

```

fprintf(debug_fp, " PID %d ", getpid());
fprintf(debug_fp, "\n=====");

fprintf(debug_fp, "in_neword_struct {\n");

fprintf(debug_fp, "ts_C_ID = %d (%X)\n",
in_neword->s_C_ID, in_neword->s_C_ID);
fprintf(debug_fp, "ts_W_ID = %d (%X)\n",
in_neword->s_W_ID, in_neword->s_W_ID);
fprintf(debug_fp, "ts_D_ID = %d (%X)\n",
in_neword->s_D_ID, in_neword->s_D_ID);
fprintf(debug_fp, "ts_O_OL_CNT = %d (%X)\n",
in_neword->s_O_OL_CNT, in_neword->s_O_OL_CNT);
fprintf(debug_fp, "ts_all_local = %d (%X)\n",
in_neword->s_all_local, in_neword->s_all_local);
// fprintf(debug_fp, "ts_transtatus = %d (%X)\n",
// in_neword->s_transtatus, in_neword->s_transtatus);
// fprintf(debug_fp, "tduplicate_items= %d (%X)\n",
// in_neword->duplicate_items, in_neword->duplicate_items);

fprintf(debug_fp, "titems {\n");
items = in_neword->s_O_OL_CNT;
for (j=0; j<items; j++) {
if(j != 0)
fprintf(debug_fp, "\n");
fprintf(debug_fp, "\tts_OL_I_ID[%d] = %d (%X)\n",
j, in_neword->in_item[j].s_OL_I_ID, in_neword->in_item[j].s_OL_I_ID);
fprintf(debug_fp, "\tts_OL_SUPPLY_W_ID[%d] = %d (%X)\n",
j, in_neword->in_item[j].s_OL_SUPPLY_W_ID, in_neword->in_item[j].s_OL_SUPPLY_W_ID);
fprintf(debug_fp, "\tts_OL_QUANTITY[%d] = %d (%X)\n",
j, in_neword->in_item[j].s_OL_QUANTITY, in_neword->in_item[j].s_OL_QUANTITY);
}
fprintf(debug_fp, "t)\n)\n");

fprintf(debug_fp, "out_neword_struct {\n");
fprintf(debug_fp, "ts_C_LAST = %s\n",
neword_ptr->s_C_LAST);
fprintf(debug_fp, "ts_C_CREDIT = %s\n",
neword_ptr->s_C_CREDIT);
fprintf(debug_fp, "ts_W_TAX = %04.4f\n",
neword_ptr->s_W_TAX);
fprintf(debug_fp, "ts_D_TAX = %04.4f\n",
neword_ptr->s_D_TAX);
fprintf(debug_fp, "ts_C_DISCOUNT = %04.4f\n",
neword_ptr->s_C_DISCOUNT);
fprintf(debug_fp, "ts_O_ID = %d (%X)\n",
neword_ptr->s_O_ID, neword_ptr->s_O_ID);
fprintf(debug_fp, "ts_O_OL_CNT = %d (%X)\n",
neword_ptr->s_O_OL_CNT, neword_ptr->s_O_OL_CNT);
fprintf(debug_fp, "ts_O_ENTRY_D = %s\n",
neword_ptr->s_O_ENTRY_D.time);
fprintf(debug_fp, "ts_total_amount = %2f\n",
neword_ptr->s_total_amount);
fprintf(debug_fp, "ts_transtatus = %d (%X)\n",
neword_ptr->s_transtatus, neword_ptr->s_transtatus);
fprintf(debug_fp, "tdeadlocks = %d (%X)\n",
neword_ptr->deadlocks, neword_ptr->deadlocks);

// fprintf(debug_fp, "ts_W_ID = %d (%X)\n",
// neword_ptr->s_W_ID, neword_ptr->s_W_ID);
// fprintf(debug_fp, "ts_D_ID = %d (%X)\n",
// neword_ptr->s_D_ID, neword_ptr->s_D_ID);
// fprintf(debug_fp, "ts_all_local = %d (%X)\n",
// neword_ptr->s_all_local, neword_ptr->s_all_local);
// fprintf(debug_fp, "tduplicate_items= %d (%X)\n",
// neword_ptr->duplicate_items, neword_ptr->duplicate_items);

fprintf(debug_fp, "titems {\n");

```



```

fprintf(debug_fp,"ts_C_STREET_2 = %s\n",
        payment_ptr->s_C_STREET_2);
fprintf(debug_fp,"ts_C_CITY = %s\n",
        payment_ptr->s_C_CITY);
fprintf(debug_fp,"ts_C_STATE = %s\n",
        payment_ptr->s_C_STATE);
fprintf(debug_fp,"ts_C_ZIP = %s\n",
        payment_ptr->s_C_ZIP);
fprintf(debug_fp,"ts_C_PHONE = %s\n",
        payment_ptr->s_C_PHONE);
fprintf(debug_fp,"ts_C_SINCE = %s\n",
        payment_ptr->s_C_SINCE_time);
fprintf(debug_fp,"ts_C_CREDIT = %s\n",
        payment_ptr->s_C_CREDIT);
fprintf(debug_fp,"ts_C_DATA = %s\n",
        payment_ptr->s_C_DATA);
fprintf(debug_fp,"ts_transtatus = %d (%X)\n",
        payment_ptr->s_transtatus,payment_ptr->s_transtatus);
fprintf(debug_fp,"tdeadlocks = %d (%X)\n",
        payment_ptr->deadlocks,payment_ptr->deadlocks);
fclose(debug_fp);
}

```

```

/*-----*/
/* stk_debug */
/*-----*/
void stk_debug (struct out_stocklev_struct *stocklev,
               struct in_stocklev_struct *in_stocklev,
               char *msg)
{
    char debug_fn[DEBUG_PATH_SIZE + DEBUG_FILENAME_SZ];

```

```

    InitializeDebug();
    strncpy(debug_fn, debugPath, DEBUG_PATH_SIZE);
    strcat(debug_fn, "stk.debug.out");
    stk_print(stocklev, in_stocklev, debug_fn, msg);
}

```

```

/*-----*/
/* stk_print */
/*-----*/
void stk_print (struct out_stocklev_struct *stocklev,
               struct in_stocklev_struct *in_stocklev,
               char *filename,
               char *msg)
{

```

```

    FILE *debug_fp;
    char timeStamp[27];

    current_tmstamp(&timeStamp[0]);
    timeStamp[19] = (char)NULL;

    if ((debug_fp = fopen(filename, "a+")) == NULL)
    {
        return;
    }

```

```

    fprintf(debug_fp,"Stock level debug information follows %s (%s)\n", timeStamp, msg);
    fprintf(debug_fp, " PID %d ", getpid());
    fprintf(debug_fp,"n=====n");

```

```

    fprintf(debug_fp,"in_stocklev_struct {n");
    fprintf(debug_fp,"ts_W_ID = %d (%X)\n",
            in_stocklev->s_W_ID, in_stocklev->s_W_ID);
    fprintf(debug_fp,"ts_D_ID = %d (%X)\n",
            in_stocklev->s_D_ID, in_stocklev->s_D_ID);
    fprintf(debug_fp,"ts_threshold = %d (%X)\n",
            in_stocklev->s_threshold, in_stocklev->s_threshold);

```

```

fprintf(debug_fp,")\n");

```

```

fprintf(debug_fp,"out_stocklev_struct {n");
fprintf(debug_fp,"ts_transtatus = %d (%X)\n",
        stocklev->s_transtatus, stocklev->s_transtatus);
fprintf(debug_fp,"tdeadlocks = %d (%X)\n",
        stocklev->deadlocks, stocklev->deadlocks);
fprintf(debug_fp,"ts_low_stock = %d (%X)\n",
        stocklev->s_low_stock, stocklev->s_low_stock);
fprintf(debug_fp,")\n");
fclose(debug_fp);
}

```

```

void current_tmstamp(char *buf)
{

```

```

    time_t t = time(NULL);
    strncpy(buf,ctime(&t),19);
}

```

Src.Common/tpccmisc.c

```

/*-----*/
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
/*-----*/

```

```

/*
 *
 * tpccmisc.c - Miscellaneous routines
 *
 */

```

```

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

```

```

double current_time_ms(void);
double current_time(void);

```

```

/* Current time in SECONDS, precision SECONDS */
double current_time(void)
{

```

```

    /* use time() to get seconds */
    return(time(NULL));
}

```

```

/* Current time in SECONDS, precision MILLISECONDS */
double current_time_ms(void)
{

```

```

    /* gettimeofday() returns seconds and microseconds */
    /* convert to fractional seconds */
    struct timeval t;
    gettimeofday(&t,NULL);
    return (t.tv_sec + (double)t.tv_usec/(1000*1000));
}

```

Src.Srv/Makefile

```

#####
#####
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use, duplication or
## disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#####
#####
#
# Makefile - Makefile for Src.Srv
#
#
include $(TPCC_ROOT)/Makefile.config

#
#####
# Preprocessor, Compiler and Linker Flags
#
#####
BND_OPTS = GRANT PUBLIC \
            MESSAGES $.bnd.msg
PRP_OPTS = BINDFILE \
            EXPLAIN ALL \
            MESSAGES $.prep.msg

INCLUDE = -I$(TPCC_SQLLIB)/include -I$(TPCC_ROOT)/include

CFLAGS = $(CFLAGS_OS) $(INCLUDE) $(CFLAGS_DEBUG) \
         -D$(DB2EDITION) -D$(DB2VERSION) \
         -DSQLA_NOLINES -DLINT_ARGS

LDLFLAGS = $(LDLFLAGS_STORP) $(LDLFLAGS_LIB)

#
#####
# File Collections
#
#####
STORED_PROCS = new ord del

UTIL_OBJ = $(TPCC_ROOT)/Src.Common/tpccmisc$(OBJEXT) \
           $(TPCC_ROOT)/Src.Common/tpccdbg$(OBJEXT)

EXE = news ords dels

#
#####
# User Targets
#
#####
all: connect explain catalog $(EXE) install plan disconnect

clean: connect uncatalog unexplain disconnect
      - $(ERASE) $(TPCC_SPDIR)$(SLASH)news
      - $(ERASE) $(TPCC_SPDIR)$(SLASH)ords
      - $(ERASE) $(TPCC_SPDIR)$(SLASH)dels
      - $(ERASE) *.bnd *.msg *.out *$(OBJEXT) $(EXE) tpcc_all_sql.c
      - $(ERASE) TPCC_ALL.*.plan

```

```

#
#####
# Helper Targets
#
#####
catalog:uncatalog
- perl $(TPCC_ROOT)$(SLASH)utils$(SLASH)genproc.pl $(STORED_PROCS)
- db2 -tvf cat-proc.ddl +o -z cat-proc.out
- db2 -td% -vf cat-func.ddl +o -z cat-func.out

uncatalog:
- perl $(TPCC_ROOT)$(SLASH)utils$(SLASH)genproc.pl $(STORED_PROCS)
- db2 -td% -vf uncat-func.ddl +o -z uncat-func.out
- db2 -tvf uncat-proc.ddl +o -z uncat-proc.out

explain:
- perl $(TPCC_ROOT)$(SLASH)utils$(SLASH)fixup_explain.pl
- db2 -tvf $(TPCC_ROOT)$(SLASH)utils$(SLASH)EXPLAIN.DDL +o -z EXPLAIN.out

unexplain:
- db2 -tvf $(TPCC_ROOT)$(SLASH)utils$(SLASH)UNEXPLAIN.DDL +o -z
UNEXPLAIN.out

connect:
- db2 connect to $(TPCC_DBNAME)

disconnect:
- db2 connect reset
- db2 terminate

plan:
- db2exfmt -d $(TPCC_DBNAME) -e $(TPCC_SCHEMA) -s $(TPCC_SCHEMA) -w -1 -n
TPCC_ALL -g -# 0 -o TPCC_ALL.exfmt.plan
- (export DB2EXPLN_BUFFER=3000000; db2expln -d $(TPCC_DBNAME) -c
$(TPCC_SCHEMA) -p TPCC_ALL -s 0 -g -o TPCC_ALL.expln.plan )

rebind: connect catalog
db2 bind tpcc_all_sql.bnd $(BND_OPTS) QUERYOPT 7

#
#####
# Install Targets
#
#####

install: $(EXE)
- mkdir $(TPCC_SPDIR)
$(COPY) ords $(TPCC_SPDIR)
$(COPY) news $(TPCC_SPDIR)
$(COPY) dels $(TPCC_SPDIR)

#
#####
# Build Rules
#
#####

.SUFFIXES: $(OBJEXT) .c .sqc

# d230437mte: QUERYOPT 7 required for UNION ALL
# Only stock needs CS , and that can be specified on the SELECT statement
tpcc_all_sql.c:
@echo "Prepping $*.sqc"
-db2 prep $*.sqc $(PRP_OPTS) ISOLATION RR
@echo "Binding $*.bnd"
db2 bind $*.bnd $(BND_OPTS) QUERYOPT 7

# Stored procedures are built in a special way

```

```

tpcc_all_sql$(OBJEXT):
$(CC) -c tpcc_all_sql.c $(CFLAGS) -D$(TPCC_SPTYPE) $(CFLAGS_OUT)@$

$(EXE): $(UTIL_OBJ) tpcc_all_sql.o
$(LD_STORP) $(LDLFLAGS) $(UTIL_OBJ) tpcc_all_sql.o $(LDFLAGS_OUT)@$

#
#####
# Dependencies
#
#####

# Executables (Stored Procedures)
$(EXE): $(UTIL_OBJ) tpcc_all_sql.o

# Source
tpcc_all_sql$(OBJEXT): tpcc_all_sql.c

# Headers
tpcc_all_sql.c: $(TPCC_ROOT)/include/db2tpcc.h

Src.Srv/cat-func.ddl
-----
-- Licensed Materials - Property of IBM
--
-- Governed under the terms of the International
-- License Agreement for Non-Warranted Sample Code.
--
-- (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
-- All Rights Reserved.
--
-- US Government Users Restricted Rights - Use, duplication or
-- disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
-----
-- cat-func.ddl - Create table functions
--
--
--
-- DELIVERY
--
CREATE FUNCTION DEL( W_ID INTEGER
, D_ID SMALLINT
, CARRIER_ID SMALLINT
)
RETURNS TABLE ( O_ID INTEGER )

SPECIFIC DELIVERY

MODIFIES SQL DATA DETERMINISTIC NO EXTERNAL ACTION LANGUAGE SQL

VAR: BEGIN ATOMIC

DECLARE O_ID INTEGER ;
DECLARE C_ID INTEGER ;
DECLARE AMOUNT DECIMAL(12,2) ;

/* Delete the order from new order table */

SET VAR.O_ID = ( SELECT NO_O_ID

FROM OLD TABLE ( DELETE

FROM ( SELECT NO_O_ID

```

```

FROM NEW_ORDER

WHERE NO_W_ID = DEL.W_ID
AND NO_D_ID = DEL.D_ID

ORDER BY NO_O_ID ASC

FETCH FIRST 1 ROW ONLY
) AS NEW_ORDER
) AS D
)
;

/* Update the order as delivered and retrieve the customer id */

SET VAR.C_ID = ( SELECT O_C_ID

FROM OLD TABLE ( UPDATE ORDERS

SET O_CARRIER_ID = DEL.CARRIER_ID

WHERE O_W_ID = DEL.W_ID
AND O_D_ID = DEL.D_ID
AND O_ID = VAR.O_ID
) AS U
)
;

SET VAR.AMOUNT = ( SELECT SUM( OL_AMOUNT )

FROM OLD TABLE ( UPDATE ORDER_LINE

SET OL_DELIVERY_D = CURRENT_TIMESTAMP

WHERE OL_W_ID = DEL.W_ID
AND OL_D_ID = DEL.D_ID
AND OL_O_ID = VAR.O_ID
) AS U
)
;

/* Charge the customer */

UPDATE CUSTOMER

SET C_BALANCE = C_BALANCE + VAR.AMOUNT
, C_DELIVERY_CNT = C_DELIVERY_CNT + SMALLINT( 1 )

WHERE C_W_ID = DEL.W_ID
AND C_D_ID = DEL.D_ID
AND C_ID = VAR.C_ID
;

/* Return the order id to the caller (or NULL) */

RETURN VALUES VAR.O_ID ;

END
%

--
-- ORDER STATUS
--

CREATE FUNCTION ORD_C_LAST( W_ID INTEGER
, D_ID SMALLINT
, C_LAST VARCHAR(16)
)

```

```

RETURNS TABLE( O_ID INTEGER
, O_CARRIER_ID SMALLINT
, O_ENTRY_D TIMESTAMP
, C_BALANCE DECIMAL(12,2)
, C_FIRST VARCHAR(16)
, C_MIDDLE CHAR(2)
, C_ID INTEGER
)

SPECIFIC ORD_C_LAST

READS SQL DATA NO EXTERNAL ACTION DETERMINISTIC LANGUAGE SQL

VAR: BEGIN ATOMIC

DECLARE C_BALANCE DECIMAL(12,2);
DECLARE C_FIRST VARCHAR(16);
DECLARE C_MIDDLE CHAR(2);
DECLARE C_ID INTEGER;
DECLARE O_ID INTEGER;
DECLARE O_CARRIER_ID SMALLINT;
DECLARE O_ENTRY_D TIMESTAMP;

/* Retrieve the Customer information */

SET ( C_BALANCE, C_FIRST, C_MIDDLE, C_ID )

= ( SELECT C_BALANCE, C_FIRST, C_MIDDLE, C_ID

FROM ( SELECT C_ID
, C_BALANCE
, C_FIRST
, C_MIDDLE
, COUNT(*) OVER() AS COUNT
, ROWNUMBER() OVER (ORDER BY C_FIRST) AS NUM

FROM CUSTOMER
WHERE C_W_ID = ORD_C_LAST.W_ID
AND C_D_ID = ORD_C_LAST.D_ID
AND C_LAST = ORD_C_LAST.C_LAST

) AS V1

WHERE NUM = (COUNT + BIGINT(1)) / BIGINT(2)
)
;

SET ( O_ID, O_CARRIER_ID, O_ENTRY_D )
= ( SELECT O_ID
, O_CARRIER_ID
, O_ENTRY_D

FROM ORDERS

WHERE O_W_ID = ORD_C_LAST.W_ID
AND O_D_ID = ORD_C_LAST.D_ID
AND O_C_ID = VAR.C_ID

ORDER BY O_ID DESC
FETCH FIRST 1 ROW ONLY

)
;

RETURN VALUES ( VAR.O_ID
, VAR.O_CARRIER_ID
, VAR.O_ENTRY_D
, VAR.C_BALANCE
, VAR.C_FIRST

```

```

, VAR.C_MIDDLE
, VAR.C_ID
)
;

END
%

CREATE FUNCTION ORD_C_ID( W_ID INTEGER
, D_ID SMALLINT
, C_ID INTEGER
)

RETURNS TABLE( O_ID INTEGER
, O_CARRIER_ID SMALLINT
, O_ENTRY_D TIMESTAMP
, C_BALANCE DECIMAL(12,2)
, C_FIRST VARCHAR(16)
, C_MIDDLE CHAR(2)
, C_LAST VARCHAR(16)
)

SPECIFIC ORD_C_ID

READS SQL DATA NO EXTERNAL ACTION DETERMINISTIC LANGUAGE SQL

VAR: BEGIN ATOMIC

DECLARE C_BALANCE DECIMAL(12,2);
DECLARE C_FIRST VARCHAR(16);
DECLARE C_MIDDLE CHAR(2);
DECLARE C_LAST VARCHAR(16);
DECLARE O_ID INTEGER;
DECLARE O_CARRIER_ID SMALLINT;
DECLARE O_ENTRY_D TIMESTAMP;

/* Retrieve the Customer information */

SET ( C_BALANCE, C_FIRST, C_MIDDLE, C_LAST )

= ( SELECT C_BALANCE, C_FIRST, C_MIDDLE, C_LAST

FROM CUSTOMER

WHERE C_ID = ORD_C_ID.C_ID
AND C_W_ID = ORD_C_ID.W_ID
AND C_D_ID = ORD_C_ID.D_ID

)
;

SET ( O_ID, O_CARRIER_ID, O_ENTRY_D )

= ( SELECT O_ID
, O_CARRIER_ID
, O_ENTRY_D

FROM ORDERS

WHERE O_W_ID = ORD_C_ID.W_ID
AND O_D_ID = ORD_C_ID.D_ID
AND O_C_ID = ORD_C_ID.C_ID

ORDER BY O_ID DESC
FETCH FIRST 1 ROW ONLY

)
;

RETURN VALUES ( VAR.O_ID
, VAR.O_CARRIER_ID

```

```

, VAR.O_ENTRY_D
, VAR.C_BALANCE
, VAR.C_FIRST
, VAR.C_MIDDLE
, VAR.C_LAST
);

END
%

--
-- PAYMENT
--

CREATE FUNCTION PAY_C_LAST( W_ID INTEGER
, D_ID SMALLINT
, C_W_ID INTEGER
, C_D_ID SMALLINT
, C_LAST VARCHAR(16)
, H_AMOUNT DECIMAL(6,2)
, BAD_CREDIT_PREFIX VARCHAR(28)
)

RETURNS TABLE( W_STREET_1 CHAR(20)
, W_STREET_2 CHAR(20)
, W_CITY CHAR(20)
, W_STATE CHAR(2)
, W_ZIP CHAR(9)
, D_STREET_1 CHAR(20)
, D_STREET_2 CHAR(20)
, D_CITY CHAR(20)
, D_STATE CHAR(2)
, D_ZIP CHAR(9)
, C_ID INTEGER
, C_FIRST VARCHAR(16)
, C_MIDDLE CHAR(2)
, C_STREET_1 VARCHAR(20)
, C_STREET_2 VARCHAR(20)
, C_CITY VARCHAR(20)
, C_STATE CHAR(2)
, C_ZIP CHAR(9)
, C_PHONE CHAR(16)
, C_SINCE TIMESTAMP
, C_CREDIT CHAR(2)
, C_CREDIT_LIM DECIMAL(12,2)
, C_DISCOUNT INTEGER
, C_BALANCE DECIMAL(12,2)
, C_DATA CHAR(200)
, H_DATE TIMESTAMP
)

SPECIFIC PAY_C_LAST

MODIFIES SQL DATA DETERMINISTIC NO EXTERNAL ACTION LANGUAGE SQL

VAR: BEGIN ATOMIC

DECLARE W_NAME CHAR(10);
DECLARE D_NAME CHAR(10);

DECLARE W_STREET_1 CHAR(20);
DECLARE W_STREET_2 CHAR(20);
DECLARE W_CITY CHAR(20);
DECLARE W_STATE CHAR(2);
DECLARE W_ZIP CHAR(9);

DECLARE D_STREET_1 CHAR(20);
DECLARE D_STREET_2 CHAR(20);

```

```

DECLARE D_CITY CHAR(20);
DECLARE D_STATE CHAR(2);
DECLARE D_ZIP CHAR(9);

DECLARE C_ID INTEGER;

DECLARE C_FIRST VARCHAR(16);
DECLARE C_MIDDLE CHAR(2);
DECLARE C_STREET_1 VARCHAR(20);
DECLARE C_STREET_2 VARCHAR(20);
DECLARE C_CITY VARCHAR(20);
DECLARE C_STATE CHAR(2);
DECLARE C_ZIP CHAR(9);
DECLARE C_PHONE CHAR(16);
DECLARE C_SINCE TIMESTAMP;
DECLARE C_CREDIT CHAR(2);
DECLARE C_CREDIT_LIM DECIMAL(12,2);
DECLARE C_DISCOUNT INTEGER;
DECLARE C_BALANCE DECIMAL(12,2);
DECLARE C_DATA CHAR(200);

DECLARE H_DATE TIMESTAMP;

/* Generate the current date and time for the payment date */
SET H_DATE = CURRENT TIMESTAMP;

/* Update District and retrieve its data */

SET ( D_NAME, D_STREET_1, D_STREET_2, D_CITY, D_STATE, D_ZIP )
= ( SELECT D_NAME, D_STREET_1, D_STREET_2, D_CITY, D_STATE, D_ZIP
    FROM OLD TABLE ( UPDATE DISTRICT
        SET D_YTD = D_YTD + PAY_C_LAST.H_AMOUNT
        WHERE D_W_ID = PAY_C_LAST.W_ID
        AND D_ID = PAY_C_LAST.D_ID
    ) AS U
);

/* Determine the C_ID */

SET ( C_ID )
= ( SELECT C_ID
    FROM ( SELECT C_ID
        , COUNT(*) OVER() AS COUNT
        , ROWNUMBER() OVER (ORDER BY C_FIRST) AS NUM
        FROM CUSTOMER
        WHERE C_LAST = PAY_C_LAST.C_LAST
        AND C_W_ID = PAY_C_LAST.C_W_ID
        AND C_D_ID = PAY_C_LAST.C_D_ID
    ) AS T
    WHERE NUM = (COUNT + BIGINT(1)) / BIGINT(2)
);

/* Update the middle customer */

SET ( C_ID, C_FIRST, C_MIDDLE, C_STREET_1, C_STREET_2
    , C_CITY, C_STATE, C_ZIP, C_PHONE, C_SINCE, C_CREDIT, C_CREDIT_LIM
    , C_DISCOUNT, C_BALANCE, C_DATA )
= ( SELECT C_ID, C_FIRST, C_MIDDLE, C_STREET_1, C_STREET_2
    , C_CITY, C_STATE, C_ZIP, C_PHONE, C_SINCE, C_CREDIT, C_CREDIT_LIM

```

```

    , C_DISCOUNT, C_BALANCE
    , CASE WHEN C_CREDIT = 'BC' THEN SUBSTR(C_DATA, 1, 200) ELSE NULL
END AS C_DATA
FROM NEW TABLE ( UPDATE CUSTOMER
    SET C_BALANCE = C_BALANCE - PAY_C_LAST.H_AMOUNT
    , C_YTD_PAYMENT = C_YTD_PAYMENT +
PAY_C_LAST.H_AMOUNT
    , C_PAYMENT_CNT = C_PAYMENT_CNT + SMALLINT(1)
    , C_DATA = CASE WHEN C_CREDIT = 'BC'
        THEN CHAR(C_ID) -- 11 bytes long
        || BAD_CREDIT_PREFIX -- 28 bytes long
        || SUBSTR(C_DATA, 1, 461) -- 461 + 39 = 500
        ELSE C_DATA
    END
    WHERE C_W_ID = PAY_C_LAST.C_W_ID
    AND C_D_ID = PAY_C_LAST.C_D_ID
    AND C_ID = VAR.C_ID
) AS U
);

/* Update the warehouse */

SET ( W_NAME, W_STREET_1, W_STREET_2, W_CITY, W_STATE, W_ZIP )
= ( SELECT W_NAME, W_STREET_1, W_STREET_2, W_CITY, W_STATE, W_ZIP
    FROM OLD TABLE ( UPDATE WAREHOUSE
        SET W_YTD = W_YTD + PAY_C_LAST.H_AMOUNT
        WHERE W_ID = PAY_C_LAST.W_ID
    ) AS U
);

/* Finally insert into the warehouse */

INSERT
    INTO HISTORY ( H_C_ID, H_C_D_ID, H_C_W_ID, H_D_ID, H_W_ID, H_DATA,
H_DATE, H_AMOUNT )
VALUES ( VAR.C_ID
    , PAY_C_LAST.C_D_ID
    , PAY_C_LAST.C_W_ID
    , PAY_C_LAST.D_ID
    , PAY_C_LAST.W_ID
    , VAR.W_NAME || CHAR(' ', 4) || VAR.D_NAME
    , VAR.H_DATE
    , PAY_C_LAST.H_AMOUNT
);

/* Done - return the collected data */

RETURN VALUES ( W_STREET_1, W_STREET_2, W_CITY, W_STATE, W_ZIP
    , D_STREET_1, D_STREET_2, D_CITY, D_STATE, D_ZIP
    , C_ID, C_FIRST, C_MIDDLE, C_STREET_1, C_STREET_2
    , C_CITY, C_STATE, C_ZIP, C_PHONE, C_SINCE, C_CREDIT,
C_CREDIT_LIM
    , C_DISCOUNT, C_BALANCE, C_DATA, H_DATE
);

END

```

```

%
CREATE FUNCTION PAY_C_ID( W_ID INTEGER
    , D_ID SMALLINT
    , C_W_ID INTEGER
    , C_D_ID SMALLINT
    , C_ID INTEGER
    , H_AMOUNT DECIMAL(6,2)
    , BAD_CREDIT_PREFIX VARCHAR(34)
)
RETURNS TABLE( W_STREET_1 CHAR(20)
    , W_STREET_2 CHAR(20)
    , W_CITY CHAR(20)
    , W_STATE CHAR(2)
    , W_ZIP CHAR(9)
    , D_STREET_1 CHAR(20)
    , D_STREET_2 CHAR(20)
    , D_CITY CHAR(20)
    , D_STATE CHAR(2)
    , D_ZIP CHAR(9)
    , C_LAST VARCHAR(16)
    , C_FIRST VARCHAR(16)
    , C_MIDDLE CHAR(2)
    , C_STREET_1 VARCHAR(20)
    , C_STREET_2 VARCHAR(20)
    , C_CITY VARCHAR(20)
    , C_STATE CHAR(2)
    , C_ZIP CHAR(9)
    , C_PHONE CHAR(16)
    , C_SINCE TIMESTAMP
    , C_CREDIT CHAR(2)
    , C_CREDIT_LIM DECIMAL(12,2)
    , C_DISCOUNT REAL
    , C_BALANCE DECIMAL(12,2)
    , C_DATA CHAR(200)
    , H_DATE TIMESTAMP
)
SPECIFIC PAY_C_ID
MODIFIES SQL DATA DETERMINISTIC NO EXTERNAL ACTION LANGUAGE SQL

VAR: BEGIN ATOMIC

DECLARE W_NAME CHAR(10);
DECLARE D_NAME CHAR(10);

DECLARE W_STREET_1 CHAR(20);
DECLARE W_STREET_2 CHAR(20);
DECLARE W_CITY CHAR(20);
DECLARE W_STATE CHAR(2);
DECLARE W_ZIP CHAR(9);

DECLARE D_STREET_1 CHAR(20);
DECLARE D_STREET_2 CHAR(20);
DECLARE D_CITY CHAR(20);
DECLARE D_STATE CHAR(2);
DECLARE D_ZIP CHAR(9);

DECLARE C_LAST VARCHAR(16);

DECLARE C_FIRST VARCHAR(16);
DECLARE C_MIDDLE CHAR(2);
DECLARE C_STREET_1 VARCHAR(20);
DECLARE C_STREET_2 VARCHAR(20);
DECLARE C_CITY VARCHAR(20);
DECLARE C_STATE CHAR(2);
DECLARE C_ZIP CHAR(9);
DECLARE C_PHONE CHAR(16);
DECLARE C_SINCE TIMESTAMP;

```

```

DECLARE C_CREDIT CHAR(2);
DECLARE C_CREDIT_LIM DECIMAL(12,2);
DECLARE C_DISCOUNT REAL;
DECLARE C_BALANCE DECIMAL(12,2);
DECLARE C_DATA CHAR(200);
DECLARE H_DATE TIMESTAMP;

/* Generate the current date and time for the payment date */
SET H_DATE = CURRENT_TIMESTAMP;

/* Update District and retrieve its data */

SET ( D_NAME, D_STREET_1, D_STREET_2, D_CITY, D_STATE, D_ZIP )
= ( SELECT D_NAME, D_STREET_1, D_STREET_2, D_CITY, D_STATE, D_ZIP
    FROM OLD TABLE ( UPDATE DISTRICT
        SET D_YTD = D_YTD + PAY_C_ID.H_AMOUNT
        WHERE D_W_ID = PAY_C_ID.W_ID
        AND D_ID = PAY_C_ID.D_ID
    ) AS U
);

/* Update the middle customer */

SET ( C_LAST, C_FIRST, C_MIDDLE, C_STREET_1, C_STREET_2
    , C_CITY, C_STATE, C_ZIP, C_PHONE, C_SINCE, C_CREDIT, C_CREDIT_LIM
    , C_DISCOUNT, C_BALANCE, C_DATA )
= ( SELECT C_LAST, C_FIRST, C_MIDDLE, C_STREET_1, C_STREET_2
    , C_CITY, C_STATE, C_ZIP, C_PHONE, C_SINCE, C_CREDIT, C_CREDIT_LIM
    , C_DISCOUNT, C_BALANCE
    , CASE WHEN C_CREDIT = 'BC' THEN SUBSTR(C_DATA, 1, 200) ELSE NULL
    END AS C_DATA
    FROM NEW TABLE ( UPDATE CUSTOMER
        SET C_BALANCE = C_BALANCE - PAY_C_ID.H_AMOUNT
        , C_YTD_PAYMENT = C_YTD_PAYMENT + PAY_C_ID.H_AMOUNT
        , C_PAYMENT_CNT = C_PAYMENT_CNT + SMALLINT( 1 )
        , C_DATA = CASE WHEN C_CREDIT = 'BC'
            THEN BAD_CREDIT_PREFIX -- 34 bytes long
            || SUBSTR( C_DATA, 1, 466 ) -- 466 + 34 = 500 bytes
            ELSE C_DATA
        END
        WHERE C_W_ID = PAY_C_ID.C_W_ID
        AND C_D_ID = PAY_C_ID.C_D_ID
        AND C_ID = PAY_C_ID.C_ID
    ) AS U
);

/* Update the warehouse */

SET ( W_NAME, W_STREET_1, W_STREET_2, W_CITY, W_STATE, W_ZIP )
= ( SELECT W_NAME, W_STREET_1, W_STREET_2, W_CITY, W_STATE, W_ZIP
    FROM OLD TABLE ( UPDATE WAREHOUSE
        SET W_YTD = W_YTD + PAY_C_ID.H_AMOUNT
        WHERE W_ID = PAY_C_ID.W_ID
    ) AS U
);

```

```

;
/* Finally insert into the warehouse */
INSERT
    INTO HISTORY ( H_C_ID, H_C_D_ID, H_C_W_ID, H_D_ID, H_W_ID, H_DATA,
    H_DATE, H_AMOUNT )
VALUES ( PAY_C_ID.C_ID
    , PAY_C_ID.C_D_ID
    , PAY_C_ID.C_W_ID
    , PAY_C_ID.D_ID
    , PAY_C_ID.W_ID
    , VAR.W_NAME || CHAR( ' ', 4 ) || VAR.D_NAME
    , VAR.H_DATE
    , PAY_C_ID.H_AMOUNT
);

/* Done - return the collected data */

RETURN VALUES ( W_STREET_1, W_STREET_2, W_CITY, W_STATE, W_ZIP
    , D_STREET_1, D_STREET_2, D_CITY, D_STATE, D_ZIP
    , C_LAST, C_FIRST, C_MIDDLE, C_STREET_1, C_STREET_2
    , C_CITY, C_STATE, C_ZIP, C_PHONE, C_SINCE, C_CREDIT,
    C_CREDIT_LIM
    , C_DISCOUNT, C_BALANCE, C_DATA, H_DATE
);

END
%
--
-- NEW ORDER
--

CREATE FUNCTION NEW_OL_ALL( I_ID INT
    , I_QTY SMALLINT
    , W_ID INT
    , SUPP_W_ID INT
    , O_ID INT
    , D_ID SMALLINT
)
RETURNS TABLE( I_PRICE DECIMAL(5,2)
    , I_NAME CHAR(24)
    , I_DATA VARCHAR(50)
    , OL_DIST_INFO CHAR(24)
    , S_DATA VARCHAR(50)
    , S_QUANTITY SMALLINT
)

SPECIFIC NEW_OL_ALL
MODIFIES SQL DATA DETERMINISTIC NO EXTERNAL ACTION LANGUAGE SQL

VAR: BEGIN ATOMIC

DECLARE I_PRICE DECIMAL(5,2);
DECLARE I_NAME CHAR(24);
DECLARE I_DATA VARCHAR(50);
DECLARE OL_DIST_INFO CHAR(24);
DECLARE S_DATA VARCHAR(50);
DECLARE S_QUANTITY SMALLINT;

SET ( I_PRICE, I_NAME, I_DATA )

```

```

= ( SELECT
    I_PRICE
    , I_NAME
    , I_DATA
    FROM ITEM
    WHERE ITEM.I_ID = NEW_OL_ALL.I_ID
);

SET ( OL_DIST_INFO, S_DATA, S_QUANTITY )
= ( SELECT OL_DIST_INFO
    , S_DATA
    , S_QUANTITY
    FROM NEW TABLE ( UPDATE STOCK
        INCLUDE ( OL_DIST_INFO CHAR( 24 ) )
        SET S_QUANTITY = CASE WHEN S_QUANTITY -
NEW_OL_ALL.I_QTY >= 10
            THEN S_QUANTITY - NEW_OL_ALL.I_QTY
            ELSE S_QUANTITY - NEW_OL_ALL.I_QTY + 91
        END
        , S_ORDER_CNT = S_ORDER_CNT + SMALLINT( 1 )
        , S_YTD = S_YTD + NEW_OL_ALL.I_QTY
        , S_REMOTE_CNT = CASE WHEN
NEW_OL_ALL.SUPP_W_ID = NEW_OL_ALL.W_ID
            THEN S_REMOTE_CNT
            ELSE S_REMOTE_CNT + SMALLINT( 1 )
        END
        , OL_DIST_INFO = CASE D_ID WHEN SMALLINT( 1 )
        THEN S_DIST_01
        WHEN SMALLINT( 2 ) THEN
        S_DIST_02
        WHEN SMALLINT( 3 ) THEN
        S_DIST_03
        WHEN SMALLINT( 4 ) THEN
        S_DIST_04
        WHEN SMALLINT( 5 ) THEN
        S_DIST_05
        WHEN SMALLINT( 6 ) THEN
        S_DIST_06
        WHEN SMALLINT( 7 ) THEN
        S_DIST_07
        WHEN SMALLINT( 8 ) THEN
        S_DIST_08
        WHEN SMALLINT( 9 ) THEN
        S_DIST_09
        WHEN SMALLINT( 10 ) THEN
        S_DIST_10
        END
        WHERE S_I_ID = NEW_OL_ALL.I_ID
        AND S_W_ID = NEW_OL_ALL.SUPP_W_ID
    ) AS U
);

RETURN VALUES( VAR.I_PRICE
    , VAR.I_NAME
    , VAR.I_DATA
    , VAR.OL_DIST_INFO
    , VAR.S_DATA
    , VAR.S_QUANTITY
);

```

```

;
END
%
CREATE FUNCTION NEW_OL_LOCAL( I_ID INT
, I_QTY SMALLINT
, W_ID INT
, O_ID INT
, D_ID SMALLINT
)
RETURNS TABLE( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, OL_DIST_INFO CHAR(24)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT
)
SPECIFIC NEW_OL_LOCAL
MODIFIES SQL DATA DETERMINISTIC NO EXTERNAL ACTION LANGUAGE SQL
VAR: BEGIN ATOMIC
DECLARE I_PRICE DECIMAL(5,2) ;
DECLARE I_NAME CHAR(24) ;
DECLARE I_DATA VARCHAR(50) ;
DECLARE OL_DIST_INFO CHAR(24) ;
DECLARE S_DATA VARCHAR(50) ;
DECLARE S_QUANTITY SMALLINT ;
SET ( I_PRICE , I_NAME , I_DATA )
= ( SELECT
I_PRICE
, I_NAME
, I_DATA
FROM ITEM
WHERE ITEM.I_ID = NEW_OL_LOCAL.I_ID
);
SET ( OL_DIST_INFO , S_DATA , S_QUANTITY )
= ( SELECT OL_DIST_INFO
, S_DATA
, S_QUANTITY
FROM NEW TABLE ( UPDATE STOCK
INCLUDE ( OL_DIST_INFO CHAR( 24 ) )
SET S_QUANTITY = CASE WHEN S_QUANTITY -
NEW_OL_LOCAL.I_QTY >= 10
THEN S_QUANTITY - NEW_OL_LOCAL.I_QTY
ELSE S_QUANTITY - NEW_OL_LOCAL.I_QTY + 91
END
, S_ORDER_CNT = S_ORDER_CNT + SMALLINT( 1 )
, S_YTD = S_YTD + NEW_OL_LOCAL.I_QTY
, OL_DIST_INFO = CASE D_ID WHEN SMALLINT( 1 )
WHEN SMALLINT( 2 ) THEN
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
END
WHERE S_I_ID = NEW_OL_LOCAL.I_ID
AND S_W_ID = NEW_OL_LOCAL.W_ID
) AS U
)
;
RETURN VALUES( VAR.I_PRICE
, VAR.I_NAME
, VAR.I_DATA
, VAR.OL_DIST_INFO
, VAR.S_DATA
, VAR.S_QUANTITY
)
;
END
%
CREATE FUNCTION NEW_WH ( O_ID INTEGER
, W_ID INTEGER
, D_ID SMALLINT
, C_ID INTEGER
, O_OL_CNT SMALLINT
, O_ALL_LOCAL SMALLINT
)
RETURNS TABLE ( W_TAX REAL
, C_DISCOUNT REAL
, C_LAST VARCHAR(16)
, C_CREDIT CHAR(2)
, O_ENTRY_D TIMESTAMP
)
SPECIFIC NEW_WH
MODIFIES SQL DATA DETERMINISTIC NO EXTERNAL ACTION LANGUAGE SQL
VAR: BEGIN ATOMIC
DECLARE C_DISCOUNT REAL ;
DECLARE C_LAST VARCHAR(16) ;
DECLARE C_CREDIT CHAR(2) ;
DECLARE W_TAX REAL ;
DECLARE O_ENTRY_D TIMESTAMP ;
SET O_ENTRY_D = CURRENT_TIMESTAMP ;
INSERT
INTO NEW_ORDER ( NO_O_ID, NO_D_ID, NO_W_ID )
VALUES ( O_ID
, D_ID

```

```

, W_ID
)
;
INSERT
INTO ORDERS ( O_C_ID, O_ENTRY_D, O_CARRIER_ID, O_OL_CNT, O_ALL_LOCAL
O_ID, O_W_ID, O_D_ID )
VALUES ( C_ID
, O_ENTRY_D
, 0
, O_OL_CNT
, O_ALL_LOCAL
, O_ID
, W_ID
, D_ID
)
;
SET ( C_DISCOUNT, C_LAST, C_CREDIT )
= ( SELECT C_DISCOUNT, C_LAST, C_CREDIT
FROM CUSTOMER
WHERE C_ID = NEW_WH.C_ID
AND C_W_ID = W_ID
AND C_D_ID = D_ID
)
;
SET W_TAX
= ( SELECT W_TAX
FROM WAREHOUSE
WHERE W_ID = NEW_WH.W_ID
)
;
RETURN VALUES ( W_TAX , C_DISCOUNT , C_LAST , C_CREDIT, O_ENTRY_D ) ;
END
%
Src.Srv/cat-proc.ddl
CREATE PROCEDURE news
(in new_in varchar(262) FOR BIT DATA,
out new_out varchar(682) FOR BIT DATA)
LANGUAGE C
PARAMETER STYLE GENERAL
EXTERNAL NAME '/home/tpcc/sql/lib/function/news!news'
not fenced;
CREATE PROCEDURE ords
(in ord_in varchar(42) FOR BIT DATA,
out ord_out varchar(822) FOR BIT DATA)
LANGUAGE C
PARAMETER STYLE GENERAL
EXTERNAL NAME '/home/tpcc/sql/lib/function/ords!ords'
not fenced;
CREATE PROCEDURE dels
(in del_in varchar(14) FOR BIT DATA,
out del_out varchar(50) FOR BIT DATA)
LANGUAGE C

```

```
PARAMETER STYLE GENERAL
EXTERNAL NAME '/home/tpcc/sqllib/function/dels\dels'
not fenced;
```

Src.Srv/tpcc all sql.sqc

```
/*-----
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
**-----
/*
 * tpcc_all_sql.sqc - Client/Server code for TPCC
 */

#include <stdlib.h>
#include <errno.h>
#include "db2tpcc.h"
#include "tpccapp.h"
#include "tpccdbg.h"

#include "sqlca.h"
#include "sql.h"

// -----
// New Order SERVER
// -----

int static is_ORIGINAL( char *string, short length );

SQL_API_RC new_order_internal( char *pin, char *pout )
{
    struct out_neword_struct *neword;

    struct in_neword_struct *in_neword;

    struct sqlca sqlca ;

    int fbadItemDetected = 0 ;

    EXEC SQL BEGIN DECLARE SECTION;

    char c_last [ 16 ];
    char c_credit [ 2 ];
    float c_discount ;
    float dist_tax ;
    float ware_tax ;

    sqlint32 w_id ;
    short d_id ;
    sqlint32 c_id ;

    sqlint32 next_o_id ;

    short s_quantity ;

    sqlint32 supply_w_id ;

    short inputItemCount ;
```

```
char stockDistrictInformation [ 24 ];
char item_name[ 24 ];

char o_entry_d [27];

short allLocal ;

float item_price ;

struct i_data_type { short len ; char data[ 50 ] ; } i_data ;
struct s_data_type { short len ; char data[ 50 ] ; } s_data ;

sqlint32 id0, id1, id2, id3, id4, id5, id6, id7;
sqlint32 id8, id9, id10, id11, id12, id13, id14;

sqlint32 supply_w_id0, supply_w_id1, supply_w_id2, supply_w_id3;
sqlint32 supply_w_id4, supply_w_id5, supply_w_id6, supply_w_id7;
sqlint32 supply_w_id8, supply_w_id9, supply_w_id10, supply_w_id11;
sqlint32 supply_w_id12, supply_w_id13, supply_w_id14;

short ol_quantity0, ol_quantity1, ol_quantity2, ol_quantity3;
short ol_quantity4, ol_quantity5, ol_quantity6, ol_quantity7;
short ol_quantity8, ol_quantity9, ol_quantity10, ol_quantity11;
short ol_quantity12, ol_quantity13, ol_quantity14;

EXEC SQL END DECLARE SECTION;

int storedProcRc ;
int inputItemArrayIndex ;

char stockDistrictInformationArray [15][25];

#define stockDistrictInformation stockDistrictInformationArray[ inputItemArrayIndex ]

// Redirected input fields

#define w_id in_neword->s_W_ID
#define d_id in_neword->s_D_ID
#define c_id in_neword->s_C_ID

#define inputItemCount in_neword->s_O_OL_CNT

#define allLocal in_neword->s_all_local

// Redirected output fields

#define c_last neword->s_C_LAST
#define c_credit neword->s_C_CREDIT
#define c_discount neword->s_C_DISCOUNT
#define ware_tax neword->s_W_TAX
#define dist_tax neword->s_D_TAX
#define s_quantity neword->item[ inputItemArrayIndex ].s_S_QUANTITY
#define o_entry_d neword->s_O_ENTRY_D_time

// This output field becomes an input field to order_line

#define next_o_id neword->s_O_ID

// item price/name

#define item_name neword->item[ inputItemArrayIndex ].s_I_NAME

float i_priceArray[ 15 ];

#define item_price i_priceArray[ inputItemArrayIndex ]

// Handle the generic/brand distinction
```

```
struct i_data_type i_dataArray[ 15 ];
struct s_data_type s_dataArray[ 15 ];

#define i_data i_dataArray[ inputItemArrayIndex ]
#define s_data s_dataArray[ inputItemArrayIndex ]

// Redirect hostvars to input structure

#define id0 in_neword->in_item[0].s_OL_I_ID
#define id1 in_neword->in_item[1].s_OL_I_ID
#define id2 in_neword->in_item[2].s_OL_I_ID
#define id3 in_neword->in_item[3].s_OL_I_ID
#define id4 in_neword->in_item[4].s_OL_I_ID
#define id5 in_neword->in_item[5].s_OL_I_ID
#define id6 in_neword->in_item[6].s_OL_I_ID
#define id7 in_neword->in_item[7].s_OL_I_ID
#define id8 in_neword->in_item[8].s_OL_I_ID
#define id9 in_neword->in_item[9].s_OL_I_ID
#define id10 in_neword->in_item[10].s_OL_I_ID
#define id11 in_neword->in_item[11].s_OL_I_ID
#define id12 in_neword->in_item[12].s_OL_I_ID
#define id13 in_neword->in_item[13].s_OL_I_ID
#define id14 in_neword->in_item[14].s_OL_I_ID

#define ol_quantity0 in_neword->in_item[ 0 ].s_OL_QUANTITY
#define ol_quantity1 in_neword->in_item[ 1 ].s_OL_QUANTITY
#define ol_quantity2 in_neword->in_item[ 2 ].s_OL_QUANTITY
#define ol_quantity3 in_neword->in_item[ 3 ].s_OL_QUANTITY
#define ol_quantity4 in_neword->in_item[ 4 ].s_OL_QUANTITY
#define ol_quantity5 in_neword->in_item[ 5 ].s_OL_QUANTITY
#define ol_quantity6 in_neword->in_item[ 6 ].s_OL_QUANTITY
#define ol_quantity7 in_neword->in_item[ 7 ].s_OL_QUANTITY
#define ol_quantity8 in_neword->in_item[ 8 ].s_OL_QUANTITY
#define ol_quantity9 in_neword->in_item[ 9 ].s_OL_QUANTITY
#define ol_quantity10 in_neword->in_item[ 10 ].s_OL_QUANTITY
#define ol_quantity11 in_neword->in_item[ 11 ].s_OL_QUANTITY
#define ol_quantity12 in_neword->in_item[ 12 ].s_OL_QUANTITY
#define ol_quantity13 in_neword->in_item[ 13 ].s_OL_QUANTITY
#define ol_quantity14 in_neword->in_item[ 14 ].s_OL_QUANTITY

#define supply_w_id0 in_neword->in_item[ 0 ].s_OL_SUPPLY_W_ID
#define supply_w_id1 in_neword->in_item[ 1 ].s_OL_SUPPLY_W_ID
#define supply_w_id2 in_neword->in_item[ 2 ].s_OL_SUPPLY_W_ID
#define supply_w_id3 in_neword->in_item[ 3 ].s_OL_SUPPLY_W_ID
#define supply_w_id4 in_neword->in_item[ 4 ].s_OL_SUPPLY_W_ID
#define supply_w_id5 in_neword->in_item[ 5 ].s_OL_SUPPLY_W_ID
#define supply_w_id6 in_neword->in_item[ 6 ].s_OL_SUPPLY_W_ID
#define supply_w_id7 in_neword->in_item[ 7 ].s_OL_SUPPLY_W_ID
#define supply_w_id8 in_neword->in_item[ 8 ].s_OL_SUPPLY_W_ID
#define supply_w_id9 in_neword->in_item[ 9 ].s_OL_SUPPLY_W_ID
#define supply_w_id10 in_neword->in_item[ 10 ].s_OL_SUPPLY_W_ID
#define supply_w_id11 in_neword->in_item[ 11 ].s_OL_SUPPLY_W_ID
#define supply_w_id12 in_neword->in_item[ 12 ].s_OL_SUPPLY_W_ID
#define supply_w_id13 in_neword->in_item[ 13 ].s_OL_SUPPLY_W_ID
#define supply_w_id14 in_neword->in_item[ 14 ].s_OL_SUPPLY_W_ID

EXEC SQL DECLARE ISOL_Remote_1 CURSOR FOR

WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, ( I_PRICE * I_QTY ) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
```



```

FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 ) , :id0 , :ol_quantity0 , :supply_w_id0 )
) AS X ( OL_NUMBER , I_ID , I_QTY
, I_SUPPLY_W_ID )
) AS ITEMLIST
, TABLE( NEW_OL_ALL( I_ID
, I_QTY
, W_ID
, I_SUPPLY_W_ID
, O_ID
, D_ID
) ) AS NEW_OL_ALL
WHERE NEW_OL_ALL.I_PRICE IS NOT NULL
)
SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
)
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )
SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM DATA
) AS INS
;
EXEC SQL DECLARE ISOL_Remote_2 CURSOR FOR

```

```

WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, ( I_PRICE * I_QTY ) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 ) , :id0 , :ol_quantity0 , :supply_w_id0 )
( SMALLINT( 2 ) , :id1 , :ol_quantity1 , :supply_w_id1 )
) AS X ( OL_NUMBER , I_ID , I_QTY
, I_SUPPLY_W_ID )
) AS ITEMLIST
, TABLE( NEW_OL_ALL( I_ID
, I_QTY
, W_ID
, I_SUPPLY_W_ID
, O_ID
, D_ID
) ) AS NEW_OL_ALL
WHERE NEW_OL_ALL.I_PRICE IS NOT NULL
)
SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
)
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )
SELECT O_ID
, D_ID
, W_ID

```

```

, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM DATA
) AS INS
;
EXEC SQL DECLARE ISOL_Remote_3 CURSOR FOR
WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, ( I_PRICE * I_QTY ) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, I_QTY
) AS NEW_OL_ALL
WHERE NEW_OL_ALL.I_PRICE IS NOT NULL
)
SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
)
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )
SELECT O_ID
, D_ID
, W_ID

```

```

,OL_I_ID
,OL_SUPPLY_W_ID
,OL_DELIVERY_D
,OL_QUANTITY
,OL_AMOUNT
,OL_DIST_INFO
)
INCLUDE ( I_PRICE DECIMAL(5,2)
,I_NAME CHAR(24)
,I_DATA VARCHAR(50)
,S_DATA VARCHAR(50)
,S_QUANTITY SMALLINT )

SELECT O_ID
,D_ID
,W_ID
,OL_NUMBER
,I_ID
,I_SUPPLY_W_ID
,OL_DELIVERY_D
,I_QTY
,TOTAL_PRICE
,OL_DIST_INFO
,I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

FROM DATA

) AS INS
;

EXEC SQL DECLARE ISOL_Remote_4 CURSOR FOR

WITH DATA AS ( SELECT O_ID
,D_ID
,W_ID
,OL_NUMBER
,I_ID
,I_SUPPLY_W_ID
,(TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
,I_QTY
,(I_PRICE * I_QTY) AS TOTAL_PRICE
,OL_DIST_INFO
,I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

FROM ( SELECT :next_o_id as O_ID
,:w_id AS W_ID
,:d_id as D_ID
,OL_NUMBER
,I_ID
,I_SUPPLY_W_ID
,I_QTY

FROM Table( VALUES

( SMALLINT( 1) ,:id0 ,:ol_quantity0 ,:supply_w_id0 )
( SMALLINT( 2) ,:id1 ,:ol_quantity1 ,:supply_w_id1 )
( SMALLINT( 3) ,:id2 ,:ol_quantity2 ,:supply_w_id2 )
( SMALLINT( 4) ,:id3 ,:ol_quantity3 ,:supply_w_id3 )

) AS X ( OL_NUMBER , I_ID , I_QTY
,I_SUPPLY_W_ID )
) AS ITEMLIST
, TABLE( NEW_OL_ALL( I_ID
,I_QTY
,W_ID
,I_SUPPLY_W_ID
,O_ID
,D_ID
) AS NEW_OL_ALL

WHERE NEW_OL_ALL.I_PRICE IS NOT NULL

SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY

FROM NEW TABLE ( INSERT INTO ORDER_LINE

( OL_O_ID
,OL_D_ID
,OL_W_ID
,OL_NUMBER
,OL_I_ID
,OL_SUPPLY_W_ID
,OL_DELIVERY_D
,OL_QUANTITY
,OL_AMOUNT
,OL_DIST_INFO
)

INCLUDE ( I_PRICE DECIMAL(5,2)
,I_NAME CHAR(24)
,I_DATA VARCHAR(50)
,S_DATA VARCHAR(50)
,S_QUANTITY SMALLINT )

SELECT O_ID
,D_ID
,W_ID
,OL_NUMBER
,I_ID
,I_SUPPLY_W_ID
,OL_DELIVERY_D
,I_QTY
,TOTAL_PRICE
,OL_DIST_INFO
,I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

FROM DATA

) AS INS
;

EXEC SQL DECLARE ISOL_Remote_5 CURSOR FOR

WITH DATA AS ( SELECT O_ID
,D_ID
,W_ID
,OL_NUMBER
,I_ID
,I_SUPPLY_W_ID
,(TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
,I_QTY
,(I_PRICE * I_QTY) AS TOTAL_PRICE
,OL_DIST_INFO
,I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

FROM ( SELECT :next_o_id as O_ID
,:w_id AS W_ID
,:d_id as D_ID
,OL_NUMBER
,I_ID
,I_SUPPLY_W_ID
,I_QTY

FROM Table( VALUES

( SMALLINT( 1) ,:id0 ,:ol_quantity0 ,:supply_w_id0 )
( SMALLINT( 2) ,:id1 ,:ol_quantity1 ,:supply_w_id1 )
( SMALLINT( 3) ,:id2 ,:ol_quantity2 ,:supply_w_id2 )
( SMALLINT( 4) ,:id3 ,:ol_quantity3 ,:supply_w_id3 )

) AS X ( OL_NUMBER , I_ID , I_QTY
,I_SUPPLY_W_ID )
) AS ITEMLIST
, TABLE( NEW_OL_ALL( I_ID
,I_QTY
,W_ID
,I_SUPPLY_W_ID
,O_ID
,D_ID
) AS NEW_OL_ALL

WHERE NEW_OL_ALL.I_PRICE IS NOT NULL

SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY

FROM NEW TABLE ( INSERT INTO ORDER_LINE

( OL_O_ID
,OL_D_ID
,OL_W_ID
,OL_NUMBER
,OL_I_ID
,OL_SUPPLY_W_ID
,OL_DELIVERY_D
,OL_QUANTITY
,OL_AMOUNT
,OL_DIST_INFO
)

INCLUDE ( I_PRICE DECIMAL(5,2)
,I_NAME CHAR(24)
,I_DATA VARCHAR(50)
,S_DATA VARCHAR(50)
,S_QUANTITY SMALLINT )

SELECT O_ID
,D_ID
,W_ID
,OL_NUMBER
,I_ID
,I_SUPPLY_W_ID
,OL_DELIVERY_D
,I_QTY
,TOTAL_PRICE
,OL_DIST_INFO
,I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

FROM DATA

) AS INS
;

```

```

,W_ID
,I_SUPPLY_W_ID
,O_ID
,D_ID
)
) AS NEW_OL_ALL

WHERE NEW_OL_ALL.I_PRICE IS NOT NULL

)

SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY

FROM NEW TABLE ( INSERT INTO ORDER_LINE

( OL_O_ID
,OL_D_ID
,OL_W_ID
,OL_NUMBER
,OL_I_ID
,OL_SUPPLY_W_ID
,OL_DELIVERY_D
,OL_QUANTITY
,OL_AMOUNT
,OL_DIST_INFO
)

INCLUDE ( I_PRICE DECIMAL(5,2)
,I_NAME CHAR(24)
,I_DATA VARCHAR(50)
,S_DATA VARCHAR(50)
,S_QUANTITY SMALLINT )

SELECT O_ID
,D_ID
,W_ID
,OL_NUMBER
,I_ID
,I_SUPPLY_W_ID
,OL_DELIVERY_D
,I_QTY
,TOTAL_PRICE
,OL_DIST_INFO
,I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

FROM DATA

) AS INS
;

EXEC SQL DECLARE ISOL_Remote_5 CURSOR FOR

WITH DATA AS ( SELECT O_ID
,D_ID
,W_ID
,OL_NUMBER
,I_ID
,I_SUPPLY_W_ID
,(TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
,I_QTY
,(I_PRICE * I_QTY) AS TOTAL_PRICE
,OL_DIST_INFO
,I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

FROM ( SELECT :next_o_id as O_ID
,:w_id AS W_ID
,:d_id as D_ID
,OL_NUMBER
,I_ID
,I_SUPPLY_W_ID
,I_QTY

FROM Table( VALUES

( SMALLINT( 1) ,:id0 ,:ol_quantity0 ,:supply_w_id0 )
( SMALLINT( 2) ,:id1 ,:ol_quantity1 ,:supply_w_id1 )
( SMALLINT( 3) ,:id2 ,:ol_quantity2 ,:supply_w_id2 )
( SMALLINT( 4) ,:id3 ,:ol_quantity3 ,:supply_w_id3 )

) AS X ( OL_NUMBER , I_ID , I_QTY
,I_SUPPLY_W_ID )
) AS ITEMLIST
, TABLE( NEW_OL_ALL( I_ID
,I_QTY
,W_ID
,I_SUPPLY_W_ID
,O_ID
,D_ID
) AS NEW_OL_ALL

WHERE NEW_OL_ALL.I_PRICE IS NOT NULL

SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY

FROM NEW TABLE ( INSERT INTO ORDER_LINE

( OL_O_ID
,OL_D_ID
,OL_W_ID
,OL_NUMBER
,OL_I_ID
,OL_SUPPLY_W_ID
,OL_DELIVERY_D
,OL_QUANTITY
,OL_AMOUNT
,OL_DIST_INFO
)

INCLUDE ( I_PRICE DECIMAL(5,2)
,I_NAME CHAR(24)
,I_DATA VARCHAR(50)
,S_DATA VARCHAR(50)
,S_QUANTITY SMALLINT )

SELECT O_ID
,D_ID
,W_ID
,OL_NUMBER
,I_ID
,I_SUPPLY_W_ID
,OL_DELIVERY_D
,I_QTY
,TOTAL_PRICE
,OL_DIST_INFO
,I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

FROM DATA

) AS INS
;

```

```

,I_QTY

FROM Table( VALUES

( SMALLINT( 1) ,:id0 ,:ol_quantity0 ,:supply_w_id0 )
( SMALLINT( 2) ,:id1 ,:ol_quantity1 ,:supply_w_id1 )
( SMALLINT( 3) ,:id2 ,:ol_quantity2 ,:supply_w_id2 )
( SMALLINT( 4) ,:id3 ,:ol_quantity3 ,:supply_w_id3 )
( SMALLINT( 5) ,:id4 ,:ol_quantity4 ,:supply_w_id4 )

) AS X ( OL_NUMBER , I_ID , I_QTY
,I_SUPPLY_W_ID )
) AS ITEMLIST
, TABLE( NEW_OL_ALL( I_ID
,I_QTY
,W_ID
,I_SUPPLY_W_ID
,O_ID
,D_ID
) AS NEW_OL_ALL

WHERE NEW_OL_ALL.I_PRICE IS NOT NULL

)

SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY

FROM NEW TABLE ( INSERT INTO ORDER_LINE

( OL_O_ID
,OL_D_ID
,OL_W_ID
,OL_NUMBER
,OL_I_ID
,OL_SUPPLY_W_ID
,OL_DELIVERY_D
,OL_QUANTITY
,OL_AMOUNT
,OL_DIST_INFO
)

INCLUDE ( I_PRICE DECIMAL(5,2)
,I_NAME CHAR(24)
,I_DATA VARCHAR(50)
,S_DATA VARCHAR(50)
,S_QUANTITY SMALLINT )

SELECT O_ID
,D_ID
,W_ID
,OL_NUMBER
,I_ID
,I_SUPPLY_W_ID
,OL_DELIVERY_D
,I_QTY
,TOTAL_PRICE
,OL_DIST_INFO
,I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

FROM DATA

) AS INS
;

```

```
EXEC SQL DECLARE ISOL_Remote_6 CURSOR FOR
```

```
WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, ( I_PRICE * I_QTY ) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
```

```
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, I_QTY
```

```
FROM Table( VALUES
```

```
( SMALLINT( 1 ) , :id0 , :ol_quantity0 , :supply_w_id0 )
( SMALLINT( 2 ) , :id1 , :ol_quantity1 , :supply_w_id1 )
( SMALLINT( 3 ) , :id2 , :ol_quantity2 , :supply_w_id2 )
( SMALLINT( 4 ) , :id3 , :ol_quantity3 , :supply_w_id3 )
( SMALLINT( 5 ) , :id4 , :ol_quantity4 , :supply_w_id4 )
( SMALLINT( 6 ) , :id5 , :ol_quantity5 , :supply_w_id5 )
```

```
) AS X ( OL_NUMBER , I_ID , I_QTY
```

```
I_SUPPLY_W_ID )
```

```
) AS ITEMLIST
```

```
, TABLE( NEW_OL_ALL( I_ID
, I_QTY
, W_ID
, I_SUPPLY_W_ID
, O_ID
, D_ID
) )
```

```
) AS NEW_OL_ALL
```

```
WHERE NEW_OL_ALL.I_PRICE IS NOT NULL
```

```
)
```

```
SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
```

```
FROM NEW TABLE ( INSERT INTO ORDER_LINE
```

```
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
) )
```

```
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )
```

```
SELECT O_ID
```

```
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
```

```
FROM DATA
```

```
) AS INS
```

```
;
```

```
EXEC SQL DECLARE ISOL_Remote_7 CURSOR FOR
```

```
WITH DATA AS ( SELECT O_ID
```

```
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, ( I_PRICE * I_QTY ) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
```

```
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, I_QTY
```

```
FROM Table( VALUES
```

```
( SMALLINT( 1 ) , :id0 , :ol_quantity0 , :supply_w_id0 )
( SMALLINT( 2 ) , :id1 , :ol_quantity1 , :supply_w_id1 )
( SMALLINT( 3 ) , :id2 , :ol_quantity2 , :supply_w_id2 )
( SMALLINT( 4 ) , :id3 , :ol_quantity3 , :supply_w_id3 )
( SMALLINT( 5 ) , :id4 , :ol_quantity4 , :supply_w_id4 )
( SMALLINT( 6 ) , :id5 , :ol_quantity5 , :supply_w_id5 )
( SMALLINT( 7 ) , :id6 , :ol_quantity6 , :supply_w_id6 )
```

```
) AS X ( OL_NUMBER , I_ID , I_QTY
```

```
I_SUPPLY_W_ID )
```

```
) AS ITEMLIST
```

```
, TABLE( NEW_OL_ALL( I_ID
, I_QTY
, W_ID
```

```
, I_SUPPLY_W_ID
, O_ID
, D_ID
) ) AS NEW_OL_ALL
```

```
WHERE NEW_OL_ALL.I_PRICE IS NOT NULL
```

```
)
```

```
SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
```

```
FROM NEW TABLE ( INSERT INTO ORDER_LINE
```

```
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
) )
```

```
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )
```

```
SELECT O_ID
```

```
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
```

```
FROM DATA
```

```
) AS INS
```

```
;
```

```
EXEC SQL DECLARE ISOL_Remote_8 CURSOR FOR
```

```
WITH DATA AS ( SELECT O_ID
```

```
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, ( I_PRICE * I_QTY ) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
```

```
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, I_QTY
```

```

FROM Table( VALUES
( SMALLINT( 1 ) ,:id0 ,:ol_quantity0 ,:supply_w_id0 )
( SMALLINT( 2 ) ,:id1 ,:ol_quantity1 ,:supply_w_id1 )
( SMALLINT( 3 ) ,:id2 ,:ol_quantity2 ,:supply_w_id2 )
( SMALLINT( 4 ) ,:id3 ,:ol_quantity3 ,:supply_w_id3 )
( SMALLINT( 5 ) ,:id4 ,:ol_quantity4 ,:supply_w_id4 )
( SMALLINT( 6 ) ,:id5 ,:ol_quantity5 ,:supply_w_id5 )
( SMALLINT( 7 ) ,:id6 ,:ol_quantity6 ,:supply_w_id6 )
( SMALLINT( 8 ) ,:id7 ,:ol_quantity7 ,:supply_w_id7 )
) AS X( OL_NUMBER , I_ID , I_QTY
I_SUPPLY_W_ID )
) AS ITEMLIST
, TABLE( NEW_OL_ALL( I_ID
, I_QTY
, W_ID
, I_SUPPLY_W_ID
, O_ID
, D_ID
) AS NEW_OL_ALL
WHERE NEW_OL_ALL.I_PRICE IS NOT NULL
)
SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
)
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )
SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

```

```

FROM DATA
) AS INS
;
EXEC SQL DECLARE ISOL_Remote_9 CURSOR FOR
WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, (I_PRICE * I_QTY) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 ) ,:id0 ,:ol_quantity0 ,:supply_w_id0 )
( SMALLINT( 2 ) ,:id1 ,:ol_quantity1 ,:supply_w_id1 )
( SMALLINT( 3 ) ,:id2 ,:ol_quantity2 ,:supply_w_id2 )
( SMALLINT( 4 ) ,:id3 ,:ol_quantity3 ,:supply_w_id3 )
( SMALLINT( 5 ) ,:id4 ,:ol_quantity4 ,:supply_w_id4 )
( SMALLINT( 6 ) ,:id5 ,:ol_quantity5 ,:supply_w_id5 )
( SMALLINT( 7 ) ,:id6 ,:ol_quantity6 ,:supply_w_id6 )
( SMALLINT( 8 ) ,:id7 ,:ol_quantity7 ,:supply_w_id7 )
( SMALLINT( 9 ) ,:id8 ,:ol_quantity8 ,:supply_w_id8 )
) AS X( OL_NUMBER , I_ID , I_QTY
I_SUPPLY_W_ID )
) AS ITEMLIST
, TABLE( NEW_OL_ALL( I_ID
, I_QTY
, W_ID
, I_SUPPLY_W_ID
, O_ID
, D_ID
) AS NEW_OL_ALL
WHERE NEW_OL_ALL.I_PRICE IS NOT NULL
)
SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE

```

```

( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
)
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )
SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM DATA
) AS INS
;
EXEC SQL DECLARE ISOL_Remote_10 CURSOR FOR
WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, (I_PRICE * I_QTY) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 ) ,:id0 ,:ol_quantity0 ,:supply_w_id0 )
( SMALLINT( 2 ) ,:id1 ,:ol_quantity1 ,:supply_w_id1 )
( SMALLINT( 3 ) ,:id2 ,:ol_quantity2 ,:supply_w_id2 )
( SMALLINT( 4 ) ,:id3 ,:ol_quantity3 ,:supply_w_id3 )
( SMALLINT( 5 ) ,:id4 ,:ol_quantity4 ,:supply_w_id4 )

```

```

( SMALLINT( 6 ) ,:id5 ,:ol_quantity5 ,:supply_w_id5 )
( SMALLINT( 7 ) ,:id6 ,:ol_quantity6 ,:supply_w_id6 )
( SMALLINT( 8 ) ,:id7 ,:ol_quantity7 ,:supply_w_id7 )
( SMALLINT( 9 ) ,:id8 ,:ol_quantity8 ,:supply_w_id8 )
( SMALLINT( 10 ) ,:id9 ,:ol_quantity9 ,:supply_w_id9 )

) AS X ( OL_NUMBER , I_ID , I_QTY
, I_SUPPLY_W_ID )
) AS ITEMLIST
, TABLE( NEW_OL_ALL( I_ID
, I_QTY
, W_ID
, I_SUPPLY_W_ID
, O_ID
, D_ID
)
) AS NEW_OL_ALL
WHERE NEW_OL_ALL.I_PRICE IS NOT NULL
)

SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
)
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )
SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM DATA
) AS INS
EXEC SQL DECLARE ISOL_Remote_11 CURSOR FOR
WITH DATA AS ( SELECT O_ID

```

```

, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, ( I_PRICE * I_QTY ) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 ) ,:id0 ,:ol_quantity0 ,:supply_w_id0 )
( SMALLINT( 2 ) ,:id1 ,:ol_quantity1 ,:supply_w_id1 )
( SMALLINT( 3 ) ,:id2 ,:ol_quantity2 ,:supply_w_id2 )
( SMALLINT( 4 ) ,:id3 ,:ol_quantity3 ,:supply_w_id3 )
( SMALLINT( 5 ) ,:id4 ,:ol_quantity4 ,:supply_w_id4 )
( SMALLINT( 6 ) ,:id5 ,:ol_quantity5 ,:supply_w_id5 )
( SMALLINT( 7 ) ,:id6 ,:ol_quantity6 ,:supply_w_id6 )
( SMALLINT( 8 ) ,:id7 ,:ol_quantity7 ,:supply_w_id7 )
( SMALLINT( 9 ) ,:id8 ,:ol_quantity8 ,:supply_w_id8 )
( SMALLINT( 10 ) ,:id9 ,:ol_quantity9 ,:supply_w_id9 )
( SMALLINT( 11 ) ,:id10 ,:ol_quantity10 ,:supply_w_id10 )
) AS X ( OL_NUMBER , I_ID , I_QTY
, I_SUPPLY_W_ID )
) AS ITEMLIST
, TABLE( NEW_OL_ALL( I_ID
, I_QTY
, W_ID
, I_SUPPLY_W_ID
, O_ID
, D_ID
)
) AS NEW_OL_ALL
WHERE NEW_OL_ALL.I_PRICE IS NOT NULL
)

SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID

```

```

, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
)
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )
SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM DATA
) AS INS
;
EXEC SQL DECLARE ISOL_Remote_12 CURSOR FOR
WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, ( I_PRICE * I_QTY ) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 ) ,:id0 ,:ol_quantity0 ,:supply_w_id0 )
( SMALLINT( 2 ) ,:id1 ,:ol_quantity1 ,:supply_w_id1 )
( SMALLINT( 3 ) ,:id2 ,:ol_quantity2 ,:supply_w_id2 )
( SMALLINT( 4 ) ,:id3 ,:ol_quantity3 ,:supply_w_id3 )
( SMALLINT( 5 ) ,:id4 ,:ol_quantity4 ,:supply_w_id4 )
( SMALLINT( 6 ) ,:id5 ,:ol_quantity5 ,:supply_w_id5 )
( SMALLINT( 7 ) ,:id6 ,:ol_quantity6 ,:supply_w_id6 )
( SMALLINT( 8 ) ,:id7 ,:ol_quantity7 ,:supply_w_id7 )

```

```

( SMALLINT( 9 ) , :id8 , :ol_quantity8 , :supply_w_id8 )
( SMALLINT( 10 ) , :id9 , :ol_quantity9 , :supply_w_id9 )
( SMALLINT( 11 ) , :id10 , :ol_quantity10 , :supply_w_id10 )
( SMALLINT( 12 ) , :id11 , :ol_quantity11 , :supply_w_id11 )

) AS X ( OL_NUMBER , I_ID , I_QTY
I_SUPPLY_W_ID )
) AS ITEMLIST
, TABLE( NEW_OL_ALL( I_ID
, I_QTY
, W_ID
, I_SUPPLY_W_ID
, O_ID
, D_ID
)
) AS NEW_OL_ALL
WHERE NEW_OL_ALL.I_PRICE IS NOT NULL
)

SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
)
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )

SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

FROM DATA

) AS INS
;

EXEC SQL DECLARE ISOL_Remote_13 CURSOR FOR

WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID

```

```

, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, (I_PRICE * I_QTY) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, I_QTY

FROM Table( VALUES

( SMALLINT( 1 ) , :id0 , :ol_quantity0 , :supply_w_id0 )
( SMALLINT( 2 ) , :id1 , :ol_quantity1 , :supply_w_id1 )
( SMALLINT( 3 ) , :id2 , :ol_quantity2 , :supply_w_id2 )
( SMALLINT( 4 ) , :id3 , :ol_quantity3 , :supply_w_id3 )
( SMALLINT( 5 ) , :id4 , :ol_quantity4 , :supply_w_id4 )
( SMALLINT( 6 ) , :id5 , :ol_quantity5 , :supply_w_id5 )
( SMALLINT( 7 ) , :id6 , :ol_quantity6 , :supply_w_id6 )
( SMALLINT( 8 ) , :id7 , :ol_quantity7 , :supply_w_id7 )
( SMALLINT( 9 ) , :id8 , :ol_quantity8 , :supply_w_id8 )
( SMALLINT( 10 ) , :id9 , :ol_quantity9 , :supply_w_id9 )
( SMALLINT( 11 ) , :id10 , :ol_quantity10 , :supply_w_id10 )
( SMALLINT( 12 ) , :id11 , :ol_quantity11 , :supply_w_id11 )
( SMALLINT( 13 ) , :id12 , :ol_quantity12 , :supply_w_id12 )

) AS X ( OL_NUMBER , I_ID , I_QTY
I_SUPPLY_W_ID )
) AS ITEMLIST
, TABLE( NEW_OL_ALL( I_ID
, I_QTY
, W_ID
, I_SUPPLY_W_ID
, O_ID
, D_ID
)
) AS NEW_OL_ALL
WHERE NEW_OL_ALL.I_PRICE IS NOT NULL
)

SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
( OL_O_ID
, OL_D_ID
, OL_W_ID

```

```

, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
)

INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )

SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

FROM DATA

) AS INS
;

EXEC SQL DECLARE ISOL_Remote_14 CURSOR FOR

WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, (I_PRICE * I_QTY) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, I_QTY

FROM Table( VALUES

( SMALLINT( 1 ) , :id0 , :ol_quantity0 , :supply_w_id0 )
( SMALLINT( 2 ) , :id1 , :ol_quantity1 , :supply_w_id1 )
( SMALLINT( 3 ) , :id2 , :ol_quantity2 , :supply_w_id2 )
( SMALLINT( 4 ) , :id3 , :ol_quantity3 , :supply_w_id3 )
( SMALLINT( 5 ) , :id4 , :ol_quantity4 , :supply_w_id4 )
( SMALLINT( 6 ) , :id5 , :ol_quantity5 , :supply_w_id5 )
( SMALLINT( 7 ) , :id6 , :ol_quantity6 , :supply_w_id6 )

```

```

( SMALLINT( 8 ) ,:id7 ,:ol_quantity7 ,:supply_w_id7 )
( SMALLINT( 9 ) ,:id8 ,:ol_quantity8 ,:supply_w_id8 )
( SMALLINT( 10 ) ,:id9 ,:ol_quantity9 ,:supply_w_id9 )
( SMALLINT( 11 ) ,:id10 ,:ol_quantity10 ,:supply_w_id10 )
( SMALLINT( 12 ) ,:id11 ,:ol_quantity11 ,:supply_w_id11 )
( SMALLINT( 13 ) ,:id12 ,:ol_quantity12 ,:supply_w_id12 )
( SMALLINT( 14 ) ,:id13 ,:ol_quantity13 ,:supply_w_id13 )

) AS X ( OL_NUMBER , I_ID , I_QTY
, I_SUPPLY_W_ID )
) AS ITEMLIST
, TABLE( NEW_OL_ALL( I_ID
, I_QTY
, W_ID
, I_SUPPLY_W_ID
, O_ID
, D_ID
) ) AS NEW_OL_ALL
WHERE NEW_OL_ALL.I_PRICE IS NOT NULL
)

SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
)
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )
SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM DATA
) AS INS
;

```

```

EXEC SQL DECLARE ISOL_Remote_15 CURSOR FOR
WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, (I_PRICE * I_QTY) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 ) ,:id0 ,:ol_quantity0 ,:supply_w_id0 )
( SMALLINT( 2 ) ,:id1 ,:ol_quantity1 ,:supply_w_id1 )
( SMALLINT( 3 ) ,:id2 ,:ol_quantity2 ,:supply_w_id2 )
( SMALLINT( 4 ) ,:id3 ,:ol_quantity3 ,:supply_w_id3 )
( SMALLINT( 5 ) ,:id4 ,:ol_quantity4 ,:supply_w_id4 )
( SMALLINT( 6 ) ,:id5 ,:ol_quantity5 ,:supply_w_id5 )
( SMALLINT( 7 ) ,:id6 ,:ol_quantity6 ,:supply_w_id6 )
( SMALLINT( 8 ) ,:id7 ,:ol_quantity7 ,:supply_w_id7 )
( SMALLINT( 9 ) ,:id8 ,:ol_quantity8 ,:supply_w_id8 )
( SMALLINT( 10 ) ,:id9 ,:ol_quantity9 ,:supply_w_id9 )
( SMALLINT( 11 ) ,:id10 ,:ol_quantity10 ,:supply_w_id10 )
( SMALLINT( 12 ) ,:id11 ,:ol_quantity11 ,:supply_w_id11 )
( SMALLINT( 13 ) ,:id12 ,:ol_quantity12 ,:supply_w_id12 )
( SMALLINT( 14 ) ,:id13 ,:ol_quantity13 ,:supply_w_id13 )
( SMALLINT( 15 ) ,:id14 ,:ol_quantity14 ,:supply_w_id14 )
) AS X ( OL_NUMBER , I_ID , I_QTY
, I_SUPPLY_W_ID )
) AS ITEMLIST
, TABLE( NEW_OL_ALL( I_ID
, I_QTY
, W_ID
, I_SUPPLY_W_ID
, O_ID
, D_ID
) ) AS NEW_OL_ALL
WHERE NEW_OL_ALL.I_PRICE IS NOT NULL

```

```

)
SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
)
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )
SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM DATA
) AS INS
;
EXEC SQL DECLARE ISOL_Local_1 CURSOR FOR
WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, W_ID AS I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, (I_PRICE * I_QTY) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 ) ,:id0 ,:ol_quantity0 )
) AS X ( OL_NUMBER , I_ID , I_QTY
) AS ITEMLIST

```

```

        ,TABLE( NEW_OL_LOCAL( I_ID
                    ,I_QTY
                    ,W_ID
                    ,O_ID
                    ,D_ID
                )
            ) AS NEW_OL_LOCAL
    WHERE NEW_OL_LOCAL.I_PRICE IS NOT NULL
)

SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
    ( OL_O_ID
    ,OL_D_ID
    ,OL_W_ID
    ,OL_NUMBER
    ,OL_I_ID
    ,OL_SUPPLY_W_ID
    ,OL_DELIVERY_D
    ,OL_QUANTITY
    ,OL_AMOUNT
    ,OL_DIST_INFO
    )
    INCLUDE ( I_PRICE DECIMAL(5,2)
    ,I_NAME CHAR(24)
    ,I_DATA VARCHAR(50)
    ,S_DATA VARCHAR(50)
    ,S_QUANTITY SMALLINT )

    SELECT O_ID
    ,D_ID
    ,W_ID
    ,OL_NUMBER
    ,I_ID
    ,I_SUPPLY_W_ID
    ,OL_DELIVERY_D
    ,I_QTY
    ,TOTAL_PRICE
    ,OL_DIST_INFO
    ,I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

    FROM DATA
    ) AS INS
;

EXEC SQL DECLARE ISOL_Local_2 CURSOR FOR

WITH DATA AS ( SELECT O_ID
    ,D_ID
    ,W_ID
    ,OL_NUMBER
    ,I_ID
    ,W_ID AS I_SUPPLY_W_ID
    ,(TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
    ,I_QTY
    ,(I_PRICE * I_QTY) AS TOTAL_PRICE
    ,OL_DIST_INFO
    ,I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

    FROM ( SELECT :next_o_id as O_ID
        ,:w_id AS W_ID
        ,:d_id as D_ID
        ,OL_NUMBER
        ,I_ID
    )

```

```

        ,I_QTY
    FROM Table( VALUES
        ( SMALLINT(1) ,:id0 ,:ol_quantity0 )
        ,( SMALLINT(2) ,:id1 ,:ol_quantity1 )
    ) AS X ( OL_NUMBER , I_ID , I_QTY )
    ) AS ITEMLIST
    ,TABLE( NEW_OL_LOCAL( I_ID
                    ,I_QTY
                    ,W_ID
                    ,O_ID
                    ,D_ID
                )
            ) AS NEW_OL_LOCAL
    WHERE NEW_OL_LOCAL.I_PRICE IS NOT NULL
)

SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
    ( OL_O_ID
    ,OL_D_ID
    ,OL_W_ID
    ,OL_NUMBER
    ,OL_I_ID
    ,OL_SUPPLY_W_ID
    ,OL_DELIVERY_D
    ,OL_QUANTITY
    ,OL_AMOUNT
    ,OL_DIST_INFO
    )
    INCLUDE ( I_PRICE DECIMAL(5,2)
    ,I_NAME CHAR(24)
    ,I_DATA VARCHAR(50)
    ,S_DATA VARCHAR(50)
    ,S_QUANTITY SMALLINT )

    SELECT O_ID
    ,D_ID
    ,W_ID
    ,OL_NUMBER
    ,I_ID
    ,I_SUPPLY_W_ID
    ,OL_DELIVERY_D
    ,I_QTY
    ,TOTAL_PRICE
    ,OL_DIST_INFO
    ,I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

    FROM DATA
    ) AS INS
;

EXEC SQL DECLARE ISOL_Local_3 CURSOR FOR

WITH DATA AS ( SELECT O_ID
    ,D_ID
    ,W_ID
    ,OL_NUMBER
    ,I_ID
    ,W_ID AS I_SUPPLY_W_ID
    ,(TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D

```

```

        ,I_QTY
    ,( I_PRICE * I_QTY ) AS TOTAL_PRICE
    ,OL_DIST_INFO
    ,I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

    FROM ( SELECT :next_o_id as O_ID
        ,:w_id AS W_ID
        ,:d_id as D_ID
        ,OL_NUMBER
        ,I_ID
        ,I_QTY
    )
    FROM Table( VALUES
        ( SMALLINT(1) ,:id0 ,:ol_quantity0 )
        ,( SMALLINT(2) ,:id1 ,:ol_quantity1 )
        ,( SMALLINT(3) ,:id2 ,:ol_quantity2 )
    ) AS X ( OL_NUMBER , I_ID , I_QTY )
    ) AS ITEMLIST
    ,TABLE( NEW_OL_LOCAL( I_ID
                    ,I_QTY
                    ,W_ID
                    ,O_ID
                    ,D_ID
                )
            ) AS NEW_OL_LOCAL
    WHERE NEW_OL_LOCAL.I_PRICE IS NOT NULL
)

SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
    ( OL_O_ID
    ,OL_D_ID
    ,OL_W_ID
    ,OL_NUMBER
    ,OL_I_ID
    ,OL_SUPPLY_W_ID
    ,OL_DELIVERY_D
    ,OL_QUANTITY
    ,OL_AMOUNT
    ,OL_DIST_INFO
    )
    INCLUDE ( I_PRICE DECIMAL(5,2)
    ,I_NAME CHAR(24)
    ,I_DATA VARCHAR(50)
    ,S_DATA VARCHAR(50)
    ,S_QUANTITY SMALLINT )

    SELECT O_ID
    ,D_ID
    ,W_ID
    ,OL_NUMBER
    ,I_ID
    ,I_SUPPLY_W_ID
    ,OL_DELIVERY_D
    ,I_QTY
    ,TOTAL_PRICE
    ,OL_DIST_INFO
    ,I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

    FROM DATA
    ) AS INS

```



```

;
EXEC SQL DECLARE ISOL_Local_4 CURSOR FOR
WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, W_ID AS I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, ( I_PRICE * I_QTY ) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 ) , :id0 , :ol_quantity0 )
, ( SMALLINT( 2 ) , :id1 , :ol_quantity1 )
, ( SMALLINT( 3 ) , :id2 , :ol_quantity2 )
, ( SMALLINT( 4 ) , :id3 , :ol_quantity3 )
) AS X ( OL_NUMBER , I_ID , I_QTY )
) AS ITEMLIST
, TABLE( NEW_OL_LOCAL( I_ID
, I_QTY
, W_ID
, O_ID
, D_ID
) AS NEW_OL_LOCAL
WHERE NEW_OL_LOCAL.I_PRICE IS NOT NULL
)
SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
)
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )
SELECT O_ID
, D_ID
, W_ID

```

```

, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM DATA
) AS INS
;
EXEC SQL DECLARE ISOL_Local_5 CURSOR FOR
WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, W_ID AS I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, ( I_PRICE * I_QTY ) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 ) , :id0 , :ol_quantity0 )
, ( SMALLINT( 2 ) , :id1 , :ol_quantity1 )
, ( SMALLINT( 3 ) , :id2 , :ol_quantity2 )
, ( SMALLINT( 4 ) , :id3 , :ol_quantity3 )
, ( SMALLINT( 5 ) , :id4 , :ol_quantity4 )
) AS X ( OL_NUMBER , I_ID , I_QTY )
) AS ITEMLIST
, TABLE( NEW_OL_LOCAL( I_ID
, I_QTY
, W_ID
, O_ID
, D_ID
) AS NEW_OL_LOCAL
WHERE NEW_OL_LOCAL.I_PRICE IS NOT NULL
)
SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
)

```

```

, OL_AMOUNT
, OL_DIST_INFO
)
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )
SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM DATA
) AS INS
;
EXEC SQL DECLARE ISOL_Local_6 CURSOR FOR
WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, W_ID AS I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, ( I_PRICE * I_QTY ) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 ) , :id0 , :ol_quantity0 )
, ( SMALLINT( 2 ) , :id1 , :ol_quantity1 )
, ( SMALLINT( 3 ) , :id2 , :ol_quantity2 )
, ( SMALLINT( 4 ) , :id3 , :ol_quantity3 )
, ( SMALLINT( 5 ) , :id4 , :ol_quantity4 )
, ( SMALLINT( 6 ) , :id5 , :ol_quantity5 )
) AS X ( OL_NUMBER , I_ID , I_QTY )
) AS ITEMLIST
, TABLE( NEW_OL_LOCAL( I_ID
, I_QTY
, W_ID
, O_ID
, D_ID
) AS NEW_OL_LOCAL
WHERE NEW_OL_LOCAL.I_PRICE IS NOT NULL
)

```



```

;
EXEC SQL DECLARE ISOL_Local_9 CURSOR FOR
WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, W_ID AS I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, (I_PRICE * I_QTY) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 ) , :id0 , :ol_quantity0 )
, ( SMALLINT( 2 ) , :id1 , :ol_quantity1 )
, ( SMALLINT( 3 ) , :id2 , :ol_quantity2 )
, ( SMALLINT( 4 ) , :id3 , :ol_quantity3 )
, ( SMALLINT( 5 ) , :id4 , :ol_quantity4 )
, ( SMALLINT( 6 ) , :id5 , :ol_quantity5 )
, ( SMALLINT( 7 ) , :id6 , :ol_quantity6 )
, ( SMALLINT( 8 ) , :id7 , :ol_quantity7 )
, ( SMALLINT( 9 ) , :id8 , :ol_quantity8 )
) AS X ( OL_NUMBER , I_ID , I_QTY )
) AS ITEMLIST
, TABLE( NEW_OL_LOCAL( I_ID
, I_QTY
, W_ID
, O_ID
, D_ID
) AS NEW_OL_LOCAL
WHERE NEW_OL_LOCAL.I_PRICE IS NOT NULL
)
SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
)
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)

```

```

, S_QUANTITY SMALLINT )
SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM DATA
) AS INS
;
EXEC SQL DECLARE ISOL_Local_10 CURSOR FOR
WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, W_ID AS I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, (I_PRICE * I_QTY) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 ) , :id0 , :ol_quantity0 )
, ( SMALLINT( 2 ) , :id1 , :ol_quantity1 )
, ( SMALLINT( 3 ) , :id2 , :ol_quantity2 )
, ( SMALLINT( 4 ) , :id3 , :ol_quantity3 )
, ( SMALLINT( 5 ) , :id4 , :ol_quantity4 )
, ( SMALLINT( 6 ) , :id5 , :ol_quantity5 )
, ( SMALLINT( 7 ) , :id6 , :ol_quantity6 )
, ( SMALLINT( 8 ) , :id7 , :ol_quantity7 )
, ( SMALLINT( 9 ) , :id8 , :ol_quantity8 )
, ( SMALLINT( 10 ) , :id9 , :ol_quantity9 )
) AS X ( OL_NUMBER , I_ID , I_QTY )
) AS ITEMLIST
, TABLE( NEW_OL_LOCAL( I_ID
, I_QTY
, W_ID
, O_ID
, D_ID
) AS NEW_OL_LOCAL
WHERE NEW_OL_LOCAL.I_PRICE IS NOT NULL
)
SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY

```

```

FROM NEW TABLE ( INSERT INTO ORDER_LINE
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
)
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )
SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM DATA
) AS INS
;
EXEC SQL DECLARE ISOL_Local_11 CURSOR FOR
WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, W_ID AS I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, (I_PRICE * I_QTY) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 ) , :id0 , :ol_quantity0 )
, ( SMALLINT( 2 ) , :id1 , :ol_quantity1 )
, ( SMALLINT( 3 ) , :id2 , :ol_quantity2 )
, ( SMALLINT( 4 ) , :id3 , :ol_quantity3 )
, ( SMALLINT( 5 ) , :id4 , :ol_quantity4 )
, ( SMALLINT( 6 ) , :id5 , :ol_quantity5 )
, ( SMALLINT( 7 ) , :id6 , :ol_quantity6 )
, ( SMALLINT( 8 ) , :id7 , :ol_quantity7 )
, ( SMALLINT( 9 ) , :id8 , :ol_quantity8 )
, ( SMALLINT( 10 ) , :id9 , :ol_quantity9 )
) AS X ( OL_NUMBER , I_ID , I_QTY )
) AS ITEMLIST
, TABLE( NEW_OL_LOCAL( I_ID
, I_QTY
, W_ID
, O_ID
, D_ID
) AS NEW_OL_LOCAL
WHERE NEW_OL_LOCAL.I_PRICE IS NOT NULL
)
SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY

```

```

        , ( SMALLINT( 11 ) , :id10 , :ol_quantity10 )
    ) AS X ( OL_NUMBER , I_ID , I_QTY )
) AS ITEMLIST
, TABLE( NEW_OL_LOCAL( I_ID
, I_QTY
, W_ID
, O_ID
, D_ID
)
) AS NEW_OL_LOCAL
WHERE NEW_OL_LOCAL.I_PRICE IS NOT NULL
)
SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
)
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )
SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM DATA
) AS INS
;
EXEC SQL DECLARE ISOL_Local_12 CURSOR FOR
WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, W_ID AS I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, (I_PRICE * I_QTY) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

```

```

FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 ) , :id0 , :ol_quantity0 )
, ( SMALLINT( 2 ) , :id1 , :ol_quantity1 )
, ( SMALLINT( 3 ) , :id2 , :ol_quantity2 )
, ( SMALLINT( 4 ) , :id3 , :ol_quantity3 )
, ( SMALLINT( 5 ) , :id4 , :ol_quantity4 )
, ( SMALLINT( 6 ) , :id5 , :ol_quantity5 )
, ( SMALLINT( 7 ) , :id6 , :ol_quantity6 )
, ( SMALLINT( 8 ) , :id7 , :ol_quantity7 )
, ( SMALLINT( 9 ) , :id8 , :ol_quantity8 )
, ( SMALLINT( 10 ) , :id9 , :ol_quantity9 )
, ( SMALLINT( 11 ) , :id10 , :ol_quantity10 )
, ( SMALLINT( 12 ) , :id11 , :ol_quantity11 )
)
) AS X ( OL_NUMBER , I_ID , I_QTY )
) AS ITEMLIST
, TABLE( NEW_OL_LOCAL( I_ID
, I_QTY
, W_ID
, O_ID
, D_ID
)
) AS NEW_OL_LOCAL
WHERE NEW_OL_LOCAL.I_PRICE IS NOT NULL
)
SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY
, OL_AMOUNT
, OL_DIST_INFO
)
INCLUDE ( I_PRICE DECIMAL(5,2)
, I_NAME CHAR(24)
, I_DATA VARCHAR(50)
, S_DATA VARCHAR(50)
, S_QUANTITY SMALLINT )
SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, I_SUPPLY_W_ID
, I_SUPPLY_W_ID
, OL_DELIVERY_D
, I_QTY
, TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM DATA
) AS INS
;
EXEC SQL DECLARE ISOL_Local_13 CURSOR FOR
WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, W_ID AS I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, (I_PRICE * I_QTY) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY

```

```

FROM DATA
) AS INS
;
EXEC SQL DECLARE ISOL_Local_13 CURSOR FOR
WITH DATA AS ( SELECT O_ID
, D_ID
, W_ID
, OL_NUMBER
, I_ID
, W_ID AS I_SUPPLY_W_ID
, (TIMESTAMP('0001-01-01 00:00:00')) AS OL_DELIVERY_D
, I_QTY
, (I_PRICE * I_QTY) AS TOTAL_PRICE
, OL_DIST_INFO
, I_PRICE, I_NAME, I_DATA, S_DATA, S_QUANTITY
FROM ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 ) , :id0 , :ol_quantity0 )
, ( SMALLINT( 2 ) , :id1 , :ol_quantity1 )
, ( SMALLINT( 3 ) , :id2 , :ol_quantity2 )
, ( SMALLINT( 4 ) , :id3 , :ol_quantity3 )
, ( SMALLINT( 5 ) , :id4 , :ol_quantity4 )
, ( SMALLINT( 6 ) , :id5 , :ol_quantity5 )
, ( SMALLINT( 7 ) , :id6 , :ol_quantity6 )
, ( SMALLINT( 8 ) , :id7 , :ol_quantity7 )
, ( SMALLINT( 9 ) , :id8 , :ol_quantity8 )
, ( SMALLINT( 10 ) , :id9 , :ol_quantity9 )
, ( SMALLINT( 11 ) , :id10 , :ol_quantity10 )
, ( SMALLINT( 12 ) , :id11 , :ol_quantity11 )
, ( SMALLINT( 13 ) , :id12 , :ol_quantity12 )
)
) AS X ( OL_NUMBER , I_ID , I_QTY )
) AS ITEMLIST
, TABLE( NEW_OL_LOCAL( I_ID
, I_QTY
, W_ID
, O_ID
, D_ID
)
) AS NEW_OL_LOCAL
WHERE NEW_OL_LOCAL.I_PRICE IS NOT NULL
)
SELECT I_PRICE , I_NAME , I_DATA , OL_DIST_INFO , S_DATA , S_QUANTITY
FROM NEW TABLE ( INSERT INTO ORDER_LINE
( OL_O_ID
, OL_D_ID
, OL_W_ID
, OL_NUMBER
, OL_I_ID
, OL_SUPPLY_W_ID
, OL_DELIVERY_D
, OL_QUANTITY

```



```

;

// Start processing

in_neword = (struct in_neword_struct *) pin;
neword = (struct out_neword_struct *) pout;

#ifdef DEBUGIT
new_debug( neword, in_neword, "SP upon entry");
#endif

// Using I_PRICE == 0 as a flag to the client that the ITEM was not fetched (hence bad).

for ( inputItemArrayIndex = 0; inputItemArrayIndex < in_neword->s_O_OL_CNT;
inputItemArrayIndex++)
{
i_priceArray[ inputItemArrayIndex ] = 0;
}

neword->deadlocks = -1;

retry_tran:

neword->deadlocks++;

EXEC SQL

SELECT D_TAX, D_NEXT_O_ID INTO :dist_tax , :next_o_id

FROM OLD TABLE ( UPDATE DISTRICT

SET D_NEXT_O_ID = D_NEXT_O_ID + 1

WHERE D_W_ID = :w_id
AND D_ID = :d_id

) AS OT

;

if ( sqlca.sqlcode != 0 )
{
DLCHK( retry_tran );
sqlerror( NEWORD_SQL, "DISTRICT", __FILE__, __LINE__, &sqlca );
goto ferror;
}

#define NEW_CURSOR_OPEN_ERROR
{
if( sqlca.sqlcode != 0 )
{
goto sql_error;
}
}

#define NEW_CURSOR_ERROR
{
if( sqlca.sqlcode == 0 )
{
neword->s_O_OL_CNT ++;
}
else
if( sqlca.sqlcode == +100 )
{
break;
}
else
goto sql_error;
}

if ( allLocal )
{

```

```

switch( inputItemCount )
{
case 1:
EXEC SQL OPEN ISOL_Local_1;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0; inputItemArrayIndex < inputItemCount;
inputItemArrayIndex++)
{
EXEC SQL FETCH ISOL_Local_1
INTO :item_price, :item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity;
NEW_CURSOR_ERROR
}
break;
case 2:
EXEC SQL OPEN ISOL_Local_2;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0; inputItemArrayIndex < inputItemCount;
inputItemArrayIndex++)
{
EXEC SQL FETCH ISOL_Local_2
INTO :item_price, :item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity;
NEW_CURSOR_ERROR
}
break;
case 3:
EXEC SQL OPEN ISOL_Local_3;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0; inputItemArrayIndex < inputItemCount;
inputItemArrayIndex++)
{
EXEC SQL FETCH ISOL_Local_3
INTO :item_price, :item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity;
NEW_CURSOR_ERROR
}
break;
case 4:
EXEC SQL OPEN ISOL_Local_4;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0; inputItemArrayIndex < inputItemCount;
inputItemArrayIndex++)
{
EXEC SQL FETCH ISOL_Local_4
INTO :item_price, :item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity;
NEW_CURSOR_ERROR
}
break;
case 5:
EXEC SQL OPEN ISOL_Local_5;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0; inputItemArrayIndex < inputItemCount;
inputItemArrayIndex++)
{
EXEC SQL FETCH ISOL_Local_5
INTO :item_price, :item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity;
NEW_CURSOR_ERROR
}
break;
case 6:
EXEC SQL OPEN ISOL_Local_6;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0; inputItemArrayIndex < inputItemCount;
inputItemArrayIndex++)
{
EXEC SQL FETCH ISOL_Local_6
INTO :item_price, :item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity;
NEW_CURSOR_ERROR
}
break;
case 7:
EXEC SQL OPEN ISOL_Local_7;

```

```

NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0; inputItemArrayIndex < inputItemCount;
inputItemArrayIndex++)
{
EXEC SQL FETCH ISOL_Local_7
INTO :item_price, :item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity;
NEW_CURSOR_ERROR
}
break;
case 8:
EXEC SQL OPEN ISOL_Local_8;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0; inputItemArrayIndex < inputItemCount;
inputItemArrayIndex++)
{
EXEC SQL FETCH ISOL_Local_8
INTO :item_price, :item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity;
NEW_CURSOR_ERROR
}
break;
case 9:
EXEC SQL OPEN ISOL_Local_9;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0; inputItemArrayIndex < inputItemCount;
inputItemArrayIndex++)
{
EXEC SQL FETCH ISOL_Local_9
INTO :item_price, :item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity;
NEW_CURSOR_ERROR
}
break;
case 10:
EXEC SQL OPEN ISOL_Local_10;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0; inputItemArrayIndex < inputItemCount;
inputItemArrayIndex++)
{
EXEC SQL FETCH ISOL_Local_10
INTO :item_price, :item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity;
NEW_CURSOR_ERROR
}
break;
case 11:
EXEC SQL OPEN ISOL_Local_11;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0; inputItemArrayIndex < inputItemCount;
inputItemArrayIndex++)
{
EXEC SQL FETCH ISOL_Local_11
INTO :item_price, :item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity;
NEW_CURSOR_ERROR
}
break;
case 12:
EXEC SQL OPEN ISOL_Local_12;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0; inputItemArrayIndex < inputItemCount;
inputItemArrayIndex++)
{
EXEC SQL FETCH ISOL_Local_12
INTO :item_price, :item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity;
NEW_CURSOR_ERROR
}
break;
case 13:
EXEC SQL OPEN ISOL_Local_13;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0; inputItemArrayIndex < inputItemCount;
inputItemArrayIndex++)
{

```



```

for ( inputItemArrayIndex = 0 ;
      inputItemArrayIndex < in_newword->s_O_OL_CNT // from input
      && i_priceArray[ inputItemArrayIndex ] != 0 ;
      inputItemArrayIndex++ )
{
    // s_l_NAME, and s_S_QUANTITY already set as output host variables

    newword->item[ inputItemArrayIndex ].s_l_PRICE = i_priceArray[ inputItemArrayIndex ] ;

    if ( is_ORIGINAL( s_dataArray[ inputItemArrayIndex ].data,
                    s_dataArray[ inputItemArrayIndex ].len )
        && is_ORIGINAL( i_dataArray[ inputItemArrayIndex ].data,
                    i_dataArray[ inputItemArrayIndex ].len ) )
    {
        newword->item[ inputItemArrayIndex ].s_brand_generic = 'B' ;
    }
    else
    {
        newword->item[ inputItemArrayIndex ].s_brand_generic = 'G' ;
    }
}

EXEC SQL

SELECT W_TAX, C_DISCOUNT, C_LAST, C_CREDIT, O_ENTRY_D

INTO :ware_tax, :c_discount, :c_last, :c_credit, :o_entry_d

FROM TABLE ( NEW_WH ( :next_o_id
                    , :w_id
                    , :d_id
                    , :c_id
                    , :inputItemCount
                    , :allLocal
                    )
              ) AS NEW_WH_TABLE
;

if ( sqlca.sqlcode == 0 )
{
    if ( newword->s_O_OL_CNT == in_newword->s_O_OL_CNT )
    {
        newword->s_transtatus = TRAN_OK ;

        EXEC SQL COMMIT ;

        if ( sqlca.sqlcode != 0 )
        {
            sqlerror(NEWWORD_SQL, "COMMIT", __FILE__, __LINE__, &sqlca) ;
            goto ferror ;
        }
    }
    else
    {
        newword->s_transtatus = INVALID_ITEM ;

        EXEC SQL ROLLBACK WORK ;

        if ( sqlca.sqlcode != 0 )
        {
            newword->s_transtatus = FATAL_SQLERROR ;

            sqlerror(NEWWORD_SQL, "ROLLBACK FAILED (INVALID ITEM)", __FILE__,
                    __LINE__, &sqlca) ;
            // no point in ferror
        }
    }
}
else

```

```

{
    DLCHK( retry_tran ) ;

    sqlerror( NEWWORD_SQL, "NEW_WH", __FILE__, __LINE__, &sqlca) ;
    goto ferror ;
}

/*-----*/
/* Return to client */
/*-----*/

mexit:

if ( sqlca.sqlcode >= 0 )
{
    storedProcRc = SQLZ_HOLD_PROC ;
}
else
{
    storedProcRc = SQLZ_DISCONNECT_PROC ;
}

#ifdef DEBUGIT
new_debug( newword, in_newword, "SP prior to return") ;
#endif

return ( storedProcRc ) ;

sql_error:

{
    char tempstr[ 4096 ] ;

    DLCHK( retry_tran ) ;

    sprintf( tempstr, "inputItemCount=%d, :next_o_id=%d, :d_id=%d, :w_id=%d",
            inputItemCount, next_o_id, d_id, w_id ) ;
    sqlerror( NEWWORD_SQL, tempstr, __FILE__, __LINE__, &sqlca ) ;
}

ferror:

newword->s_transtatus = FATAL_SQLERROR ;

EXEC SQL ROLLBACK WORK ;

if ( sqlca.sqlcode != 0 )
{
    sqlerror( NEWWORD_SQL, "ROLLBACK FAILED", __FILE__, __LINE__, &sqlca ) ;
}

goto mexit ;
}

/*
** A little function to search for the string "ORIGINAL" given a string and
** its length
*/
static unsigned char skip[256] = {8,8,8,8,8,8,8,8, /*0-9*/
                                8,8,8,8,8,8,8,8, /*10-19*/
                                8,8,8,8,8,8,8,8, /*20-29*/
                                8,8,8,8,8,8,8,8, /*30-39*/
                                8,8,8,8,8,8,8,8, /*40-49*/
                                8,8,8,8,8,8,8,8, /*50-59*/
                                8,8,8,8,8,1,8,8,8,8, /*60-69*/
                                8,4,8,3,8,8,0,8,2,7, /*70-79*/
                                8,8,6,8,8,8,8,8,8,8, /*80-89*/
                                8,8,8,8,8,8,8,8,8, /*90-99*/
                                8,8,8,8,8,8,8,8,8, /*100-109*/

```

```

                                8,8,8,8,8,8,8,8, /*110-119*/
                                8,8,8,8,8,8,8,8,8, /*120-129*/
                                8,8,8,8,8,8,8,8,8, /*130-139*/
                                8,8,8,8,8,8,8,8,8, /*140-149*/
                                8,8,8,8,8,8,8,8,8, /*150-159*/
                                8,8,8,8,8,8,8,8,8, /*160-169*/
                                8,8,8,8,8,8,8,8,8, /*170-179*/
                                8,8,8,8,8,8,8,8,8, /*180-189*/
                                8,8,8,8,8,8,8,8,8, /*190-199*/
                                8,8,8,8,8,8,8,8,8, /*200-209*/
                                8,8,8,8,8,8,8,8,8, /*210-219*/
                                8,8,8,8,8,8,8,8,8, /*220-229*/
                                8,8,8,8,8,8,8,8,8, /*230-239*/
                                8,8,8,8,8,8,8,8,8, /*240-249*/
                                8,8,8,8,8); /*250-254*/

static int is_ORIGINAL( char *string, short length )
{
    char *cur_string;
    char *end_string;
    unsigned char *skips;
    int skip_dist;
    int result = 0;

    cur_string = string+7;
    end_string = string + length;
    skips = skip;

    while (cur_string < end_string)
    {
        skip_dist = skips[*cur_string];
        while ( (skip_dist > 0) && (cur_string < end_string) )
        {
            skip_dist = skips[*cur_string += skip_dist];
        }

        if (cur_string >= end_string)
            goto exit;

        if ( cur_string[-4] != 'G' )
            goto noMatch;

        if ( memcmp( cur_string-7, "ORIGINAL", 8 ) == 0 )
        {
            result = 1;
            goto exit;
        }
    }
noMatch:
    cur_string += 8;
} /* end while */

exit:
return ( result ) ;
}

// -----
// Order Status SERVER
// -----

#undef w_id
#undef d_id
#undef c_id_input
#undef o_id
#undef o_entry_d
#undef o_carrier_d
#undef c_id
#undef c_first
#undef c_middle
#undef c_last
#undef c_balance

```



```

SQL_API_RC order_status_internal( char *pin, char *pout )
{
    struct in_ordstat_struct * in_ordstat = (struct in_ordstat_struct *) pin ;
    struct out_ordstat_struct * ordstat = (struct out_ordstat_struct *) pout ;

    struct sqlca sqlca ;

    EXEC SQL BEGIN DECLARE SECTION;

    // From input values

    ##sqlint32 w_id ;
    ##short d_id ;
    sqlint32 c_id_input ;

    struct s_data_type { short len ; char data[ 16 ] ; } c_last_input ;

    // From queries

    // From initial query

    sqlint32 o_id ;
    ##sqlint32 c_id ;
    short o_carrier_id ;
    ##sqlint64 o_entry_d ;

    char c_first[ 16 ] ;
    char c_middle[ 2 ] ;
    ##char c_last[ 16 ] ;
    double c_balance ;

    // From cursor

    sqlint32 ol_i_id ;
    sqlint32 ol_supply_w_id ;
    short ol_quantity ;
    float ol_amount ;
    char ol_delivery_d [27] ;
    ##char o_entry_d [27] ;

    EXEC SQL END DECLARE SECTION;

    ##struct s_data_type { short len ; char data[ 16 ] ; } c_last_input ;

    int storedProcRc ;
    int itemArrayIndex = 0 ;

    #define w_id      in_ordstat->s_W_ID ;
    #define d_id      in_ordstat->s_D_ID ;
    #define c_id_input in_ordstat->s_C_ID ;
    #define o_id      ordstat->s_O_ID ;
    #define o_entry_d  ordstat->s_O_ENTRY_D_time ;
    #define o_carrier_id ordstat->s_O_CARRIER_ID ;
    #define c_id      ordstat->s_C_ID ;
    #define c_first   ordstat->s_C_FIRST ;
    #define c_middle   ordstat->s_C_MIDDLE ;
    #define c_last     ordstat->s_C_LAST ;
    #define c_balance  ordstat->s_C_BALANCE ;

    EXEC SQL DECLARE read_orderline_cur CURSOR FOR

        SELECT OL_I_ID, OL_SUPPLY_W_ID, OL_QUANTITY, OL_AMOUNT,
        OL_DELIVERY_D

        FROM ORDER_LINE

        WHERE OL_W_ID = :w_id
        AND OL_D_ID = :d_id
        AND OL_O_ID = :o_id

```

```

        FOR FETCH ONLY ;

        ordstat->deadlocks = -1 ;

    #ifdef DEBUGIT
        ord_debug(ordstat, in_ordstat, "SP upon entry");
    #endif

    retry_tran:

        ordstat->deadlocks ++ ;

        if ( c_id_input == 0 )
        {
            c_last_input.len = strlen( in_ordstat->s_C_LAST ) ;
            memcpy( c_last_input.data , in_ordstat->s_C_LAST , c_last_input.len ) ;

            EXEC SQL

                SELECT O_ID, O_CARRIER_ID, O_ENTRY_D, C_BALANCE, C_FIRST, C_MIDDLE, C_ID

                INTO :o_id, :o_carrier_id , :o_entry_d , :c_balance, :c_first, :c_middle, :c_id

                FROM TABLE ( ORD_C_LAST( :w_id
                    , :d_id
                    , :c_last_input
                    )
                ) AS ORD_C_LAST

                ;
        }
        else
        {
            EXEC SQL

                SELECT O_ID, O_CARRIER_ID, O_ENTRY_D , C_BALANCE, C_FIRST,
                C_MIDDLE ,C_LAST

                INTO :o_id, :o_carrier_id , :o_entry_d , :c_balance, :c_first, :c_middle, :c_last

                FROM TABLE ( ORD_C_ID( :w_id
                    , :d_id
                    , :c_id_input
                    )
                ) AS ORD_C_ID

                ;
        }

        if ( sqlca.sqlcode != 0 )
        {
            DLCHK( retry_tran ) ;
            sqlerror( ORDSTAT_SQL, "READ CUST and ORDERS", __FILE__, __LINE__,
            &sqlca ) ;
            goto ferror ;
        }

        /*-----*/
        /* Read ORDER_LINES */
        /*-----*/

        EXEC SQL OPEN read_orderline_cur ;

        if ( sqlca.sqlcode != 0 )
        {
            DLCHK( retry_tran ) ;
            sqlerror(ORDSTAT_SQL, "OPEN CURSOR read_orderline_cur", __FILE__, __LINE__,
            &sqlca ) ;
            goto ferror ;
        }

```

```

        itemArrayIndex = 0 ;
        {
            do
            {
                EXEC SQL FETCH read_orderline_cur

                    INTO :ol_i_id , :ol_supply_w_id , :ol_quantity , :ol_amount , :ol_delivery_d ;

                if ( sqlca.sqlcode == 0 )
                {
                    ordstat->item[ itemArrayIndex ].s_OL_I_ID      = ol_i_id ;
                    ordstat->item[ itemArrayIndex ].s_OL_SUPPLY_W_ID = ol_supply_w_id ;
                    ordstat->item[ itemArrayIndex ].s_OL_QUANTITY    = ol_quantity ;
                    ordstat->item[ itemArrayIndex ].s_OL_AMOUNT      = ol_amount ;
                    strcpy(ordstat->item[ itemArrayIndex ].s_OL_DELIVERY_D_time, ol_delivery_d) ;

                    itemArrayIndex++ ;
                }
                else
                if ( sqlca.sqlcode < 0 )
                {
                    DLCHK( retry_tran ) ;
                    sqlerror(ORDSTAT_SQL, "FETCH CURSOR read_orderline_cur", __FILE__,
                    __LINE__, &sqlca ) ;
                    goto ferror ;
                }
            }
            while ( sqlca.sqlcode == 0 ) ;
        }

        ordstat->s_ol_cnt = itemArrayIndex ;

        EXEC SQL COMMIT ;

        if ( sqlca.sqlcode == 0 )
        {
            ordstat->s_transtatus = TRAN_OK ;
        }
        else
        {
            DLCHK( retry_tran ) ;
            sqlerror(ORDSTAT_SQL, "COMMIT", __FILE__, __LINE__, &sqlca) ;
            goto ferror ;
        }

        mexit:

        if ( sqlca.sqlcode >= 0 )
        {
            storedProcRc = SQLZ_HOLD_PROC ;
        }
        else
        {
            storedProcRc = SQLZ_DISCONNECT_PROC ;
        }

        #ifdef DEBUGIT
            ord_debug(ordstat, in_ordstat, "SP prior to return");
        #endif

        return ( storedProcRc ) ;

        ferror:

        ordstat->s_transtatus = FATAL_SQLERROR ;

        EXEC SQL ROLLBACK WORK ;

        if ( sqlca.sqlcode != 0 )

```

```

    {
        sqlerror(ORDSTAT_SQL, "ROLLBACK FAILED", __FILE__, __LINE__, &sqlca);
    }

    goto mexit;
}

// -----
// Delivery SERVER
// -----

#undef d_id
#undef c_id
#undef w_id
#undef o_carrier_id
#undef ol_delivery_d

SQL_API_RC delivery_internal ( char * pin, char * pout )
{
    struct in_delivery_struct * in_delivery = (struct in_delivery_struct *) pin ;
    struct out_delivery_struct * delivery = (struct out_delivery_struct *) pout ;

    struct sqlca sqlca ;

    int storedProcRc ;

    short district_id ;
    sqlint32 customer_id ;

    EXEC SQL BEGIN DECLARE SECTION;

    // input

    ///sqlint32 w_id ;
    ///short d_id ;
    ///sqlint32 c_id ;
    ///short o_carrier_id ;
    ///sqlint64 ol_delivery_d ;

    // output

    short no_o_id_indicator = 0 ;
    sqlint32 no_o_id ;

    EXEC SQL END DECLARE SECTION;

#define d_id district_id
#define c_id customer_id

#define w_id in_delivery->s_W_ID
#define o_carrier_id in_delivery->s_O_CARRIER_ID
#define ol_delivery_d in_delivery->s_O_DELIVERY_D_time

    delivery->deadlocks = -1 ;

#ifdef DEBUGIT
    del_debug( delivery, in_delivery, "SP upon entry");
#endif

    // Deadlock Handling
    // -----
    // Since we COMMIT inside the for() loop, we must take special
    // care while handling deadlocks. This is best explained by
    // an example.
    //
    // Assume we deadlock on d_id=6. This means that an order from the
    // first 5 districts have already been delivered. We will then
    // restart the loop (retry_tran). However, the loop will restart
    // at d_id = 1! This means that the second (and all subsequent)
    // time through the loop, we will deliver orders for districts that

```

```

// have already been delivered, with the net result being more than
// 10 orders being delivered.
//
// The solution to this problem is to initialize the starting point
// of the loop "before" the retry_tran label. This will ensure that
// if we deadlock, we will restart the loop with the same district
// that we deadlocked on, and we won't deliver any extra orders.
//
// NOTE: If we ever change this back to one COMMIT per transaction
// (instead of one COMMIT per iteration), then the initialization
// of d_id must be moved back into the for loop. (A rollback due
// to deadlock in this case would rollback all delivered orders so
// far, so we'd need to re-deliver them all on the next iteration.)

    d_id = 1;

retry_tran:

    delivery->deadlocks++;

    for ( ; d_id <= DISTRICTS_PER_WAREHOUSE ; d_id++ )
    {
        no_o_id = 0 ;
        no_o_id_indicator = 0 ;

        EXEC SQL BEGIN COMPOUND NOT ATOMIC STATIC

            SELECT O_ID

            INTO :no_o_id :no_o_id_indicator

            FROM TABLE ( DEL( :w_id , :d_id , :o_carrier_id ) ) AS T ;

        COMMIT ;

        END COMPOUND ;

        if ( sqlca.sqlcode == 0 )
        {
            delivery->s_O_ID[ d_id - 1 ] = no_o_id ;
        }
        else
        {
            DLCHK( retry_tran );

            sqlerror( DELIVERY_SQL , "DELIVERY", __FILE__, __LINE__, &sqlca);
            goto ferror ;
        }

        delivery->s_transtatus = TRAN_OK ;

mexit:

        if ( sqlca.sqlcode >= 0 )
        {
            storedProcRc = SQLZ_HOLD_PROC ;
        }
        else
        {
            storedProcRc = SQLZ_DISCONNECT_PROC ;
        }

#ifdef DEBUGIT
        del_debug( delivery, in_delivery, "SP prior to return");
#endif

        return ( storedProcRc );

ferror:

```

```

        delivery->s_transtatus = FATAL_SQLERROR ;

        EXEC SQL ROLLBACK WORK ;

        if ( sqlca.sqlcode != 0 )
        {
            sqlerror( DELIVERY_SQL, "ROLLBACK FAILED", __FILE__, __LINE__, &sqlca );
        }

        goto mexit ;
    }

// -----
// Stored Procedure Stubs
// -----

SQL_API_RC SQL_API_FN news( char *pin, char *pout )
{
    return new_order_internal( pin, pout ) ;
}

SQL_API_RC SQL_API_FN ords( char *pin, char *pout )
{
    return order_status_internal( pin, pout ) ;
}

SQL_API_RC SQL_API_FN dels ( char * pin, char * pout )
{
    return delivery_internal( pin, pout ) ;
}

Src.Srv/uncat-func.ddl

-----
-- Licensed Materials - Property of IBM
--
-- Governed under the terms of the International
-- License Agreement for Non-Warranted Sample Code.
--
-- (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
-- All Rights Reserved.
--
-- US Government Users Restricted Rights - Use, duplication or
-- disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
-----

-- uncat-func.ddl - Drop table function DDL
--
--
-- STOCK LEVEL
DROP SPECIFIC FUNCTION STOCK_LEVEL %
-- DELIVERY
DROP SPECIFIC FUNCTION DELIVERY %
-- ORDER STATUS
DROP SPECIFIC FUNCTION ORD_C_LAST %
DROP SPECIFIC FUNCTION ORD_C_ID %
-- PAYMENT
DROP SPECIFIC FUNCTION PAY_C_LAST %
DROP SPECIFIC FUNCTION PAY_C_ID %
-- NEW ORDER
DROP SPECIFIC FUNCTION NEW_OL_ALL %
DROP SPECIFIC FUNCTION NEW_OL_LOCAL %
DROP SPECIFIC FUNCTION NEW_WH %

DROP PROCEDURE news
    (varchar(262),varchar(682));

```

```
DROP PROCEDURE news
      (varchar(270),varchar(662));
DROP PROCEDURE news;
```

Src.Srv/uncat-proc.ddl

```
DROP PROCEDURE pays;

DROP PROCEDURE ords
      (varchar(42),varchar(822));
DROP PROCEDURE ords
      (varchar(42),varchar(446));
DROP PROCEDURE ords;

DROP PROCEDURE dels
      (varchar(14),varchar(50));
DROP PROCEDURE dels
      (varchar(22),varchar(50));
DROP PROCEDURE dels;
```

```
DROP PROCEDURE stks;
```

include/db2tpcc.h

```

*****
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****/

/*
 * db2tpcc.h - Macros and Miscellany
 *
 */

#ifndef __DB2TPCC_H
#define __DB2TPCC_H

#include <sys/types.h>

#include "lval.h"

/* *****
/* Transaction Return Codes (s_transtatus) */
/* *****

#define INVALID_ITEM      100
#define TRAN_OK          0
#define FATAL_SQLERROR   -1

/* *****
/* Definition of Unused and Bad Items */
/* *****
/* Define unused item ID to be 0. This allows the SUT to determine the
/* number of items in the order as required by 2.4.1.3 and 2.4.2.2 since
/* the assumption that any item with OL_I_ID = 0 is unused will be true.
/* This in turn requires that the value used for an invalid item is
/* equal to ITEMS + 1.
/* *****

```

```

#define INVALID_ITEM_ID (2 * ITEMS) + 1
#define UNUSED_ITEM_ID 0

#define MIN_WAREHOUSE 1
#define MAX_WAREHOUSE WAREHOUSES

/* *****
/* NURand Constants */
/* C_C_LAST_RUN and C_C_LAST_LOAD must adhere to clause 2.1.6.
/* Analysis indicates that a C_LAST delta of 85 is optimal.
/* *****

#define C_C_LAST_RUN 88
#define C_C_LAST_LOAD 173
#define C_C_ID 319
#define C_OL_I_ID 3849
#define A_C_LAST 255
#define A_C_ID 1023
#define A_OL_I_ID 8191

/* *****
/* Transaction Type Identifiers */
/* *****

#define CLIENT_SQL 0
#define NEWORD_SQL 1
#define PAYMENT_SQL 2
#define ORDSTAT_SQL 3
#define DELIVERY_SQL 4
#define STOCKLEV_SQL 5

#define SPGENERAL_PAD 3
#define SPGENERAL_ADJUST sizeof(int16_t)

struct in_neword_struct {
  int16_t len;
  int16_t pad[SPGENERAL_PAD];
  struct in_items_struct {
    int32_t s_OL_I_ID;
    int32_t s_OL_SUPPLY_W_ID;
    int16_t s_OL_QUANTITY;
    int16_t pad1[3];
  } in_item[15];
  int32_t s_C_ID;
  int32_t s_W_ID;
  int16_t s_D_ID;
  int16_t s_O_OL_CNT; /* init by SUT */
  int16_t s_all_local;
  int16_t duplicate_items;
};

struct out_neword_struct {
  int16_t len;
  int16_t pad[SPGENERAL_PAD];
  struct items_struct {
    float s_I_PRICE;
    float s_OL_AMOUNT;
    int16_t s_S_QUANTITY;
    int16_t pad2;
    char s_I_NAME[25];
    char s_brand_generic;
  } item[15];
  float s_W_TAX;
  float s_D_TAX;
  float s_C_DISCOUNT;
  float s_total_amount;
  int32_t s_O_ID;
  int16_t s_O_OL_CNT;
  int16_t s_transtatus;
  int16_t deadlocks;
};

```

```

char s_C_LAST[17];
char s_C_CREDIT[3];
char s_O_ENTRY_D_time[27];
};

struct in_payment_struct {
  int16_t len;
  int16_t pad[SPGENERAL_PAD];
  float s_H_AMOUNT;
  int32_t s_W_ID;
  int32_t s_C_W_ID;
  int32_t s_C_ID;
  int16_t s_C_D_ID;
  int16_t s_D_ID;
  char s_C_LAST[17];
};

struct out_payment_struct {
  int16_t len;
  int16_t pad[SPGENERAL_PAD];
  double s_C_CREDIT_LIM;
  double s_C_BALANCE;
  float s_C_DISCOUNT;
  int32_t s_C_ID;
  int16_t s_transtatus;
  int16_t deadlocks;
  char s_W_STREET_1[21];
  char s_W_STREET_2[21];
  char s_W_CITY[21];
  char s_W_STATE[3];
  char s_W_ZIP[10];
  char s_D_STREET_1[21];
  char s_D_STREET_2[21];
  char s_D_CITY[21];
  char s_D_STATE[3];
  char s_D_ZIP[10];
  char s_C_FIRST[17];
  char s_C_MIDDLE[3];
  char s_C_LAST[17];
  char s_C_STREET_1[21];
  char s_C_STREET_2[21];
  char s_C_CITY[21];
  char s_C_STATE[3];
  char s_C_ZIP[10];
  char s_C_PHONE[17];
  char s_C_CREDIT[3];
  char s_C_DATA[20];
  char s_H_DATE_time[27];
  char s_C_SINCE_time[27];
};

struct in_ordstat_struct {
  int16_t len;
  int16_t pad[SPGENERAL_PAD];
  int32_t s_C_ID;
  int32_t s_W_ID;
  int16_t s_D_ID;
  int16_t pad1[3];
  char s_C_LAST[17];
};

struct out_ordstat_struct {
  int16_t len;
  int16_t pad[SPGENERAL_PAD];
  double s_C_BALANCE;
  int32_t s_C_ID;
  int32_t s_O_ID;
  int16_t s_O_CARRIER_ID;
  int16_t s_ol_cnt;
};

```

```

int16_t pad1[2];
struct oitems_struct {
    double s_OL_AMOUNT;
    int32_t s_OL_ID;
    int32_t s_OL_SUPPLY_W_ID;
    int16_t s_OL_QUANTITY;
    int16_t pad2;
    char s_OL_DELIVERY_D_time[27];
} item[15];
int16_t s_transtatus;
int16_t deadlocks;
char s_C_FIRST[17];
char s_C_MIDDLE[3];
char s_C_LAST[17];
char s_O_ENTRY_D_time[27];
int16_t pad3[2];
};

struct in_delivery_struct {
    int16_t len;
    int16_t pad[SPGENERAL_PAD];
    int32_t s_W_ID;
    int16_t s_O_CARRIER_ID;
};

struct out_delivery_struct {
    int16_t len;
    int16_t pad[SPGENERAL_PAD];
    int32_t s_O_ID[10];
    int16_t s_transtatus;
    int16_t deadlocks;
};

struct in_stocklev_struct {
    int16_t len;
    int16_t pad[SPGENERAL_PAD];
    int32_t s_threshold;
    int32_t s_W_ID;
    int16_t s_D_ID;
};

struct out_stocklev_struct {
    int16_t len;
    int16_t pad[SPGENERAL_PAD];
    int32_t s_low_stock;
    int16_t s_transtatus;
    int16_t deadlocks;
};

/* ***** */
/* Transaction Prototypes */
/* ***** */

#ifdef __cplusplus
extern "C" {
#endif

extern int neword_sql(struct in_neword_struct*, struct out_neword_struct*);
extern int payment_sql(struct in_payment_struct*, struct out_payment_struct*);
extern int ordstat_sql(struct in_ordstat_struct*, struct out_ordstat_struct*);
extern int delivery_sql(struct in_delivery_struct*, struct out_delivery_struct*);
extern int stocklev_sql(struct in_stocklev_struct*, struct out_stocklev_struct*);

#ifdef __cplusplus
}
#endif

/* ***** */
/* DB2 Connect/Disconnect & Thread Context Wrappers */
/* ***** */

```

```

#ifdef __cplusplus
extern "C" {
#endif

extern int connect_to_TM(char*);
extern int connect_to_TM_auth(char*, char*, char*);
extern int disconnect_from_TM(void);

#ifdef __cplusplus
}
#endif

#ifdef __DB2TPCC_H

include/lval.h

/* lval.h - generated automatically at 20060905.1052 */

#ifdef __LVAL_H
#define __LVAL_H
#define WAREHOUSES 28032
#define DISTRICTS_PER_WAREHOUSE 10
#define CUSTOMERS_PER_DISTRICT 3000
#define ITEMS 100000
#define STOCK_PER_WAREHOUSE 100000
#define MIN_OL_PER_ORDER 5
#define MAX_OL_PER_ORDER 15
#define NU_ORDERS_PER_DISTRICT 900
#endif /* __LVAL_H

include/tpccapp.h

/* ***** */
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****

/*
 * tpccapp.h - Application Macros
 */

#ifdef __TPCCAPP_H
#define __TPCCAPP_H

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

#define daricall

#include "sqlca.h"
#include "sqlcodes.h"

#ifdef SWAP_ENDIAN
#define SWAP_BYTE(Var) SwapEndian((void*)&Var, sizeof(Var))

```

```

FUNCTION: SwapEndian
PURPOSE: Swap the byte order of a structure
EXAMPLE: int l=0x12345678; SWAP_BYTE(l); l => 0x78563412;
IMPLEMENTATION: Fold Addr in half, swap header & tail by XOR op
e.g.: *a = 0x12 [ Addr + 0];
      *b = 0x78 [ Addr + 4 - 0 - 1 = Addr+3];
      *a ^= *b; // sets *a to 0x6A
      *b ^= *a; // sets *b to 0x12
      *a ^= *b; // sets *a to 0x78

      Now *a => 0x78 && *b => 0x12
*****

void SwapEndian(void *Addr, int nb)
{
    int i;
    for (i=0; i<nb/2; i++)
    {
        char *a = (char*)Addr+i;
        char *b = (char*)Addr+(nb-i-1);

        *a ^= *b;
        *b ^= *a;
        *a ^= *b;
    }
}
#endif //SWAP_ENDIAN

/* ***** */
/* SQLCODE Macros */
/* ***** */

#define DLCHK(a) \
    if (sqlca.sqlcode == SQL_RC_E911) { goto a; }

#define NACOMPCHK(last) \
    if (sqlca.sqlcode != SQL_RC_E1339) { last = -1; } \
    else { int a = ((sqlca.sqlerrmc[4] == 0x20) ? 0 : sqlca.sqlerrmc[4]-0x30); \
            int b = ((sqlca.sqlerrmc[5] == 0x20) ? 0 : sqlca.sqlerrmc[5]-0x30); \
            if (b == 0) { last = a; } else { last = a * 10 + b; } \
    }

#ifdef __TPCCAPP_H

include/tpccdbg.h

/* ***** */
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****

/*
 * tpccdbg.h - Debugging Macros
 */

#ifdef __TPCCDBG_H
#define __TPCCDBG_H

#ifdef __cplusplus
extern "C" {

```

```

#endif

extern void sqlerror (int tranType, char *msg, char *file, int line,
    SQL_STRUCTURE sqlca *psqlca);

extern void new_debug (struct out_neword_struct *neword_ptr,
    struct in_neword_struct *in_neword_ptr,
    char *msg);
extern void pay_debug (struct out_payment_struct *payment_ptr,
    struct in_payment_struct *in_payment_ptr,
    char *msg);
extern void ord_debug (struct out_ordstat_struct *ordstat_ptr,
    struct in_ordstat_struct *in_ordstat_ptr,
    char *msg);
extern void del_debug (struct out_delivery_struct *delivery_ptr,
    struct in_delivery_struct *in_delivery_ptr,
    char *msg);
extern void stk_debug (struct out_stocklev_struct *stocklev_ptr,
    struct in_stocklev_struct *in_stocklev_ptr,
    char *msg);

extern void new_print (struct out_neword_struct *neword_ptr,
    struct in_neword_struct *in_neword_ptr,
    char *filename,
    char *msg);
extern void pay_print (struct out_payment_struct *payment_ptr,
    struct in_payment_struct *in_payment_ptr,
    char *filename,
    char *msg);
extern void ord_print (struct out_ordstat_struct *ordstat_ptr,
    struct in_ordstat_struct *in_ordstat_ptr,
    char *filename,
    char *msg);
extern void del_print (struct out_delivery_struct *delivery_ptr,
    struct in_delivery_struct *in_delivery_ptr,
    char *filename,
    char *msg);
extern void stk_print (struct out_stocklev_struct *stocklev_ptr,
    struct in_stocklev_struct *in_stocklev_ptr,
    char *filename,
    char *msg);

#ifdef __cplusplus
}
#endif

#endif // __TPCCDBG_H

```

tpccenv.sh

```

#####
#####
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use, duplication or
## disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#####
#####
#
# tpccenv.sh - UNIX Environment Setup
#
# The Kit Version

```

```

export TPCC_VERSION=CK060815

# The DB2 Instance Name (for DB2)
export DB2INSTANCE=${USER}

# The OS being used (i.e. "UNIX", "LINUX", "WINDOWS")
export PLATFORM=LINUX

# The type of make command and slash used by the OS.
# (i.e. UNIX - "/", WINDOWS - "\").
# These are referenced all over the kit.
export SLASH="/";
export MAKE=make

# Specifies whether or not to use dari stored proc's for the TPC-C driver. Set to either
DARIVERSION or NONDARI;
#export TPCC_SPTYPE=NOSP
#export TPCC_SPTYPE=SPGENERAL2
export TPCC_SPTYPE=SPGENERAL
#export TPCC_SPTYPE=DARI2SQLDA

export DB2VERSION=v8

# The schema name is typically the SQL authorization ID (or username).
# This is required for runstats and EEE.
export TPCC_SCHEMA=${USER}

# DB2 EE/EEE Configuration
export DB2EDITION=EE
#export DB2EDITION=EEE
export DB2NODE=0
export DB2NODES=1; # set to the number of nodes you have. Set to 1 for EE.

# TPCC General Configuration
export TPCC_DBNAME=TPCC
export TPCC_ROOT=${HOME}/tpc-c.ibm
export TPCC_SQLLIB=${HOME}/sqllib
export TPCC_RUNDATA=${HOME}/tpccdata

# TPCC Debug Configuration
# This is the path where all error and debug logs are placed.
# To get debugging from within the stored procedures, you must
# set DB2ENVLIST="TPCC_DEBUGDIR" in tpcc.config.
export TPCC_DEBUGDIR=/tmp

# Specifies where stored procedures should be placed and if they should
# be fenced.
export TPCC_SPDIR=${TPCC_SQLLIB}/function
export TPCC_FENCED=NO

```

10 Appendix B: Tunable Parameters

10.1. Database Parameters

db2set.cfg.out

```
DB2LINUXAIO=YES
DB2_RESOURCE_POLICY=/home/tpcc/tpc-c.ibm/aff2_eight_listener.cfg
DB2_SELUDDI_COMM_BUFFER=Y
DB2_USE_ALTERNATE_PAGE_CLEANING=YES
DB2_MAX_NON_TABLE_LOCKS=500
DB2_LGPAGE_BP=ON
DB2_TRUSTED_BINDIN=ON
DB2_KEEPTABLELOCK=ON
DB2_NO_FORK_CHECK=ON
DB2_ALLOCATION_SIZE=8388608
DB2_APM_PERFORMANCE=ALL
DB2_ENABLE_BUFPD=OFF
DB2_PINNED_BP=ON
DB2_SELECTIVITY=ON
DB2ASSUMEUPDATE=ON
DB2CHECKCLIENTINTERVAL=0
DB2_HASH_JOIN=OFF
DB2CHKSQLDA=OFF
DB2_COLLECT_TS_REC_INFO=false
DB2COMM=topip
DB2CHKPTR=OFF
```

db.cfg.out

Database Configuration for Database tpcc

```
Database configuration release level = 0x0b00
Database release level = 0x0b00

Database territory = US
Database code page = 819
Database code set = ISO8859-1
Database country/region code = 1
Database collating sequence = IDENTITY
Alternate collating sequence (ALT_COLLATE) =
Database page size = 4096

Dynamic SQL Query management (DYN_QUERY_MGMT) = DISABLE

Discovery support for this database (DISCOVER_DB) = ENABLE

Restrict access = NO
Default query optimization class (DFT_QUERYOPT) = 5
Degree of parallelism (DFT_DEGREE) = 1
Continue upon arithmetic exceptions (DFT_SQLMATHWARN) = NO
Default refresh age (DFT_REFRESH_AGE) = 0
Default maintained table values for opt (DFT_MTTB_TYPES) = SYSTEM
Number of frequent values retained (NUM_FREVALUES) = 10
Number of quantiles retained (NUM_QUANTILES) = 20

Backup pending = NO

Database is consistent = YES
Rollforward pending = NO
```

```
Restore pending = NO

Multi-page file allocation enabled = YES

Log retain for recovery status = RECOVERY
User exit for logging status = NO

Self tuning memory (SELF_TUNING_MEM) = OFF
Size of database shared memory (4KB) (DATABASE_MEMORY) = 31821824
Database memory threshold (DB_MEM_THRESH) = 10
Max storage for lock list (4KB) (LOCKLIST) = 8000
Percent. of lock lists per application (MAXLOCKS) = 100
Package cache size (4KB) (PCKCACHESZ) = 1000
Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 5000
Sort list heap (4KB) (SORTHEAP) = 16

Database heap (4KB) (DBHEAP) = 8192
Catalog cache size (4KB) (CATALOGCACHE_SZ) = (MAXAPPLS*4)
Log buffer size (4KB) (LOGBUFSZ) = 3000
Utilities heap size (4KB) (UTIL_HEAP_SZ) = 5000
Buffer pool size (pages) (BUFFPAGE) = 1000
Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 30000
Percent of mem for appl. group heap (GROUPHEAP_RATIO) = 70
Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 128

SQL statement heap (4KB) (STMTHEAP) = 16384
Default application heap (4KB) (APPLHEAPSZ) = 328
Statistics heap size (4KB) (STAT_HEAP_SZ) = 4384

Interval for checking deadlock (ms) (DLCHKTIME) = 3000
Lock timeout (sec) (LOCKTIMEOUT) = -1

Changed pages threshold (CHNGPGS_THRESH) = 99
Number of asynchronous page cleaners (NUM_IOCLEANERS) = 1
Number of I/O servers (NUM_IOSERVERS) = 1
Index sort flag (INDEXSORT) = YES
Sequential detect flag (SEQDETECT) = NO
Default prefetch size (pages) (DFT_PREFETCH_SZ) = AUTOMATIC

Track modified pages (TRACKMOD) = OFF

Default number of containers = 1
Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32

Max number of active applications (MAXAPPLS) = 660
Average number of active applications (AVG_APPLS) = 1
Max DB files open per application (MAXFILOP) = 1600

Log file size (4KB) (LOGFILSIZ) = 256000
Number of primary log files (LOGPRIMARY) = 256
Number of secondary log files (LOGSECOND) = 0
Changed path to log files (NEWLOGPATH) =
Path to log files = /dev/raw/raw1300
Overflow log path (OVERFLOWLOGPATH) =
Mirror log path (MIRRORLOGPATH) =
First active log file = S0000116.LOG
Block log on disk full (BLK_LOG_DSK_FUL) = NO
Percent max primary log space by transaction (MAX_LOG) = 0
Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0

Group commit count (MINCOMMIT) = 1
Percent log file reclaimed before soft ckcpt (SOFTMAX) = 2240
Log retain for recovery enabled (LOGRETAIN) = RECOVERY
User exit for logging enabled (USEREXIT) = OFF

HADR database role = STANDARD
HADR local host name (HADR_LOCAL_HOST) =
HADR local service name (HADR_LOCAL_SVC) =
HADR remote host name (HADR_REMOTE_HOST) =
HADR remote service name (HADR_REMOTE_SVC) =
```

```
HADR instance name of remote server (HADR_REMOTE_INST) =
HADR timeout value (HADR_TIMEOUT) = 120
HADR log write synchronization mode (HADR_SYNCMODE) = NEARSYNC
```

```
First log archive method (LOGARCHMETH1) = LOGRETAIN
Options for logarchmeth1 (LOGARCHOPT1) =
Second log archive method (LOGARCHMETH2) = OFF
Options for logarchmeth2 (LOGARCHOPT2) =
Failover log archive path (FAILARCHPATH) =
Number of log archive retries on error (NUMARCHRETRY) = 5
Log archive retry Delay (secs) (ARCHRETRYDELAY) = 20
Vendor options (VENDOROPT) =
```

```
Auto restart enabled (AUTORESTART) = ON
Index re-creation time and redo index build (INDEXREC) = SYSTEM (RESTART)
Log pages during index build (LOGINDEXBUILD) = OFF
Default number of loadrec sessions (DFT_LOADREC_SES) = 1
Number of database backups to retain (NUM_DB_BACKUPS) = 12
Recovery history retention (days) (REC_HIS_RETENTN) = 366
```

```
TSM management class (TSM_MGMTCLASS) =
TSM node name (TSM_NODENAME) =
TSM owner (TSM_OWNER) =
TSM password (TSM_PASSWORD) =
```

```
Automatic maintenance (AUTO_MAINT) = OFF
Automatic database backup (AUTO_DB_BACKUP) = OFF
Automatic table maintenance (AUTO_TBL_MAINT) = OFF
Automatic runstats (AUTO_RUNSTATS) = OFF
Automatic statistics profiling (AUTO_STATS_PROF) = OFF
Automatic profile updates (AUTO_PROF_UPD) = OFF
Automatic reorganization (AUTO_REORG) = OFF
```

dbm.cfg.out

Database Manager Configuration

Node type = Enterprise Server Edition with local and remote clients

```
Database manager configuration release level = 0x0b00
```

```
CPU speed (millisec/instruction) (CPUSPEED) = 1.889377e-07
Communications bandwidth (MB/sec) (COMM_BANDWIDTH) = 1.000000e+02
```

```
Max number of concurrently active databases (NUMDB) = 1
Federated Database System Support (FEDERATED) = NO
Transaction processor monitor name (TP_MON_NAME) =
```

```
Default charge-back account (DFT_ACCOUNT_STR) =
```

```
Java Development Kit installation path (JDK_PATH) = /home/tpcc/sqllib/java/jdk64
```

```
Diagnostic error capture level (DIAGLEVEL) = 1
Notify Level (NOTIFYLEVEL) = 1
Diagnostic data directory path (DIAGPATH) =
```

Default database monitor switches

```
Buffer pool (DFT_MON_BUFPOOL) = OFF
Lock (DFT_MON_LOCK) = OFF
Sort (DFT_MON_SORT) = OFF
Statement (DFT_MON_STMT) = OFF
Table (DFT_MON_TABLE) = OFF
Timestamp (DFT_MON_TIMESTAMP) = OFF
Unit of work (DFT_MON_UOW) = OFF
Monitor health of instance and databases (HEALTH_MON) = OFF
```

```
SYSADM group name (SYSADM_GROUP) =
SYSCTRL group name (SYSCTRL_GROUP) =
SYSMAINT group name (SYSMAINT_GROUP) =
```

SYSMON group name (SYSMON_GROUP) =

Client Userid-Password Plugin (CLNT_PW_PLUGIN) =
Client Kerberos Plugin (CLNT_KRB_PLUGIN) =
Group Plugin (GROUP_PLUGIN) =
GSS Plugin for Local Authorization (LOCAL_GSSPLUGIN) =
Server Plugin Mode (SRV_PLUGIN_MODE) = UNFENCED
Server List of GSS Plugins (SRVCON_GSSPLUGIN_LIST) =
Server Userid-Password Plugin (SRVCON_PW_PLUGIN) =
Server Connection Authentication (SRVCON_AUTH) = NOT_SPECIFIED
Database manager authentication (AUTHENTICATION) = CLIENT
Cataloging allowed without authority (CATALOG_NOAUTH) = NO
Trust all clients (TRUST_ALLCLNTS) = YES
Trusted client authentication (TRUST_CLNTAUTH) = CLIENT
Bypass federated authentication (FED_NOAUTH) = NO

Default database path (DFTDBPATH) = /home/tpcc

Database monitor heap size (4KB) (MON_HEAP_SZ) = 4096
Java Virtual Machine heap size (4KB) (JAVA_HEAP_SZ) = 1024
Audit buffer size (4KB) (AUDIT_BUF_SZ) = 0
Size of instance shared memory (4KB) (INSTANCE_MEMORY) = AUTOMATIC
Backup buffer default size (4KB) (BACKBUFSZ) = 1024
Restore buffer default size (4KB) (RESTBUFSZ) = 1024

Sort heap threshold (4KB) (SHEAPTHRES) = 0

Directory cache support (DIR_CACHE) = YES

Application support layer heap size (4KB) (ASLHEAPSZ) = 15
Max requester I/O block size (bytes) (RQRIOLBK) = 4096
Query heap size (4KB) (QUERY_HEAP_SZ) = 1000

Workload impact by throttled utilities(UTIL_IMPACT_LIM) = 10

Priority of agents (AGENTPRI) = SYSTEM
Max number of existing agents (MAXAGENTS) = 660
Agent pool size (NUM_POOLAGENTS) = 0
Initial number of agents in pool (NUM_INITAGENTS) = 0
Max number of coordinating agents (MAX_COORDAGENTS) = (MAXAGENTS -
NUM_INITAGENTS)
Max no. of concurrent coordinating agents (MAXCAGENTS) = MAX_COORDAGENTS
Max number of client connections (MAX_CONNECTIONS) = MAX_COORDAGENTS

Keep fenced process (KEEPFENCED) = YES
Number of pooled fenced processes (FENCED_POOL) = MAX_COORDAGENTS
Initial number of fenced processes (NUM_INITFENCED) = 0

Index re-creation time and redo index build (INDEXREC) = RESTART

Transaction manager database name (TM_DATABASE) = 1ST_CONN
Transaction resync interval (sec) (RESYNC_INTERVAL) = 180

SPM name (SPM_NAME) =
SPM log size (SPM_LOG_FILE_SZ) = 256
SPM resync agent limit (SPM_MAX_RESYNC) = 20
SPM log path (SPM_LOG_PATH) =

TCP/IP Service name (SVCENAME) =
Discovery mode (DISCOVER) = SEARCH
Discover server instance (DISCOVER_INST) = ENABLE

Maximum query degree of parallelism (MAX_QUERYDEGREE) = ANY
Enable intra-partition parallelism (INTRA_PARALLEL) = NO

Maximum Asynchronous TQs per query (FEDERATED_ASYNC) = 0

No. of int. communication buffers(4KB)(FCM_NUM_BUFFERS) = AUTOMATIC
No. of int. communication channels (FCM_NUM_CHANNELS) = AUTOMATIC
Node connection elapse time (sec) (CONN_ELAPSE) = 10

Max number of node connection retries (MAX_CONNRETRIES) = 5
Max time difference between nodes (min) (MAX_TIME_DIFF) = 60

db2start/db2stop timeout (min) (START_STOP_TIME) = 10

aff2_eight_listener.cfg

```
<!-- This policy is valid -->
<RESOURCE_POLICY>
  <DATABASE_RESOURCE_POLICY>
    <DBNAME>tpcc</DBNAME>
    <METHOD>NODEMASK</METHOD>

  <RESOURCE_BINDING>
    <RESOURCE>0</RESOURCE>
    <DBMEM_PERCENTAGE>49.5</DBMEM_PERCENTAGE>

  <SERVICE_NAME>50021</SERVICE_NAME>
  <SERVICE_NAME>50022</SERVICE_NAME>
  <SERVICE_NAME>50023</SERVICE_NAME>
  <SERVICE_NAME>50024</SERVICE_NAME>
  <BUFFERPOOL_BINDING>
    <NUM_CLEANERS>1</NUM_CLEANERS>
    <BUFFERPOOL_ID>5</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>13</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>21</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>29</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>37</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>45</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>53</BUFFERPOOL_ID>
  </BUFFERPOOL_BINDING>
  <BUFFERPOOL_BINDING>
    <NUM_CLEANERS>1</NUM_CLEANERS>
    <BUFFERPOOL_ID>6</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>14</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>22</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>30</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>38</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>46</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>54</BUFFERPOOL_ID>
  </BUFFERPOOL_BINDING>
  <BUFFERPOOL_BINDING>
    <NUM_CLEANERS>1</NUM_CLEANERS>
    <BUFFERPOOL_ID>7</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>15</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>23</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>31</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>39</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>47</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>55</BUFFERPOOL_ID>
  </BUFFERPOOL_BINDING>
  <BUFFERPOOL_BINDING>
    <NUM_CLEANERS>1</NUM_CLEANERS>
    <BUFFERPOOL_ID>8</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>16</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>24</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>32</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>40</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>48</BUFFERPOOL_ID>
    <BUFFERPOOL_ID>56</BUFFERPOOL_ID>
  </BUFFERPOOL_BINDING>
</RESOURCE_BINDING>

<RESOURCE_BINDING>
  <RESOURCE>1</RESOURCE>
  <DBMEM_PERCENTAGE>49.5</DBMEM_PERCENTAGE>

  <SERVICE_NAME>50025</SERVICE_NAME>
  <SERVICE_NAME>50026</SERVICE_NAME>
  <SERVICE_NAME>50027</SERVICE_NAME>
```

```
<SERVICE_NAME>50028</SERVICE_NAME>
<BUFFERPOOL_BINDING>
  <NUM_CLEANERS>1</NUM_CLEANERS>
  <BUFFERPOOL_ID>9</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>17</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>25</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>33</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>41</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>49</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>57</BUFFERPOOL_ID>
</BUFFERPOOL_BINDING>
<BUFFERPOOL_BINDING>
  <NUM_CLEANERS>1</NUM_CLEANERS>
  <BUFFERPOOL_ID>10</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>18</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>26</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>34</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>42</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>50</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>58</BUFFERPOOL_ID>
</BUFFERPOOL_BINDING>
<BUFFERPOOL_BINDING>
  <NUM_CLEANERS>1</NUM_CLEANERS>
  <BUFFERPOOL_ID>11</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>19</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>27</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>35</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>43</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>51</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>59</BUFFERPOOL_ID>
</BUFFERPOOL_BINDING>
<BUFFERPOOL_BINDING>
  <NUM_CLEANERS>1</NUM_CLEANERS>
  <BUFFERPOOL_ID>12</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>20</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>28</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>36</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>44</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>52</BUFFERPOOL_ID>
  <BUFFERPOOL_ID>60</BUFFERPOOL_ID>
</BUFFERPOOL_BINDING>
</RESOURCE_BINDING>
```

```
</DATABASE_RESOURCE_POLICY>
</RESOURCE_POLICY>
```

10.2. Transaction Monitor Parameters

inetInfo registry.reg

Windows Registry Editor Version 5.00

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\inetInfo\Parameters]
"ListenBackLog"=dword:000000fa
"DispatchEntries"=hex(7):4c,00,44,00,41,00,50,00,53,00,56,00,43,00,00,00,00,00
"MaxConnections"=dword:000061a8
"PoolThreadLimit"=dword:000000fa
"ThreadTimeout"=dword:00015180
"MaxConcurrency"=dword:ffffff
```

tcipip parameters registry.reg

Windows Registry Editor Version 5.00

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcipip\Parameters]
"NV Hostname"="client16"
```

"DataBasePath"=hex(2):25,00,53,00,79,00,73,00,74,00,65,00,6d,00,52,00,6f,00,6f,\n00,74,00,25,00,5c,00,53,00,79,00,73,00,74,00,65,00,6d,00,33,00,32,00,5c,00,\n64,00,72,00,69,00,76,00,65,00,72,00,73,00,5c,00,65,00,74,00,63,00,00,00\n"NameServer"=""\n"ForwardBroadcasts"=dword:00000000\n"IPEnableRouter"=dword:00000000\n"Domain"=""\n"Hostname"="client16"\n"SearchList"=""\n"UseDomainNameDevolution"=dword:00000001\n"EnableCMPRedirect"=dword:00000001\n"DeadGWDetectDefault"=dword:00000001\n"DontAddDefaultGatewayDefault"=dword:00000000\n"EnableSecurityFilters"=dword:00000000\n"AllowUnqualifiedQuery"=dword:00000000\n"PrioritizeRecordData"=dword:00000001\n"GlobalTcpWindowSize"=dword:00040000

[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters\\Adapters]

[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters\\Adapters\\NdisWANip]\n"LLInterface"="WANARP"\n"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\n6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\n00,61,00,63,00,65,00,73,00,5c,00,7b,00,36,00,31,00,44,00,33,00,33,00,39,00,\n37,00,37,00,2d,00,37,00,31,00,30,00,46,00,2d,00,34,00,41,00,37,00,45,00,2d,\n00,38,00,44,00,34,00,37,00,2d,00,34,00,42,00,34,00,38,00,32,00,42,00,43,00,\n35,00,32,00,33,00,46,00,38,00,7d,00,00,00,54,00,63,00,70,00,69,00,70,00,5c,\n00,50,00,61,00,72,00,61,00,6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,\n6e,00,74,00,65,00,72,00,66,00,61,00,63,00,65,00,73,00,5c,00,7b,00,39,00,41,\n00,33,00,41,00,41,00,36,00,41,00,43,00,2d,00,30,00,38,00,34,00,36,00,2d,00,\n34,00,30,00,37,00,45,00,2d,00,38,00,32,00,35,00,30,00,2d,00,46,00,30,00,33,\n00,42,00,36,00,30,00,34,00,39,00,36,00,36,00,44,00,43,00,7d,00,00,00,00,00\n"NumInterfaces"=dword:00000002\n"IpInterfaces"=hex:77,39,d3,61,0f,71,7e,4a,8d,47,4b,48,2b,c5,23,f8,ac,a6,3a,9a,\n46,58,7e,40,82,50,f0,3b,60,49,66,dc

[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters\\Adapters\\{0435C97F-9186-473F-B181-5449A2CF0042}]\n"LLInterface"=""\n"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\n6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\n00,61,00,63,00,65,00,73,00,5c,00,7b,00,30,00,34,00,33,00,35,00,43,00,39,00,\n37,00,46,00,2d,00,39,00,31,00,38,00,36,00,2d,00,34,00,37,00,33,00,46,00,2d,\n00,42,00,31,00,38,00,31,00,2d,00,35,00,34,00,34,00,39,00,41,00,32,00,43,00,\n46,00,30,00,30,00,34,00,32,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters\\Adapters\\{1E07A95A-92A0-4836-BF73-7AE38F8ACA07}]\n"LLInterface"=""\n"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\n6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\n00,61,00,63,00,65,00,73,00,5c,00,7b,00,31,00,45,00,30,00,37,00,41,00,39,00,\n35,00,41,00,2d,00,39,00,32,00,41,00,30,00,2d,00,34,00,38,00,33,00,36,00,2d,\n00,42,00,46,00,37,00,33,00,2d,00,37,00,41,00,45,00,33,00,38,00,46,00,38,00,\n41,00,43,00,41,00,30,00,37,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters\\Adapters\\{2EA04AA5-93A6-437F-9153-2F6834D3B795}]\n"LLInterface"=""\n"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\n6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\n00,61,00,63,00,65,00,73,00,5c,00,7b,00,32,00,45,00,41,00,30,00,34,00,41,00,\n41,00,35,00,2d,00,39,00,33,00,41,00,36,00,2d,00,34,00,33,00,37,00,46,00,2d,\n00,39,00,31,00,35,00,33,00,2d,00,32,00,46,00,36,00,38,00,33,00,34,00,44,00,\n33,00,42,00,37,00,39,00,35,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters\\Adapters\\{7B215199-A3F3-4836-89A6-390C5E70E801}]\n"LLInterface"=""\n"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\n6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\n00,61,00,63,00,65,00,73,00,5c,00,7b,00,37,00,42,00,32,00,31,00,35,00,31,00,\n39,00,39,00,2d,00,41,00,33,00,46,00,33,00,2d,00,34,00,38,00,33,00,36,00,2d,\n00,38,00,39,00,41,00,36,00,2d,00,33,00,39,00,30,00,43,00,35,00,45,00,37,00,\n30,00,45,00,38,00,30,00,31,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters\\Adapters\\{A32BB4A3-C9B2-4ADB-A65D-18BB314BF7F0}]\n"LLInterface"=""\n"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\n6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\n00,61,00,63,00,65,00,73,00,5c,00,7b,00,41,00,33,00,32,00,42,00,42,00,34,00,\n41,00,33,00,2d,00,43,00,39,00,42,00,32,00,2d,00,34,00,41,00,44,00,42,00,2d,\n00,41,00,36,00,35,00,44,00,2d,00,31,00,38,00,42,00,42,00,33,00,31,00,34,00,\n42,00,46,00,37,00,46,00,30,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters\\Adapters\\{A71EB7B5-37C6-42DB-BE8F-BB231FD1BE00}]\n"LLInterface"=""\n"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\n6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\n00,61,00,63,00,65,00,73,00,5c,00,7b,00,41,00,37,00,31,00,45,00,42,00,37,00,\n42,00,35,00,2d,00,33,00,37,00,43,00,36,00,2d,00,34,00,32,00,44,00,42,00,2d,\n00,42,00,45,00,38,00,46,00,2d,00,42,00,32,00,33,00,31,00,46,00,44,00,\n31,00,42,00,45,00,30,00,30,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters\\Adapters\\{BEABCC14-9C0A-4BE9-9817-14C4092418D3}]\n"LLInterface"=""\n"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\n6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\n00,61,00,63,00,65,00,73,00,5c,00,7b,00,42,00,45,00,41,00,42,00,43,00,43,00,\n31,00,34,00,2d,00,39,00,43,00,30,00,41,00,2d,00,34,00,42,00,45,00,39,00,2d,\n00,39,00,38,00,31,00,37,00,2d,00,31,00,34,00,43,00,34,00,30,00,39,00,32,00,\n34,00,31,00,38,00,44,00,33,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters\\Adapters\\{CD3F7746-9E60-4E22-9A40-7BC6CC6B2E2E}]\n"LLInterface"=""\n"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\n6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\n00,61,00,63,00,65,00,73,00,5c,00,7b,00,43,00,44,00,33,00,46,00,37,00,37,00,\n34,00,36,00,2d,00,39,00,45,00,36,00,30,00,2d,00,34,00,45,00,32,00,32,00,2d,\n00,39,00,41,00,34,00,30,00,2d,00,37,00,42,00,43,00,36,00,43,00,43,00,36,00,\n42,00,32,00,45,00,32,00,45,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters\\DNSReregisteredAdapters]

[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters\\Interfaces]

[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters\\Interfaces\\{0435C97F-9186-473F-B181-5449A2CF0042}]\n"UseZeroBroadcast"=dword:00000000\n"EnableDeadGWDetect"=dword:00000001\n"EnableDHCP"=dword:00000000\n"IPAddress"=hex(7):31,00,33,00,35,00,2e,00,31,00,2e,00,31,00,2e,00,31,00,00,00,\n00,00\n"SubnetMask"=hex(7):32,00,35,00,35,00,2e,00,32,00,35,00,35,00,2e,00,32,00,35,\n00,35,00,2e,00,30,00,00,00,00,00\n"DefaultGateway"=hex(7):00,00\n"DefaultGatewayMetric"=hex(7):00,00\n"NameServer"=""\n"Domain"=""\n"DisableDynamicUpdate"=dword:00000000

"EnableAdapterDomainNameRegistration"=dword:00000000\n"InterfaceMetric"=dword:00000001\n"TCPAllowedPorts"=hex(7):30,00,00,00,00,00,00\n"UDPAllowedPorts"=hex(7):30,00,00,00,00,00,00\n"RawIPAllowedProtocols"=hex(7):30,00,00,00,00,00,00\n"NTContextList"=hex(7):30,00,78,00,30,00,30,00,30,00,30,00,30,00,30,00,30,00,\n34,00,00,00,00,00\n"TcpWindowSize"=dword:00008000

[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters\\Interfaces\\{1E07A95A-92A0-4836-BF73-7AE38F8ACA07}]\n"UseZeroBroadcast"=dword:00000000\n"EnableDeadGWDetect"=dword:00000001\n"EnableDHCP"=dword:00000001\n"IPAddress"=hex(7):30,00,2e,00,30,00,2e,00,30,00,2e,00,30,00,00,00,00,00\n"SubnetMask"=hex(7):30,00,2e,00,30,00,2e,00,30,00,2e,00,30,00,00,00,00,00\n"DefaultGateway"=hex(7):00,00\n"DefaultGatewayMetric"=hex(7):00,00\n"NameServer"=""\n"Domain"=""\n"DisableDynamicUpdate"=dword:00000000\n"EnableAdapterDomainNameRegistration"=dword:00000000\n"InterfaceMetric"=dword:00000001\n"TCPAllowedPorts"=hex(7):30,00,00,00,00,00,00\n"UDPAllowedPorts"=hex(7):30,00,00,00,00,00,00\n"RawIPAllowedProtocols"=hex(7):30,00,00,00,00,00,00\n"NTContextList"=hex(7):00,00\n"DhcpIPAddress"="0.0.0.0"\n"DhcpSubnetMask"="255.0.0.0"

[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters\\Interfaces\\{2EA04AA5-93A6-437F-9153-2F6834D3B795}]\n"UseZeroBroadcast"=dword:00000000\n"EnableDeadGWDetect"=dword:00000001\n"EnableDHCP"=dword:00000000\n"IPAddress"=hex(7):31,00,39,00,32,00,2e,00,31,00,36,00,38,00,2e,00,31,00,32,00,\n2e,00,31,00,36,00,00,00,00,00\n"SubnetMask"=hex(7):32,00,35,00,35,00,2e,00,32,00,35,00,35,00,2e,00,32,00,35,\n00,35,00,2e,00,30,00,00,00,00,00\n"DefaultGateway"=hex(7):00,00\n"DefaultGatewayMetric"=hex(7):00,00\n"NameServer"=""\n"Domain"=""\n"DisableDynamicUpdate"=dword:00000000\n"EnableAdapterDomainNameRegistration"=dword:00000000\n"InterfaceMetric"=dword:00000001\n"TCPAllowedPorts"=hex(7):30,00,00,00,00,00,00\n"UDPAllowedPorts"=hex(7):30,00,00,00,00,00,00\n"RawIPAllowedProtocols"=hex(7):30,00,00,00,00,00,00\n"NTContextList"=hex(7):30,00,78,00,30,00,30,00,30,00,30,00,30,00,30,00,30,00,\n33,00,00,00,00,00\n"DhcpServer"="255.255.255.255"\n"Lease"=dword:00000e10\n"LeaseObtainedTime"=dword:40b39c06\n"1"=dword:40b3a30e\n"2"=dword:40b3a854\n"LeaseTerminatesTime"=dword:40b3aa16\n"IPAutoconfigurationAddress"="0.0.0.0"\n"IPAutoconfigurationMask"="255.255.0.0"\n"IPAutoconfigurationSeed"=dword:00000000\n"AddressType"=dword:00000000\n"DhcpClassidBin"=hex:\n"TcpWindowSize"=dword:00008000

[HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters\\Interfaces\\{61D33977-710F-4A7E-8D47-4B482BC523F8}]\n"UseZeroBroadcast"=dword:00000000\n"EnableDHCP"=dword:00000000\n"IPAddress"=hex(7):30,00,2e,00,30,00,2e,00,30,00,2e,00,30,00,00,00,00,00\n"SubnetMask"=hex(7):30,00,2e,00,30,00,2e,00,30,00,2e,00,30,00,00,00,00,00


```
"DefaultGateway"=hex(7):00,00
"EnableDeadGWDetect"=dword:00000001
"DontAddDefaultGateway"=dword:00000000
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfa
ces\{7B215199-A3F3-4836-89A6-390C5E70E801}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000001
"IPAddress"=hex(7):30,00,2e,00,30,00,2e,00,30,00,2e,00,30,00,00,00,00,00
"SubnetMask"=hex(7):30,00,2e,00,30,00,2e,00,30,00,2e,00,30,00,00,00,00,00
"DefaultGateway"=hex(7):00,00
"DefaultGatewayMetric"=hex(7):00,00
"NameServer"=""
"Domain"=""
"DisableDynamicUpdate"=dword:00000000
"EnableAdapterDomainNameRegistration"=dword:00000000
"InterfaceMetric"=dword:00000001
"TCPAllowedPorts"=hex(7):30,00,00,00,00,00,00
"UDPAllowedPorts"=hex(7):30,00,00,00,00,00
"RawIPAllowedProtocols"=hex(7):30,00,00,00,00,00
"NTEContextList"=hex(7):00,00
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfa
ces\{9A3AA6AC-5846-407E-8250-F03B604966DC}]
"UseZeroBroadcast"=dword:00000000
"EnableDHCP"=dword:00000000
"IPAddress"=hex(7):30,00,2e,00,30,00,2e,00,30,00,2e,00,30,00,00,00,00,00
"SubnetMask"=hex(7):30,00,2e,00,30,00,2e,00,30,00,2e,00,30,00,00,00,00,00
"DefaultGateway"=hex(7):00,00
"EnableDeadGWDetect"=dword:00000001
"DontAddDefaultGateway"=dword:00000000
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfa
ces\{A32BB4A3-C9B2-4ADB-A65D-18BB314BF7F0}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000000
"IPAddress"=hex(7):31,00,30,00,2e,00,31,00,2e,00,33,00,2e,00,31,00,00,00,00,00
"SubnetMask"=hex(7):32,00,35,00,35,00,2e,00,32,00,35,00,35,00,2e,00,32,00,35,\
00,35,00,2e,00,30,00,00,00,00,00
"DefaultGateway"=hex(7):00,00
"DefaultGatewayMetric"=hex(7):00,00
"NameServer"=""
"Domain"=""
"DisableDynamicUpdate"=dword:00000000
"EnableAdapterDomainNameRegistration"=dword:00000000
"InterfaceMetric"=dword:00000001
"TCPAllowedPorts"=hex(7):30,00,00,00,00,00,00
"UDPAllowedPorts"=hex(7):30,00,00,00,00,00
"RawIPAllowedProtocols"=hex(7):30,00,00,00,00,00
"NTEContextList"=hex(7):30,00,78,00,30,00,30,00,30,00,30,00,30,00,30,00,\
35,00,00,00,00,00
"TcpWindowSize"=dword:00040000
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfa
ces\{A71EB7B5-37C6-42DB-BE8F-BB231FD1BE00}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000000
"IPAddress"=hex(7):31,00,30,00,2e,00,31,00,36,00,2e,00,31,00,2e,00,32,00,00,00,\
00,00
"SubnetMask"=hex(7):32,00,35,00,35,00,2e,00,32,00,35,00,35,00,2e,00,32,00,35,\
00,35,00,2e,00,30,00,00,00,00,00
"DefaultGateway"=hex(7):00,00
"DefaultGatewayMetric"=hex(7):00,00
"NameServer"=""
"Domain"=""
"DisableDynamicUpdate"=dword:00000000
"EnableAdapterDomainNameRegistration"=dword:00000000
```

```
"InterfaceMetric"=dword:00000001
"TCPAllowedPorts"=hex(7):30,00,00,00,00,00
"UDPAllowedPorts"=hex(7):30,00,00,00,00,00
"RawIPAllowedProtocols"=hex(7):30,00,00,00,00,00
"NTEContextList"=hex(7):30,00,78,00,30,00,30,00,30,00,30,00,30,00,30,00,\
32,00,00,00,00,00
"DhcpServer"="255.255.255.255"
"Lease"=dword:00000e10
"LeaseObtainedTime"=dword:40be0640
"T1"=dword:40be0d48
"T2"=dword:40be128e
"LeaseTerminatesTime"=dword:40be1450
"IPAutoconfigurationAddress"="0.0.0.0"
"IPAutoconfigurationMask"="255.255.0.0"
"IPAutoconfigurationSeed"=dword:00000000
"AddressType"=dword:00000000
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfa
ces\{BEABCC14-9C0A-4BE9-9817-14C4092418D3}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000000
"IPAddress"=hex(7):31,00,30,00,2e,00,31,00,2e,00,32,00,2e,00,31,00,00,00,00,00
"SubnetMask"=hex(7):32,00,35,00,35,00,2e,00,32,00,35,00,35,00,2e,00,32,00,35,\
00,35,00,2e,00,30,00,00,00,00,00
"DefaultGateway"=hex(7):00,00
"DefaultGatewayMetric"=hex(7):00,00
"NameServer"=""
"Domain"=""
"DisableDynamicUpdate"=dword:00000000
"EnableAdapterDomainNameRegistration"=dword:00000000
"InterfaceMetric"=dword:00000001
"TCPAllowedPorts"=hex(7):30,00,00,00,00,00
"UDPAllowedPorts"=hex(7):30,00,00,00,00,00
"RawIPAllowedProtocols"=hex(7):30,00,00,00,00,00
"NTEContextList"=hex(7):30,00,78,00,30,00,30,00,30,00,30,00,30,00,30,00,\
36,00,00,00,00,00
"TcpWindowSize"=dword:00008000
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfa
ces\{CD3F7746-9E60-4E22-9A40-7BC6CC6B2E2E}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000001
"IPAddress"=hex(7):30,00,2e,00,30,00,2e,00,30,00,2e,00,30,00,00,00,00,00
"SubnetMask"=hex(7):30,00,2e,00,30,00,2e,00,30,00,2e,00,30,00,00,00,00,00
"DefaultGateway"=hex(7):00,00
"DefaultGatewayMetric"=hex(7):00,00
"NameServer"=""
"Domain"=""
"DisableDynamicUpdate"=dword:00000000
"EnableAdapterDomainNameRegistration"=dword:00000000
"InterfaceMetric"=dword:00000001
"TCPAllowedPorts"=hex(7):30,00,00,00,00,00
"UDPAllowedPorts"=hex(7):30,00,00,00,00,00
"RawIPAllowedProtocols"=hex(7):30,00,00,00,00,00
"NTEContextList"=hex(7):30,00,78,00,30,00,30,00,30,00,30,00,30,00,30,00,\
37,00,00,00,00,00
"TcpWindowSize"=dword:00008000
"DhcpIPAddress"="0.0.0.0"
"DhcpSubnetMask"="255.0.0.0"
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Persist
entRoutes]
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\WInso
ck]
"UseDelayedAcceptance"=dword:00000000
"HelperDllName"=hex(2):25,00,53,00,79,00,73,00,74,00,65,00,6d,00,52,00,6f,00,\
6f,00,74,00,25,00,5c,00,53,00,79,00,73,00,74,00,65,00,6d,00,33,00,32,00,5c,\
```

```
00,77,00,73,00,68,00,74,00,63,00,70,00,69,00,70,00,2e,00,64,00,6c,00,6c,00,\
00,00
"MaxSockAddrLength"=dword:00000010
"MinSockAddrLength"=dword:00000010
"Mapping"=hex:0b,00,00,00,03,00,00,00,02,00,00,01,00,00,00,06,00,00,00,02,\
00,00,00,01,00,00,00,00,00,02,00,00,00,00,00,00,00,06,00,00,00,00,00,\
00,00,00,00,00,06,00,00,00,00,00,00,01,00,00,00,06,00,00,00,02,00,00,\
00,02,00,00,11,00,00,00,02,00,00,00,02,00,00,00,00,00,00,00,02,00,00,\
00,00,00,11,00,00,00,00,00,00,00,00,00,11,00,00,00,00,00,00,02,\
00,00,00,11,00,00,00,02,00,00,00,03,00,00,00,00,00,00
```

tpccCom.tpcc.com settingss.txt.txt

Windows Registry Editor Version 5.00

```
[HKEY_LOCAL_MACHINE\SOFTWARE\TPCC]
"dvlyLogPath"="C:\inetpub\wwwroot\tpcc\dlvly"
"dvlyQueueLen"=dword:00004e20
"nullDB"=dword:00000000
"htmlTrace"=dword:00000000
"dbName"="tpcc"
"errorLogFile"="c:\inetpub\wwwroot\tpcc\isapi_err.log"
"htmlTraceLogFile"="c:\inetpub\wwwroot\tpcc\isapi.log"
"numUsers"=dword:00005208
"dbType"="DB2"
"dbUserName"="tpcc"
"dbPassword"="tpcc"
"dbInterfacePath"="C:\inetpub\wwwroot\tpcc\tpccdb2glue.dll"
"dvlyQueueThreshold"=dword:0000000a
"dvlyThreads"=dword:00000006
"dynamicDvly"=dword:00000000
"isapi_trace"=dword:00000000
"numWarehouse"=dword:000007d0
"numPools"=dword:00000001
"numServers"=dword:00000001
```

tpcc software registry.reg

```
Transactions not supported
Enable Object pooling
Minimum pool size 32
Maximum pool size 32
Creation timeout 100,000
Enable Object Construction
Enable Just in time activation
Concurrency Required
```

10.3. Linux Parameters

alloc hugepages.sh

```
#!/bin/bash
```

```
echo 62168 > /proc/sys/vm/nr_hugepages
```

cmdline.txt

```
root=/dev/sda2 resume=/dev/sda1 selinux=0 audit=0 console=ttyS0,115200 elevator=noop
splash=silent showopts migration_cost=-100000,1610,13327,24669
```

modprobe.conf.local

```
#
```

```
# please add local extensions to this file
#
options e1000 InterruptThrottleRate=400,400,400
```

chrt-ififo2.sh

```
#!/bin/ksh

for i in `ps -fu tpcc | grep db2loggl | awk '{print $2}'`; do
    chrt -f -p 97 $i
done

for j in `ps -fu tpcc | grep db2ag | awk '{print $2}'`; do
    chrt -f -p 96 $j
done

for k in `ps -fu tpcc | grep db2pclr | awk '{print $2}'`; do
    chrt -f -p 96 $k
done
```

zio.sh

```
echo 1000 > /sys/class/scsi_host/host15/zio_timer
echo 1000 > /sys/class/scsi_host/host16/zio_timer
echo 1000 > /sys/class/scsi_host/host17/zio_timer
echo 1000 > /sys/class/scsi_host/host18/zio_timer
echo 1000 > /sys/class/scsi_host/host19/zio_timer
echo 1000 > /sys/class/scsi_host/host20/zio_timer
echo 1000 > /sys/class/scsi_host/host21/zio_timer
echo 1000 > /sys/class/scsi_host/host22/zio_timer
echo 1000 > /sys/class/scsi_host/host23/zio_timer
echo 1000 > /sys/class/scsi_host/host24/zio_timer
echo 1000 > /sys/class/scsi_host/host25/zio_timer
echo 1000 > /sys/class/scsi_host/host26/zio_timer
```

```
echo 1 > /sys/class/scsi_host/host15/zio
echo 1 > /sys/class/scsi_host/host16/zio
echo 1 > /sys/class/scsi_host/host17/zio
echo 1 > /sys/class/scsi_host/host18/zio
echo 1 > /sys/class/scsi_host/host19/zio
echo 1 > /sys/class/scsi_host/host20/zio
echo 1 > /sys/class/scsi_host/host21/zio
echo 1 > /sys/class/scsi_host/host22/zio
echo 1 > /sys/class/scsi_host/host23/zio
echo 1 > /sys/class/scsi_host/host24/zio
echo 1 > /sys/class/scsi_host/host25/zio
echo 1 > /sys/class/scsi_host/host26/zio
```

aff1.sh

```
#!/bin/ksh

to_hex() {
    if (($# == 0)); then
        print "usage: to_hex <int> [<size>]"
        return
    fi

    let int="${1}"
    let size="${2:-0}" # length of output hex string.

    set -A digit 0 1 2 3 4 5 6 7 8 9 a b c d e f
    hex=""

    while ((int > 0)); do
        hex="${digit[${int % 16}]}${hex}"
        int=$((int / 16))
    done
}
```

```
done

if ((size != 0)); then
    typeset -Z${size} zeros=0
    typeset -R${size} hex="${zeros}${hex}"
fi

# print "${hex}"
}

for i in `ps -fu tpcc | grep db2ipccm | awk '{print $2}'`; do
    cmd="taskset -p 0x0001 $i"
    $cmd
done

j=3

for i in `ps -fu tpcc | grep db2tccpm | awk '{print $2}'`; do
    to_hex $i
    cmd="taskset -p 0x${hex} $i"
    $cmd
    j=$((j+4));
done
```

aff2.sh

```
#!/bin/ksh

to_hex() {
    if (($# == 0)); then
        print "usage: to_hex <int> [<size>]"
        return
    fi

    let int="${1}"
    let size="${2:-0}" # length of output hex string.

    set -A digit 0 1 2 3 4 5 6 7 8 9 a b c d e f
    hex=""

    while ((int > 0)); do
        hex="${digit[${int % 16}]}${hex}"
        int=$((int / 16))
    done

    if ((size != 0)); then
        typeset -Z${size} zeros=0
        typeset -R${size} hex="${zeros}${hex}"
    fi

    # print "${hex}"
}

for i in `ps -fu tpcc | grep db2loggl | awk '{print $2}'`; do
    # cmd="taskset -p 0xc000 $i"
    cmd="taskset -p 0xf00 $i"
    $cmd
done

j=49152
two=0

for i in `ps -fu tpcc | grep db2pclr | awk '{print $2}'`; do
    to_hex $i
    cmd="taskset -p 0x${hex} $i"
    $cmd
    j=$((j+4));
done
```

affinitize_irqs.sh

```
#!/bin/bash

# chassis 1
# qllogic ports
echo 00000000,00000000,00000000,00000002 > /proc/irq/177/smp_affinity
echo 00000000,00000000,00000000,00000004 > /proc/irq/185/smp_affinity
echo 00000000,00000000,00000000,00000008 > /proc/irq/193/smp_affinity
echo 00000000,00000000,00000000,00000010 > /proc/irq/201/smp_affinity
echo 00000000,00000000,00000000,00000020 > /proc/irq/209/smp_affinity
echo 00000000,00000000,00000000,00000040 > /proc/irq/217/smp_affinity

# chassis 2
# qllogic ports
echo 00000000,00000000,00000000,00000200 > /proc/irq/225/smp_affinity
echo 00000000,00000000,00000000,00000400 > /proc/irq/233/smp_affinity
echo 00000000,00000000,00000000,00000800 > /proc/irq/50/smp_affinity
echo 00000000,00000000,00000000,00001000 > /proc/irq/58/smp_affinity
echo 00000000,00000000,00000000,00002000 > /proc/irq/66/smp_affinity
echo 00000000,00000000,00000000,00002000 > /proc/irq/74/smp_affinity

# log qllogic port
echo 00000000,00000000,00000000,00004000 > /proc/irq/82/smp_affinity
```

/home/tpcc/affirq.sh

affirq.sh

```
#!/bin/ksh

# 192.168.11 network
device=$(cat /proc/interrupts | grep eth13 | awk -F : '{print $1+$0}')
print "eth13: echo 00000000,00000000,00000000,00000080 >
/proc/irq/$device/smp_affinity"
echo "00000000,00000000,00000000,00000080" > /proc/irq/$device/smp_affinity
cat /proc/irq/$device/smp_affinity

# 192.168.12 network
device=$(cat /proc/interrupts | grep eth14 | awk -F : '{print $1+$0}')
print "eth14: echo 00000000,00000000,00000000,00000100 >
/proc/irq/$device/smp_affinity"
echo "00000000,00000000,00000000,00000100" > /proc/irq/$device/smp_affinity
cat /proc/irq/$device/smp_affinity
```

doit

```
#!/bin/sh

# tune slab cache
/home/tpcc/tune_slab.sh

echo "affinitizing irq"
/home/tpcc/tpc-c.ibm/affinitize_irqs.sh

sysctl -p
modprobe raw
sleep 5
/home/tpcc/tpc-c.ibm/setraw.sh
chmod -R 777 /dev/raw/raw*

modprobe capability disable=1
modprobe capability enable=1

umount /sys/kernel/debug
umount /sys/kernel/security
```

tune_slab.sh

```
#!/bin/bash
```

```
echo "blkdev_requests 336 168 8" > /proc/slabinfo
echo "size-1024 216 108 8" > /proc/slabinfo
echo "scsi_cmd_cache 336 168 8" > /proc/slabinfo
echo "size-4096 168 84 8" > /proc/slabinfo
echo "sgpool-8 240 120 8" > /proc/slabinfo
echo "bio 240 120 8" > /proc/slabinfo
echo "kiocb 240 120 8" > /proc/slabinfo
echo "biovec-1 240 120 8" > /proc/slabinfo
```

/etc/sysctl.conf

```
# Disable response to broadcasts.
# You don't want yourself becoming a Smurf amplifier.
net.ipv4.icmp_echo_ignore_broadcasts = 1
# enable route verification on all interfaces
net.ipv4.conf.all.rp_filter = 1
kernel.sem = 250 256000 32 1024
kernel.msgmni = 2048
kernel.shmmax = 1073741824
kernel.shmall = 137438953472
fs.file-max = 524288
net.core.rmem_max = 131071
net.core.wmem_max = 131071
vm.hugetlb_shm_group = 102
```

setraw.sh

```
modprobe raw
raw /dev/raw/raw1 /dev/sdb1
raw /dev/raw/raw2 /dev/sdc1
raw /dev/raw/raw3 /dev/sdd1
raw /dev/raw/raw4 /dev/sde1
raw /dev/raw/raw5 /dev/sdf1
raw /dev/raw/raw6 /dev/sdg1
raw /dev/raw/raw7 /dev/sdh1
raw /dev/raw/raw8 /dev/sdi1
raw /dev/raw/raw9 /dev/sdj1
raw /dev/raw/raw10 /dev/sdk1
raw /dev/raw/raw11 /dev/sdl1
raw /dev/raw/raw12 /dev/sdm1
raw /dev/raw/raw13 /dev/sdn1
raw /dev/raw/raw14 /dev/sdo1
raw /dev/raw/raw15 /dev/sdp1
raw /dev/raw/raw16 /dev/sdq1
raw /dev/raw/raw17 /dev/sdr1
raw /dev/raw/raw18 /dev/sds1
raw /dev/raw/raw19 /dev/sdt1
raw /dev/raw/raw20 /dev/sdu1
raw /dev/raw/raw21 /dev/sdv1
raw /dev/raw/raw22 /dev/sdw1
raw /dev/raw/raw23 /dev/sdx1
raw /dev/raw/raw24 /dev/sdy1
raw /dev/raw/raw25 /dev/sdz1
raw /dev/raw/raw26 /dev/sdaa1
raw /dev/raw/raw27 /dev/sdab1
raw /dev/raw/raw28 /dev/sdac1
raw /dev/raw/raw29 /dev/sdad1
raw /dev/raw/raw30 /dev/sdae1
raw /dev/raw/raw31 /dev/sdaf1
raw /dev/raw/raw32 /dev/sdag1
raw /dev/raw/raw33 /dev/sdah1
raw /dev/raw/raw34 /dev/sdai1
```

```
raw /dev/raw/raw35 /dev/sdaj1
raw /dev/raw/raw36 /dev/sdak1
raw /dev/raw/raw37 /dev/sdal1
raw /dev/raw/raw38 /dev/sdam1
raw /dev/raw/raw39 /dev/sdan1
raw /dev/raw/raw40 /dev/sdao1
raw /dev/raw/raw41 /dev/sdap1
raw /dev/raw/raw42 /dev/sdaq1
raw /dev/raw/raw43 /dev/sdar1
raw /dev/raw/raw44 /dev/sdas1
raw /dev/raw/raw45 /dev/sdat1
raw /dev/raw/raw46 /dev/sdau1
raw /dev/raw/raw47 /dev/sdav1
raw /dev/raw/raw48 /dev/sdaw1
```

```
raw /dev/raw/raw101 /dev/sdb2
raw /dev/raw/raw102 /dev/sdc2
raw /dev/raw/raw103 /dev/sdd2
raw /dev/raw/raw104 /dev/sde2
raw /dev/raw/raw105 /dev/sdf2
raw /dev/raw/raw106 /dev/sdg2
raw /dev/raw/raw107 /dev/sdh2
raw /dev/raw/raw108 /dev/sdi2
raw /dev/raw/raw109 /dev/sdj2
raw /dev/raw/raw110 /dev/sdk2
raw /dev/raw/raw111 /dev/sdl2
raw /dev/raw/raw112 /dev/sdm2
raw /dev/raw/raw113 /dev/sdn2
raw /dev/raw/raw114 /dev/sdo2
raw /dev/raw/raw115 /dev/sdp2
raw /dev/raw/raw116 /dev/sdq2
raw /dev/raw/raw117 /dev/sdr2
raw /dev/raw/raw118 /dev/sds2
raw /dev/raw/raw119 /dev/sdt2
raw /dev/raw/raw120 /dev/sdu2
raw /dev/raw/raw121 /dev/sdv2
raw /dev/raw/raw122 /dev/sdw2
raw /dev/raw/raw123 /dev/sdx2
raw /dev/raw/raw124 /dev/sdy2
raw /dev/raw/raw125 /dev/sdz2
raw /dev/raw/raw126 /dev/sdaa2
raw /dev/raw/raw127 /dev/sdab2
raw /dev/raw/raw128 /dev/sdac2
raw /dev/raw/raw129 /dev/sdad2
raw /dev/raw/raw130 /dev/sdae2
raw /dev/raw/raw131 /dev/sdaf2
raw /dev/raw/raw132 /dev/sdag2
raw /dev/raw/raw133 /dev/sdah2
raw /dev/raw/raw134 /dev/sdai2
raw /dev/raw/raw135 /dev/sdaj2
raw /dev/raw/raw136 /dev/sdak2
raw /dev/raw/raw137 /dev/sdal2
raw /dev/raw/raw138 /dev/sdam2
raw /dev/raw/raw139 /dev/sdan2
raw /dev/raw/raw140 /dev/sdao2
raw /dev/raw/raw141 /dev/sdap2
raw /dev/raw/raw142 /dev/sdaq2
raw /dev/raw/raw143 /dev/sdar2
raw /dev/raw/raw144 /dev/sdas2
raw /dev/raw/raw145 /dev/sdat2
raw /dev/raw/raw146 /dev/sdau2
raw /dev/raw/raw147 /dev/sdav2
raw /dev/raw/raw148 /dev/sdaw2
```

```
raw /dev/raw/raw201 /dev/sdb3
raw /dev/raw/raw202 /dev/sdc3
raw /dev/raw/raw203 /dev/sdd3
raw /dev/raw/raw204 /dev/sde3
raw /dev/raw/raw205 /dev/sdf3
```

```
raw /dev/raw/raw206 /dev/sdg3
raw /dev/raw/raw207 /dev/sdh3
raw /dev/raw/raw208 /dev/sdi3
raw /dev/raw/raw209 /dev/sdj3
raw /dev/raw/raw210 /dev/sdk3
raw /dev/raw/raw211 /dev/sdl3
raw /dev/raw/raw212 /dev/sdm3
raw /dev/raw/raw213 /dev/sdn3
raw /dev/raw/raw214 /dev/sdo3
raw /dev/raw/raw215 /dev/sdp3
raw /dev/raw/raw216 /dev/sdq3
raw /dev/raw/raw217 /dev/sdr3
raw /dev/raw/raw218 /dev/sds3
raw /dev/raw/raw219 /dev/sdt3
raw /dev/raw/raw220 /dev/sdu3
raw /dev/raw/raw221 /dev/sdv3
raw /dev/raw/raw222 /dev/sdw3
raw /dev/raw/raw223 /dev/sdx3
raw /dev/raw/raw224 /dev/sdy3
raw /dev/raw/raw225 /dev/sdz3
raw /dev/raw/raw226 /dev/sdaa3
raw /dev/raw/raw227 /dev/sdab3
raw /dev/raw/raw228 /dev/sdac3
raw /dev/raw/raw229 /dev/sdad3
raw /dev/raw/raw230 /dev/sdae3
raw /dev/raw/raw231 /dev/sdaf3
raw /dev/raw/raw232 /dev/sdag3
raw /dev/raw/raw233 /dev/sdah3
raw /dev/raw/raw234 /dev/sdai3
raw /dev/raw/raw235 /dev/sdaj3
raw /dev/raw/raw236 /dev/sdak3
raw /dev/raw/raw237 /dev/sdal3
raw /dev/raw/raw238 /dev/sdam3
raw /dev/raw/raw239 /dev/sdan3
raw /dev/raw/raw240 /dev/sdao3
raw /dev/raw/raw241 /dev/sdap3
raw /dev/raw/raw242 /dev/sdaq3
raw /dev/raw/raw243 /dev/sdar3
raw /dev/raw/raw244 /dev/sdas3
raw /dev/raw/raw245 /dev/sdat3
raw /dev/raw/raw246 /dev/sdau3
raw /dev/raw/raw247 /dev/sdav3
raw /dev/raw/raw248 /dev/sdaw3
```

```
raw /dev/raw/raw301 /dev/sdb5
raw /dev/raw/raw302 /dev/sdc5
raw /dev/raw/raw303 /dev/sdd5
raw /dev/raw/raw304 /dev/sde5
raw /dev/raw/raw305 /dev/sdf5
raw /dev/raw/raw306 /dev/sdg5
raw /dev/raw/raw307 /dev/sdh5
raw /dev/raw/raw308 /dev/sdi5
raw /dev/raw/raw309 /dev/sdj5
raw /dev/raw/raw310 /dev/sdk5
raw /dev/raw/raw311 /dev/sdl5
raw /dev/raw/raw312 /dev/sdm5
raw /dev/raw/raw313 /dev/sdn5
raw /dev/raw/raw314 /dev/sdo5
raw /dev/raw/raw315 /dev/sdp5
raw /dev/raw/raw316 /dev/sdq5
raw /dev/raw/raw317 /dev/sdr5
raw /dev/raw/raw318 /dev/sds5
raw /dev/raw/raw319 /dev/sdt5
raw /dev/raw/raw320 /dev/sdu5
raw /dev/raw/raw321 /dev/sdv5
raw /dev/raw/raw322 /dev/sdw5
raw /dev/raw/raw323 /dev/sdx5
raw /dev/raw/raw324 /dev/sdy5
raw /dev/raw/raw325 /dev/sdz5
raw /dev/raw/raw326 /dev/sdaa5
```



```
chmod 777 /dev/raw/raw*
```

version.txt

```
Linux itcopus82 2.6.16.21-0.8-smp #1 SMP Mon Jul 3 18:25:39 UTC 2006 x86_64 x86_64  
x86_64 GNU/Linux
```



```
ALTER TABLE DISTRICT37 ADD CONSTRAINT DISTRICT37CKC CHECK (D_W_ID
BETWEEN 21025 AND 21608);
SET INTEGRITY FOR DISTRICT37 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT38 OFF;
ALTER TABLE DISTRICT38 DROP CONSTRAINT DISTRICT38CKC;
ALTER TABLE DISTRICT38 ADD CONSTRAINT DISTRICT38CKC CHECK (D_W_ID
BETWEEN 21609 AND 22192);
SET INTEGRITY FOR DISTRICT38 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT39 OFF;
ALTER TABLE DISTRICT39 DROP CONSTRAINT DISTRICT39CKC;
ALTER TABLE DISTRICT39 ADD CONSTRAINT DISTRICT39CKC CHECK (D_W_ID
BETWEEN 22193 AND 22776);
SET INTEGRITY FOR DISTRICT39 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT40 OFF;
ALTER TABLE DISTRICT40 DROP CONSTRAINT DISTRICT40CKC;
ALTER TABLE DISTRICT40 ADD CONSTRAINT DISTRICT40CKC CHECK (D_W_ID
BETWEEN 22777 AND 23360);
SET INTEGRITY FOR DISTRICT40 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT41 OFF;
ALTER TABLE DISTRICT41 DROP CONSTRAINT DISTRICT41CKC;
ALTER TABLE DISTRICT41 ADD CONSTRAINT DISTRICT41CKC CHECK (D_W_ID
BETWEEN 23361 AND 23944);
SET INTEGRITY FOR DISTRICT41 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT42 OFF;
ALTER TABLE DISTRICT42 DROP CONSTRAINT DISTRICT42CKC;
ALTER TABLE DISTRICT42 ADD CONSTRAINT DISTRICT42CKC CHECK (D_W_ID
BETWEEN 23945 AND 24528);
SET INTEGRITY FOR DISTRICT42 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT43 OFF;
ALTER TABLE DISTRICT43 DROP CONSTRAINT DISTRICT43CKC;
ALTER TABLE DISTRICT43 ADD CONSTRAINT DISTRICT43CKC CHECK (D_W_ID
BETWEEN 24529 AND 25112);
SET INTEGRITY FOR DISTRICT43 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT44 OFF;
ALTER TABLE DISTRICT44 DROP CONSTRAINT DISTRICT44CKC;
ALTER TABLE DISTRICT44 ADD CONSTRAINT DISTRICT44CKC CHECK (D_W_ID
BETWEEN 25113 AND 25696);
SET INTEGRITY FOR DISTRICT44 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT45 OFF;
ALTER TABLE DISTRICT45 DROP CONSTRAINT DISTRICT45CKC;
ALTER TABLE DISTRICT45 ADD CONSTRAINT DISTRICT45CKC CHECK (D_W_ID
BETWEEN 25697 AND 26280);
SET INTEGRITY FOR DISTRICT45 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT46 OFF;
ALTER TABLE DISTRICT46 DROP CONSTRAINT DISTRICT46CKC;
ALTER TABLE DISTRICT46 ADD CONSTRAINT DISTRICT46CKC CHECK (D_W_ID
BETWEEN 26281 AND 26864);
SET INTEGRITY FOR DISTRICT46 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT47 OFF;
ALTER TABLE DISTRICT47 DROP CONSTRAINT DISTRICT47CKC;
```

```
ALTER TABLE DISTRICT47 ADD CONSTRAINT DISTRICT47CKC CHECK (D_W_ID
BETWEEN 26865 AND 27448);
SET INTEGRITY FOR DISTRICT47 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT48 OFF;
ALTER TABLE DISTRICT48 DROP CONSTRAINT DISTRICT48CKC;
ALTER TABLE DISTRICT48 ADD CONSTRAINT DISTRICT48CKC CHECK (D_W_ID =>
27449);
SET INTEGRITY FOR DISTRICT48 ALL IMMEDIATE UNCHECKED;
connect reset;
```

DDL/CRCONST HISTORY.ddl

```
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY1 OFF;
ALTER TABLE HISTORY1 DROP CONSTRAINT HISTORY1CKC;
ALTER TABLE HISTORY1 ADD CONSTRAINT HISTORY1CKC CHECK (H_W_ID
BETWEEN 1 AND 584);
SET INTEGRITY FOR HISTORY1 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY2 OFF;
ALTER TABLE HISTORY2 DROP CONSTRAINT HISTORY2CKC;
ALTER TABLE HISTORY2 ADD CONSTRAINT HISTORY2CKC CHECK (H_W_ID
BETWEEN 585 AND 1168);
SET INTEGRITY FOR HISTORY2 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY3 OFF;
ALTER TABLE HISTORY3 DROP CONSTRAINT HISTORY3CKC;
ALTER TABLE HISTORY3 ADD CONSTRAINT HISTORY3CKC CHECK (H_W_ID
BETWEEN 1169 AND 1752);
SET INTEGRITY FOR HISTORY3 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY4 OFF;
ALTER TABLE HISTORY4 DROP CONSTRAINT HISTORY4CKC;
ALTER TABLE HISTORY4 ADD CONSTRAINT HISTORY4CKC CHECK (H_W_ID
BETWEEN 1753 AND 2336);
SET INTEGRITY FOR HISTORY4 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY5 OFF;
ALTER TABLE HISTORY5 DROP CONSTRAINT HISTORY5CKC;
ALTER TABLE HISTORY5 ADD CONSTRAINT HISTORY5CKC CHECK (H_W_ID
BETWEEN 2337 AND 2920);
SET INTEGRITY FOR HISTORY5 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY6 OFF;
ALTER TABLE HISTORY6 DROP CONSTRAINT HISTORY6CKC;
ALTER TABLE HISTORY6 ADD CONSTRAINT HISTORY6CKC CHECK (H_W_ID
BETWEEN 2921 AND 3504);
SET INTEGRITY FOR HISTORY6 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY7 OFF;
ALTER TABLE HISTORY7 DROP CONSTRAINT HISTORY7CKC;
ALTER TABLE HISTORY7 ADD CONSTRAINT HISTORY7CKC CHECK (H_W_ID
BETWEEN 3505 AND 4088);
SET INTEGRITY FOR HISTORY7 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY8 OFF;
ALTER TABLE HISTORY8 DROP CONSTRAINT HISTORY8CKC;
ALTER TABLE HISTORY8 ADD CONSTRAINT HISTORY8CKC CHECK (H_W_ID
BETWEEN 4089 AND 4672);
SET INTEGRITY FOR HISTORY8 ALL IMMEDIATE UNCHECKED;
connect reset;
```

```
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY9 OFF;
ALTER TABLE HISTORY9 DROP CONSTRAINT HISTORY9CKC;
ALTER TABLE HISTORY9 ADD CONSTRAINT HISTORY9CKC CHECK (H_W_ID
BETWEEN 4673 AND 5256);
SET INTEGRITY FOR HISTORY9 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY10 OFF;
ALTER TABLE HISTORY10 DROP CONSTRAINT HISTORY10CKC;
ALTER TABLE HISTORY10 ADD CONSTRAINT HISTORY10CKC CHECK (H_W_ID
BETWEEN 5257 AND 5840);
SET INTEGRITY FOR HISTORY10 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY11 OFF;
ALTER TABLE HISTORY11 DROP CONSTRAINT HISTORY11CKC;
ALTER TABLE HISTORY11 ADD CONSTRAINT HISTORY11CKC CHECK (H_W_ID
BETWEEN 5841 AND 6424);
SET INTEGRITY FOR HISTORY11 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY12 OFF;
ALTER TABLE HISTORY12 DROP CONSTRAINT HISTORY12CKC;
ALTER TABLE HISTORY12 ADD CONSTRAINT HISTORY12CKC CHECK (H_W_ID
BETWEEN 6425 AND 7008);
SET INTEGRITY FOR HISTORY12 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY13 OFF;
ALTER TABLE HISTORY13 DROP CONSTRAINT HISTORY13CKC;
ALTER TABLE HISTORY13 ADD CONSTRAINT HISTORY13CKC CHECK (H_W_ID
BETWEEN 7009 AND 7592);
SET INTEGRITY FOR HISTORY13 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY14 OFF;
ALTER TABLE HISTORY14 DROP CONSTRAINT HISTORY14CKC;
ALTER TABLE HISTORY14 ADD CONSTRAINT HISTORY14CKC CHECK (H_W_ID
BETWEEN 7593 AND 8176);
SET INTEGRITY FOR HISTORY14 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY15 OFF;
ALTER TABLE HISTORY15 DROP CONSTRAINT HISTORY15CKC;
ALTER TABLE HISTORY15 ADD CONSTRAINT HISTORY15CKC CHECK (H_W_ID
BETWEEN 8177 AND 8760);
SET INTEGRITY FOR HISTORY15 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY16 OFF;
ALTER TABLE HISTORY16 DROP CONSTRAINT HISTORY16CKC;
ALTER TABLE HISTORY16 ADD CONSTRAINT HISTORY16CKC CHECK (H_W_ID
BETWEEN 8761 AND 9344);
SET INTEGRITY FOR HISTORY16 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY17 OFF;
ALTER TABLE HISTORY17 DROP CONSTRAINT HISTORY17CKC;
ALTER TABLE HISTORY17 ADD CONSTRAINT HISTORY17CKC CHECK (H_W_ID
BETWEEN 9345 AND 9928);
SET INTEGRITY FOR HISTORY17 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY18 OFF;
ALTER TABLE HISTORY18 DROP CONSTRAINT HISTORY18CKC;
ALTER TABLE HISTORY18 ADD CONSTRAINT HISTORY18CKC CHECK (H_W_ID
BETWEEN 9929 AND 10512);
SET INTEGRITY FOR HISTORY18 ALL IMMEDIATE UNCHECKED;
connect reset;
```



```

DROP INDEX ORDR_IDXB32;
CREATE INDEX ORDR_IDXB32
    ON ORDERS32(O_C_ID, O_W_ID, O_D_ID, O_ID DESC) PCTFREE 20;
LEVEL2 PCTFREE 20;
connect reset;
connect to TPCC in share mode;
DROP INDEX ORDR_IDXB33;
CREATE INDEX ORDR_IDXB33
    ON ORDERS33(O_C_ID, O_W_ID, O_D_ID, O_ID DESC) PCTFREE 20;
LEVEL2 PCTFREE 20;
connect reset;
connect to TPCC in share mode;
DROP INDEX ORDR_IDXB34;
CREATE INDEX ORDR_IDXB34
    ON ORDERS34(O_C_ID, O_W_ID, O_D_ID, O_ID DESC) PCTFREE 20;
LEVEL2 PCTFREE 20;
connect reset;
connect to TPCC in share mode;
DROP INDEX ORDR_IDXB35;
CREATE INDEX ORDR_IDXB35
    ON ORDERS35(O_C_ID, O_W_ID, O_D_ID, O_ID DESC) PCTFREE 20;
LEVEL2 PCTFREE 20;
connect reset;
connect to TPCC in share mode;
DROP INDEX ORDR_IDXB36;
CREATE INDEX ORDR_IDXB36
    ON ORDERS36(O_C_ID, O_W_ID, O_D_ID, O_ID DESC) PCTFREE 20;
LEVEL2 PCTFREE 20;
connect reset;
connect to TPCC in share mode;
DROP INDEX ORDR_IDXB37;
CREATE INDEX ORDR_IDXB37
    ON ORDERS37(O_C_ID, O_W_ID, O_D_ID, O_ID DESC) PCTFREE 20;
LEVEL2 PCTFREE 20;
connect reset;
connect to TPCC in share mode;
DROP INDEX ORDR_IDXB38;
CREATE INDEX ORDR_IDXB38
    ON ORDERS38(O_C_ID, O_W_ID, O_D_ID, O_ID DESC) PCTFREE 20;
LEVEL2 PCTFREE 20;
connect reset;
connect to TPCC in share mode;
DROP INDEX ORDR_IDXB39;
CREATE INDEX ORDR_IDXB39
    ON ORDERS39(O_C_ID, O_W_ID, O_D_ID, O_ID DESC) PCTFREE 20;
LEVEL2 PCTFREE 20;
connect reset;
connect to TPCC in share mode;
DROP INDEX ORDR_IDXB40;
CREATE INDEX ORDR_IDXB40
    ON ORDERS40(O_C_ID, O_W_ID, O_D_ID, O_ID DESC) PCTFREE 20;
LEVEL2 PCTFREE 20;
connect reset;
connect to TPCC in share mode;
DROP INDEX ORDR_IDXB41;
CREATE INDEX ORDR_IDXB41
    ON ORDERS41(O_C_ID, O_W_ID, O_D_ID, O_ID DESC) PCTFREE 20;
LEVEL2 PCTFREE 20;
connect reset;
connect to TPCC in share mode;
DROP INDEX ORDR_IDXB42;
CREATE INDEX ORDR_IDXB42
    ON ORDERS42(O_C_ID, O_W_ID, O_D_ID, O_ID DESC) PCTFREE 20;
LEVEL2 PCTFREE 20;
connect reset;
connect to TPCC in share mode;
DROP INDEX ORDR_IDXB43;
CREATE INDEX ORDR_IDXB43
    ON ORDERS43(O_C_ID, O_W_ID, O_D_ID, O_ID DESC) PCTFREE 20;
LEVEL2 PCTFREE 20;

```

```

connect reset;
connect to TPCC in share mode;
DROP INDEX ORDR_IDXB44;
CREATE INDEX ORDR_IDXB44
    ON ORDERS44(O_C_ID, O_W_ID, O_D_ID, O_ID DESC) PCTFREE 20;
LEVEL2 PCTFREE 20;
connect reset;
connect to TPCC in share mode;
DROP INDEX ORDR_IDXB45;
CREATE INDEX ORDR_IDXB45
    ON ORDERS45(O_C_ID, O_W_ID, O_D_ID, O_ID DESC) PCTFREE 20;
LEVEL2 PCTFREE 20;
connect reset;
connect to TPCC in share mode;
DROP INDEX ORDR_IDXB46;
CREATE INDEX ORDR_IDXB46
    ON ORDERS46(O_C_ID, O_W_ID, O_D_ID, O_ID DESC) PCTFREE 20;
LEVEL2 PCTFREE 20;
connect reset;
connect to TPCC in share mode;
DROP INDEX ORDR_IDXB47;
CREATE INDEX ORDR_IDXB47
    ON ORDERS47(O_C_ID, O_W_ID, O_D_ID, O_ID DESC) PCTFREE 20;
LEVEL2 PCTFREE 20;
connect reset;
connect to TPCC in share mode;
DROP INDEX ORDR_IDXB48;
CREATE INDEX ORDR_IDXB48
    ON ORDERS48(O_C_ID, O_W_ID, O_D_ID, O_ID DESC) PCTFREE 20;
LEVEL2 PCTFREE 20;
connect reset;

```

DDL/CRTB_CUSTOMER.ddl

```

connect to TPCC in share mode;
DROP TABLE CUSTOMER1;
CREATE TABLE CUSTOMER1
(
    C_ID          INTEGER      NOT NULL,
    C_STATE      CHAR(2)      NOT NULL,
    C_ZIP        CHAR(9)      NOT NULL,
    C_PHONE      CHAR(16)     NOT NULL,
    C_SINCE      TIMESTAMP    NOT NULL,
    C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
    C_MIDDLE     CHAR(2)      NOT NULL,
    C_CREDIT     CHAR(2)      NOT NULL,
    C_DISCOUNT  REAL         NOT NULL,
    C_DATA       VARCHAR(500) NOT NULL,
    C_LAST       VARCHAR(16)  NOT NULL,
    C_FIRST      VARCHAR(16)  NOT NULL,
    C_STREET_1   VARCHAR(20)  NOT NULL,
    C_STREET_2   VARCHAR(20)  NOT NULL,
    C_CITY       VARCHAR(20)  NOT NULL,
    C_D_ID       SMALLINT    NOT NULL,
    C_W_ID       INTEGER      NOT NULL,
    C_DELIVERY_CNT INTEGER    NOT NULL,
    C_BALANCE    DECIMAL(12,2) NOT NULL,
    C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
    C_PAYMENT_CNT INTEGER    NOT NULL
)
IN ts_customer_001
INDEX IN is_customer_001
ORGANIZE BY KEY SEQUENCE (
    C_ID STARTING FROM 1 ENDING AT 3000,
    C_W_ID STARTING FROM 1 ENDING AT 584,
    C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;

```

```

connect to TPCC in share mode;
DROP TABLE CUSTOMER2;
CREATE TABLE CUSTOMER2
(
    C_ID          INTEGER      NOT NULL,
    C_STATE      CHAR(2)      NOT NULL,
    C_ZIP        CHAR(9)      NOT NULL,
    C_PHONE      CHAR(16)     NOT NULL,
    C_SINCE      TIMESTAMP    NOT NULL,
    C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
    C_MIDDLE     CHAR(2)      NOT NULL,
    C_CREDIT     CHAR(2)      NOT NULL,
    C_DISCOUNT  REAL         NOT NULL,
    C_DATA       VARCHAR(500) NOT NULL,
    C_LAST       VARCHAR(16)  NOT NULL,
    C_FIRST      VARCHAR(16)  NOT NULL,
    C_STREET_1   VARCHAR(20)  NOT NULL,
    C_STREET_2   VARCHAR(20)  NOT NULL,
    C_CITY       VARCHAR(20)  NOT NULL,
    C_D_ID       SMALLINT    NOT NULL,
    C_W_ID       INTEGER      NOT NULL,
    C_DELIVERY_CNT INTEGER    NOT NULL,
    C_BALANCE    DECIMAL(12,2) NOT NULL,
    C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
    C_PAYMENT_CNT INTEGER    NOT NULL
)
IN ts_customer_002
INDEX IN is_customer_002
ORGANIZE BY KEY SEQUENCE (
    C_ID STARTING FROM 1 ENDING AT 3000,
    C_W_ID STARTING FROM 585 ENDING AT 1168,
    C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER3;
CREATE TABLE CUSTOMER3
(
    C_ID          INTEGER      NOT NULL,
    C_STATE      CHAR(2)      NOT NULL,
    C_ZIP        CHAR(9)      NOT NULL,
    C_PHONE      CHAR(16)     NOT NULL,
    C_SINCE      TIMESTAMP    NOT NULL,
    C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
    C_MIDDLE     CHAR(2)      NOT NULL,
    C_CREDIT     CHAR(2)      NOT NULL,
    C_DISCOUNT  REAL         NOT NULL,
    C_DATA       VARCHAR(500) NOT NULL,
    C_LAST       VARCHAR(16)  NOT NULL,
    C_FIRST      VARCHAR(16)  NOT NULL,
    C_STREET_1   VARCHAR(20)  NOT NULL,
    C_STREET_2   VARCHAR(20)  NOT NULL,
    C_CITY       VARCHAR(20)  NOT NULL,
    C_D_ID       SMALLINT    NOT NULL,
    C_W_ID       INTEGER      NOT NULL,
    C_DELIVERY_CNT INTEGER    NOT NULL,
    C_BALANCE    DECIMAL(12,2) NOT NULL,
    C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
    C_PAYMENT_CNT INTEGER    NOT NULL
)
IN ts_customer_003
INDEX IN is_customer_003
ORGANIZE BY KEY SEQUENCE (
    C_ID STARTING FROM 1 ENDING AT 3000,
    C_W_ID STARTING FROM 1169 ENDING AT 1752,
    C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;

```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER4;
CREATE TABLE CUSTOMER4
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
IN ts_customer_004
INDEX IN is_customer_004
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 1753 ENDING AT 2336,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER5;
CREATE TABLE CUSTOMER5
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
IN ts_customer_005
INDEX IN is_customer_005
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 2337 ENDING AT 2920,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER6;
CREATE TABLE CUSTOMER6
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
IN ts_customer_006
INDEX IN is_customer_006
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 2921 ENDING AT 3504,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER7;
CREATE TABLE CUSTOMER7
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
IN ts_customer_007
INDEX IN is_customer_007
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 3505 ENDING AT 4088,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER8;
CREATE TABLE CUSTOMER8
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
IN ts_customer_008
INDEX IN is_customer_008
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 4089 ENDING AT 4672,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER9;
CREATE TABLE CUSTOMER9
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
IN ts_customer_009
INDEX IN is_customer_009
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 4673 ENDING AT 5256,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
```

```
connect reset;
```



```
connect to TPCC in share mode;
DROP TABLE CUSTOMER16;
CREATE TABLE CUSTOMER16
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
IN ts_customer_016
INDEX IN is_customer_016
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 8761 ENDING AT 9344,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER17;
CREATE TABLE CUSTOMER17
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
IN ts_customer_017
INDEX IN is_customer_017
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 9345 ENDING AT 9928,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER18;
CREATE TABLE CUSTOMER18
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
IN ts_customer_018
INDEX IN is_customer_018
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 9929 ENDING AT 10512,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER19;
CREATE TABLE CUSTOMER19
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
IN ts_customer_019
INDEX IN is_customer_019
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 10513 ENDING AT 11096,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER20;
CREATE TABLE CUSTOMER20
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
IN ts_customer_020
INDEX IN is_customer_020
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 11097 ENDING AT 11680,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER21;
CREATE TABLE CUSTOMER21
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
IN ts_customer_021
INDEX IN is_customer_021
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 11681 ENDING AT 12264,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
```

```
connect reset;
```



```
connect to TPCC in share mode;
DROP TABLE CUSTOMER22;
CREATE TABLE CUSTOMER22
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_022
INDEX IN is_customer_022
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 12265 ENDING AT 12848,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER23;
CREATE TABLE CUSTOMER23
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_023
INDEX IN is_customer_023
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 12849 ENDING AT 13432,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER24;
CREATE TABLE CUSTOMER24
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_024
INDEX IN is_customer_024
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 13433 ENDING AT 14016,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER25;
CREATE TABLE CUSTOMER25
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_025
INDEX IN is_customer_025
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 14017 ENDING AT 14600,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER26;
CREATE TABLE CUSTOMER26
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_026
INDEX IN is_customer_026
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 14601 ENDING AT 15184,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER27;
CREATE TABLE CUSTOMER27
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_027
INDEX IN is_customer_027
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 15185 ENDING AT 15768,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER28;
CREATE TABLE CUSTOMER28
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_028
INDEX IN is_customer_028
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 15769 ENDING AT 16352,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER29;
CREATE TABLE CUSTOMER29
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_029
INDEX IN is_customer_029
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 16353 ENDING AT 16936,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER30;
CREATE TABLE CUSTOMER30
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_030
INDEX IN is_customer_030
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 16937 ENDING AT 17520,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER31;
CREATE TABLE CUSTOMER31
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_031
INDEX IN is_customer_031
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 17521 ENDING AT 18104,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER32;
CREATE TABLE CUSTOMER32
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_032
INDEX IN is_customer_032
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 18105 ENDING AT 18688,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER33;
CREATE TABLE CUSTOMER33
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_033
INDEX IN is_customer_033
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 18689 ENDING AT 19272,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER34;
CREATE TABLE CUSTOMER34
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_034
INDEX IN is_customer_034
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 19273 ENDING AT 19856,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER35;
CREATE TABLE CUSTOMER35
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_035
INDEX IN is_customer_035
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 19857 ENDING AT 20440,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER36;
CREATE TABLE CUSTOMER36
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_036
INDEX IN is_customer_036
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 20441 ENDING AT 21024,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER37;
CREATE TABLE CUSTOMER37
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_037
INDEX IN is_customer_037
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 21025 ENDING AT 21608,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER38;
CREATE TABLE CUSTOMER38
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_038
INDEX IN is_customer_038
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 21609 ENDING AT 22192,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER39;
CREATE TABLE CUSTOMER39
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_039
INDEX IN is_customer_039
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 22193 ENDING AT 22776,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER40;
CREATE TABLE CUSTOMER40
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT    NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_040
INDEX IN is_customer_040
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 22777 ENDING AT 23360,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER41;
CREATE TABLE CUSTOMER41
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT    NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_041
INDEX IN is_customer_041
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 23361 ENDING AT 23944,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER42;
CREATE TABLE CUSTOMER42
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT    NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_042
INDEX IN is_customer_042
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 23945 ENDING AT 24528,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER43;
CREATE TABLE CUSTOMER43
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT    NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_043
INDEX IN is_customer_043
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 24529 ENDING AT 25112,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER44;
CREATE TABLE CUSTOMER44
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT    NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_044
INDEX IN is_customer_044
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 25113 ENDING AT 25696,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER45;
CREATE TABLE CUSTOMER45
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT    NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
```

```
IN ts_customer_045
INDEX IN is_customer_045
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 25697 ENDING AT 26280,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
```

```
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER46;
CREATE TABLE CUSTOMER46
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
IN ts_customer_046
INDEX IN is_customer_046
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 26281 ENDING AT 26864,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER47;
CREATE TABLE CUSTOMER47
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
IN ts_customer_047
INDEX IN is_customer_047
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 26865 ENDING AT 27448,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE CUSTOMER48;
CREATE TABLE CUSTOMER48
```

```
(
  C_ID      INTEGER      NOT NULL,
  C_STATE   CHAR(2)      NOT NULL,
  C_ZIP     CHAR(9)      NOT NULL,
  C_PHONE   CHAR(16)     NOT NULL,
  C_SINCE   TIMESTAMP    NOT NULL,
  C_CREDIT_LIM DECIMAL(12,2) NOT NULL,
  C_MIDDLE  CHAR(2)      NOT NULL,
  C_CREDIT  CHAR(2)      NOT NULL,
  C_DISCOUNT REAL      NOT NULL,
  C_DATA    VARCHAR(500) NOT NULL,
  C_LAST    VARCHAR(16)  NOT NULL,
  C_FIRST   VARCHAR(16)  NOT NULL,
  C_STREET_1 VARCHAR(20) NOT NULL,
  C_STREET_2 VARCHAR(20) NOT NULL,
  C_CITY    VARCHAR(20)  NOT NULL,
  C_D_ID    SMALLINT     NOT NULL,
  C_W_ID    INTEGER      NOT NULL,
  C_DELIVERY_CNT INTEGER  NOT NULL,
  C_BALANCE DECIMAL(12,2) NOT NULL,
  C_YTD_PAYMENT DECIMAL(12,2) NOT NULL,
  C_PAYMENT_CNT INTEGER  NOT NULL
)
IN ts_customer_048
INDEX IN is_customer_048
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 27449 ENDING AT 28032,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
```

```
connect reset;
```

DDL/CRTB_DISTRICT.ddl

```
connect to TPCC in share mode;
DROP TABLE DISTRICT1;
CREATE TABLE DISTRICT1
```

```
(
  D_NEXT_O_ID INTEGER      NOT NULL,
  D_TAX      REAL          NOT NULL,
  D_YTD      DECIMAL(12,2) NOT NULL,
  D_NAME     CHAR(10)     NOT NULL,
  D_STREET_1 CHAR(20)    NOT NULL,
  D_STREET_2 CHAR(20)    NOT NULL,
  D_CITY     CHAR(20)     NOT NULL,
  D_STATE    CHAR(2)      NOT NULL,
  D_ZIP      CHAR(9)      NOT NULL,
  D_ID       SMALLINT     NOT NULL,
  D_W_ID     INTEGER      NOT NULL
)
IN ts_dis_001
INDEX IN ts_dis_001
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 1 ENDING AT 584
)
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE DISTRICT2;
CREATE TABLE DISTRICT2
```

```
(
  D_NEXT_O_ID INTEGER      NOT NULL,
  D_TAX      REAL          NOT NULL,
  D_YTD      DECIMAL(12,2) NOT NULL,
  D_NAME     CHAR(10)     NOT NULL,

```

```

  D_STREET_1 CHAR(20)    NOT NULL,
  D_STREET_2 CHAR(20)    NOT NULL,
  D_CITY     CHAR(20)     NOT NULL,
  D_STATE    CHAR(2)      NOT NULL,
  D_ZIP      CHAR(9)      NOT NULL,
  D_ID       SMALLINT     NOT NULL,
  D_W_ID     INTEGER      NOT NULL
)
IN ts_dis_002
INDEX IN ts_dis_002
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 585 ENDING AT 1168
)
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE DISTRICT3;
CREATE TABLE DISTRICT3
```

```
(
  D_NEXT_O_ID INTEGER      NOT NULL,
  D_TAX      REAL          NOT NULL,
  D_YTD      DECIMAL(12,2) NOT NULL,
  D_NAME     CHAR(10)     NOT NULL,
  D_STREET_1 CHAR(20)    NOT NULL,
  D_STREET_2 CHAR(20)    NOT NULL,
  D_CITY     CHAR(20)     NOT NULL,
  D_STATE    CHAR(2)      NOT NULL,
  D_ZIP      CHAR(9)      NOT NULL,
  D_ID       SMALLINT     NOT NULL,
  D_W_ID     INTEGER      NOT NULL
)
IN ts_dis_003
INDEX IN ts_dis_003
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 1169 ENDING AT 1752
)
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE DISTRICT4;
CREATE TABLE DISTRICT4
```

```
(
  D_NEXT_O_ID INTEGER      NOT NULL,
  D_TAX      REAL          NOT NULL,
  D_YTD      DECIMAL(12,2) NOT NULL,
  D_NAME     CHAR(10)     NOT NULL,
  D_STREET_1 CHAR(20)    NOT NULL,
  D_STREET_2 CHAR(20)    NOT NULL,
  D_CITY     CHAR(20)     NOT NULL,
  D_STATE    CHAR(2)      NOT NULL,
  D_ZIP      CHAR(9)      NOT NULL,
  D_ID       SMALLINT     NOT NULL,
  D_W_ID     INTEGER      NOT NULL
)
IN ts_dis_004
INDEX IN ts_dis_004
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 1753 ENDING AT 2336
)
ALLOW OVERFLOW;
```

```
connect reset;
```

```
connect to TPCC in share mode;
DROP TABLE DISTRICT5;
CREATE TABLE DISTRICT5
```

```
(
  D_NEXT_O_ID INTEGER      NOT NULL,
  D_TAX      REAL          NOT NULL,

```

```

D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_005
INDEX IN ts_dis_005
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 2337 ENDING AT 2920
)
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT6;
CREATE TABLE DISTRICT6
(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_006
INDEX IN ts_dis_006
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 2921 ENDING AT 3504
)
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT7;
CREATE TABLE DISTRICT7
(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_007
INDEX IN ts_dis_007
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 3505 ENDING AT 4088
)
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT8;
CREATE TABLE DISTRICT8
(

```

```

D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_008
INDEX IN ts_dis_008
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 4089 ENDING AT 4672
)
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT9;
CREATE TABLE DISTRICT9
(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_009
INDEX IN ts_dis_009
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 4673 ENDING AT 5256
)
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT10;
CREATE TABLE DISTRICT10
(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_010
INDEX IN ts_dis_010
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 5257 ENDING AT 5840
)
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT11;

```

```

CREATE TABLE DISTRICT11
(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_011
INDEX IN ts_dis_011
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 5841 ENDING AT 6424
)
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT12;
CREATE TABLE DISTRICT12
(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_012
INDEX IN ts_dis_012
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 6425 ENDING AT 7008
)
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT13;
CREATE TABLE DISTRICT13
(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_013
INDEX IN ts_dis_013
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 7009 ENDING AT 7592
)
)
ALLOW OVERFLOW;
connect reset;

```

```

connect to TPCC in share mode;
DROP TABLE DISTRICT14;
CREATE TABLE DISTRICT14
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,
  D_ZIP CHAR(9) NOT NULL,
  D_ID SMALLINT NOT NULL,
  D_W_ID INTEGER NOT NULL
)
IN ts_dis_014
INDEX IN ts_dis_014
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 7593 ENDING AT 8176
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT15;
CREATE TABLE DISTRICT15
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,
  D_ZIP CHAR(9) NOT NULL,
  D_ID SMALLINT NOT NULL,
  D_W_ID INTEGER NOT NULL
)
IN ts_dis_015
INDEX IN ts_dis_015
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 8177 ENDING AT 8760
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT16;
CREATE TABLE DISTRICT16
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,
  D_ZIP CHAR(9) NOT NULL,
  D_ID SMALLINT NOT NULL,
  D_W_ID INTEGER NOT NULL
)
IN ts_dis_016
INDEX IN ts_dis_016
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 8761 ENDING AT 9344
)

```

```

ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT17;
CREATE TABLE DISTRICT17
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,
  D_ZIP CHAR(9) NOT NULL,
  D_ID SMALLINT NOT NULL,
  D_W_ID INTEGER NOT NULL
)
IN ts_dis_017
INDEX IN ts_dis_017
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 9345 ENDING AT 9928
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT18;
CREATE TABLE DISTRICT18
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,
  D_ZIP CHAR(9) NOT NULL,
  D_ID SMALLINT NOT NULL,
  D_W_ID INTEGER NOT NULL
)
IN ts_dis_018
INDEX IN ts_dis_018
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 9929 ENDING AT 10512
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT19;
CREATE TABLE DISTRICT19
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,
  D_ZIP CHAR(9) NOT NULL,
  D_ID SMALLINT NOT NULL,
  D_W_ID INTEGER NOT NULL
)
IN ts_dis_019
INDEX IN ts_dis_019
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,

```

```

  D_W_ID STARTING FROM 10513 ENDING AT 11096
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT20;
CREATE TABLE DISTRICT20
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,
  D_ZIP CHAR(9) NOT NULL,
  D_ID SMALLINT NOT NULL,
  D_W_ID INTEGER NOT NULL
)
IN ts_dis_020
INDEX IN ts_dis_020
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 11097 ENDING AT 11680
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT21;
CREATE TABLE DISTRICT21
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,
  D_ZIP CHAR(9) NOT NULL,
  D_ID SMALLINT NOT NULL,
  D_W_ID INTEGER NOT NULL
)
IN ts_dis_021
INDEX IN ts_dis_021
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 11681 ENDING AT 12264
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT22;
CREATE TABLE DISTRICT22
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,
  D_ZIP CHAR(9) NOT NULL,
  D_ID SMALLINT NOT NULL,
  D_W_ID INTEGER NOT NULL
)
IN ts_dis_022
INDEX IN ts_dis_022

```

```

ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 12265 ENDING AT 12848
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT23;
CREATE TABLE DISTRICT23
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,
  D_ZIP CHAR(9) NOT NULL,
  D_ID SMALLINT NOT NULL,
  D_W_ID INTEGER NOT NULL
)
IN ts_dis_023
INDEX IN ts_dis_023
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 12849 ENDING AT 13432
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT24;
CREATE TABLE DISTRICT24
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,
  D_ZIP CHAR(9) NOT NULL,
  D_ID SMALLINT NOT NULL,
  D_W_ID INTEGER NOT NULL
)
IN ts_dis_024
INDEX IN ts_dis_024
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 13433 ENDING AT 14016
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT25;
CREATE TABLE DISTRICT25
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,
  D_ZIP CHAR(9) NOT NULL,
  D_ID SMALLINT NOT NULL,
  D_W_ID INTEGER NOT NULL
)

```

```

IN ts_dis_025
INDEX IN ts_dis_025
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 14017 ENDING AT 14600
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT26;
CREATE TABLE DISTRICT26
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,
  D_ZIP CHAR(9) NOT NULL,
  D_ID SMALLINT NOT NULL,
  D_W_ID INTEGER NOT NULL
)
IN ts_dis_026
INDEX IN ts_dis_026
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 14601 ENDING AT 15184
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT27;
CREATE TABLE DISTRICT27
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,
  D_ZIP CHAR(9) NOT NULL,
  D_ID SMALLINT NOT NULL,
  D_W_ID INTEGER NOT NULL
)
IN ts_dis_027
INDEX IN ts_dis_027
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 15185 ENDING AT 15768
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT28;
CREATE TABLE DISTRICT28
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,
  D_ZIP CHAR(9) NOT NULL,
  D_ID SMALLINT NOT NULL,
  D_W_ID INTEGER NOT NULL,

```

```

  D_W_ID INTEGER NOT NULL
)
)
IN ts_dis_028
INDEX IN ts_dis_028
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 15769 ENDING AT 16352
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT29;
CREATE TABLE DISTRICT29
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,
  D_ZIP CHAR(9) NOT NULL,
  D_ID SMALLINT NOT NULL,
  D_W_ID INTEGER NOT NULL
)
IN ts_dis_029
INDEX IN ts_dis_029
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 16353 ENDING AT 16936
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT30;
CREATE TABLE DISTRICT30
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,
  D_ZIP CHAR(9) NOT NULL,
  D_ID SMALLINT NOT NULL,
  D_W_ID INTEGER NOT NULL
)
IN ts_dis_030
INDEX IN ts_dis_030
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 16937 ENDING AT 17520
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT31;
CREATE TABLE DISTRICT31
(
  D_NEXT_O_ID INTEGER NOT NULL,
  D_TAX REAL NOT NULL,
  D_YTD DECIMAL(12,2) NOT NULL,
  D_NAME CHAR(10) NOT NULL,
  D_STREET_1 CHAR(20) NOT NULL,
  D_STREET_2 CHAR(20) NOT NULL,
  D_CITY CHAR(20) NOT NULL,
  D_STATE CHAR(2) NOT NULL,

```



```

D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_031
INDEX IN ts_dis_031
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 17521 ENDING AT 18104
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT32;
CREATE TABLE DISTRICT32

```

```

(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_032
INDEX IN ts_dis_032
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 18105 ENDING AT 18688
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT33;
CREATE TABLE DISTRICT33

```

```

(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_033
INDEX IN ts_dis_033
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 18689 ENDING AT 19272
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT34;
CREATE TABLE DISTRICT34

```

```

(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,

```

```

D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_034
INDEX IN ts_dis_034
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 19273 ENDING AT 19856
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT35;
CREATE TABLE DISTRICT35

```

```

(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_035
INDEX IN ts_dis_035
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 19857 ENDING AT 20440
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT36;
CREATE TABLE DISTRICT36

```

```

(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_036
INDEX IN ts_dis_036
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 20441 ENDING AT 21024
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT37;
CREATE TABLE DISTRICT37

```

```

(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,

```

```

D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_037
INDEX IN ts_dis_037
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 21025 ENDING AT 21608
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT38;
CREATE TABLE DISTRICT38

```

```

(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_038
INDEX IN ts_dis_038
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 21609 ENDING AT 22192
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT39;
CREATE TABLE DISTRICT39

```

```

(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_039
INDEX IN ts_dis_039
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 22193 ENDING AT 22776
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT40;
CREATE TABLE DISTRICT40

```

```

(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,

```

```

D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)
IN ts_dis_040
INDEX IN ts_dis_040
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 22777 ENDING AT 23360
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT41;
CREATE TABLE DISTRICT41

```

```

(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)

```

```

IN ts_dis_041
INDEX IN ts_dis_041
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 23361 ENDING AT 23944
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT42;
CREATE TABLE DISTRICT42

```

```

(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)

```

```

IN ts_dis_042
INDEX IN ts_dis_042
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 23945 ENDING AT 24528
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT43;
CREATE TABLE DISTRICT43

```

```
(
```

```

D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)

```

```

IN ts_dis_043
INDEX IN ts_dis_043
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 24529 ENDING AT 25112
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT44;
CREATE TABLE DISTRICT44

```

```

(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)

```

```

IN ts_dis_044
INDEX IN ts_dis_044
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 25113 ENDING AT 25696
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT45;
CREATE TABLE DISTRICT45

```

```

(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)

```

```

IN ts_dis_045
INDEX IN ts_dis_045
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 25697 ENDING AT 26280
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT46;

```

```
CREATE TABLE DISTRICT46
```

```

(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)

```

```

IN ts_dis_046
INDEX IN ts_dis_046
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 26281 ENDING AT 26864
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT47;
CREATE TABLE DISTRICT47

```

```

(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)

```

```

IN ts_dis_047
INDEX IN ts_dis_047
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 26865 ENDING AT 27448
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT48;
CREATE TABLE DISTRICT48

```

```

(
D_NEXT_O_ID INTEGER NOT NULL,
D_TAX REAL NOT NULL,
D_YTD DECIMAL(12,2) NOT NULL,
D_NAME CHAR(10) NOT NULL,
D_STREET_1 CHAR(20) NOT NULL,
D_STREET_2 CHAR(20) NOT NULL,
D_CITY CHAR(20) NOT NULL,
D_STATE CHAR(2) NOT NULL,
D_ZIP CHAR(9) NOT NULL,
D_ID SMALLINT NOT NULL,
D_W_ID INTEGER NOT NULL
)

```

```

IN ts_dis_048
INDEX IN ts_dis_048
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 27449 ENDING AT 28032
)
ALLOW OVERFLOW;

```

```
connect reset;
```



```

ALTER TABLE HISTORY37 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY38;
CREATE TABLE HISTORY38
(
  H_C_ID      INTEGER NOT NULL,
  H_C_D_ID    SMALLINT NOT NULL,
  H_C_W_ID    INTEGER NOT NULL,
  H_D_ID      SMALLINT NOT NULL,
  H_W_ID      INTEGER NOT NULL,
  H_DATE      TIMESTAMP NOT NULL,
  H_AMOUNT    DECIMAL(6,2) NOT NULL,
  H_DATA      CHAR(24) NOT NULL
)
IN ts_history_038
INDEX IN ts_history_038;
ALTER TABLE HISTORY38 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY39;
CREATE TABLE HISTORY39
(
  H_C_ID      INTEGER NOT NULL,
  H_C_D_ID    SMALLINT NOT NULL,
  H_C_W_ID    INTEGER NOT NULL,
  H_D_ID      SMALLINT NOT NULL,
  H_W_ID      INTEGER NOT NULL,
  H_DATE      TIMESTAMP NOT NULL,
  H_AMOUNT    DECIMAL(6,2) NOT NULL,
  H_DATA      CHAR(24) NOT NULL
)
IN ts_history_039
INDEX IN ts_history_039;
ALTER TABLE HISTORY39 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY40;
CREATE TABLE HISTORY40
(
  H_C_ID      INTEGER NOT NULL,
  H_C_D_ID    SMALLINT NOT NULL,
  H_C_W_ID    INTEGER NOT NULL,
  H_D_ID      SMALLINT NOT NULL,
  H_W_ID      INTEGER NOT NULL,
  H_DATE      TIMESTAMP NOT NULL,
  H_AMOUNT    DECIMAL(6,2) NOT NULL,
  H_DATA      CHAR(24) NOT NULL
)
IN ts_history_040
INDEX IN ts_history_040;
ALTER TABLE HISTORY40 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY41;
CREATE TABLE HISTORY41
(
  H_C_ID      INTEGER NOT NULL,
  H_C_D_ID    SMALLINT NOT NULL,
  H_C_W_ID    INTEGER NOT NULL,
  H_D_ID      SMALLINT NOT NULL,
  H_W_ID      INTEGER NOT NULL,
  H_DATE      TIMESTAMP NOT NULL,
  H_AMOUNT    DECIMAL(6,2) NOT NULL,
  H_DATA      CHAR(24) NOT NULL
)
IN ts_history_041
INDEX IN ts_history_041;
ALTER TABLE HISTORY41 APPEND ON;
connect reset;

```

```

connect to TPCC in share mode;
DROP TABLE HISTORY42;
CREATE TABLE HISTORY42
(
  H_C_ID      INTEGER NOT NULL,
  H_C_D_ID    SMALLINT NOT NULL,
  H_C_W_ID    INTEGER NOT NULL,
  H_D_ID      SMALLINT NOT NULL,
  H_W_ID      INTEGER NOT NULL,
  H_DATE      TIMESTAMP NOT NULL,
  H_AMOUNT    DECIMAL(6,2) NOT NULL,
  H_DATA      CHAR(24) NOT NULL
)
IN ts_history_042
INDEX IN ts_history_042;
ALTER TABLE HISTORY42 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY43;
CREATE TABLE HISTORY43
(
  H_C_ID      INTEGER NOT NULL,
  H_C_D_ID    SMALLINT NOT NULL,
  H_C_W_ID    INTEGER NOT NULL,
  H_D_ID      SMALLINT NOT NULL,
  H_W_ID      INTEGER NOT NULL,
  H_DATE      TIMESTAMP NOT NULL,
  H_AMOUNT    DECIMAL(6,2) NOT NULL,
  H_DATA      CHAR(24) NOT NULL
)
IN ts_history_043
INDEX IN ts_history_043;
ALTER TABLE HISTORY43 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY44;
CREATE TABLE HISTORY44
(
  H_C_ID      INTEGER NOT NULL,
  H_C_D_ID    SMALLINT NOT NULL,
  H_C_W_ID    INTEGER NOT NULL,
  H_D_ID      SMALLINT NOT NULL,
  H_W_ID      INTEGER NOT NULL,
  H_DATE      TIMESTAMP NOT NULL,
  H_AMOUNT    DECIMAL(6,2) NOT NULL,
  H_DATA      CHAR(24) NOT NULL
)
IN ts_history_044
INDEX IN ts_history_044;
ALTER TABLE HISTORY44 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY45;
CREATE TABLE HISTORY45
(
  H_C_ID      INTEGER NOT NULL,
  H_C_D_ID    SMALLINT NOT NULL,
  H_C_W_ID    INTEGER NOT NULL,
  H_D_ID      SMALLINT NOT NULL,
  H_W_ID      INTEGER NOT NULL,
  H_DATE      TIMESTAMP NOT NULL,
  H_AMOUNT    DECIMAL(6,2) NOT NULL,
  H_DATA      CHAR(24) NOT NULL
)
IN ts_history_045
INDEX IN ts_history_045;
ALTER TABLE HISTORY45 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY46;

```

```

CREATE TABLE HISTORY46
(
  H_C_ID      INTEGER NOT NULL,
  H_C_D_ID    SMALLINT NOT NULL,
  H_C_W_ID    INTEGER NOT NULL,
  H_D_ID      SMALLINT NOT NULL,
  H_W_ID      INTEGER NOT NULL,
  H_DATE      TIMESTAMP NOT NULL,
  H_AMOUNT    DECIMAL(6,2) NOT NULL,
  H_DATA      CHAR(24) NOT NULL
)
IN ts_history_046
INDEX IN ts_history_046;
ALTER TABLE HISTORY46 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY47;
CREATE TABLE HISTORY47
(
  H_C_ID      INTEGER NOT NULL,
  H_C_D_ID    SMALLINT NOT NULL,
  H_C_W_ID    INTEGER NOT NULL,
  H_D_ID      SMALLINT NOT NULL,
  H_W_ID      INTEGER NOT NULL,
  H_DATE      TIMESTAMP NOT NULL,
  H_AMOUNT    DECIMAL(6,2) NOT NULL,
  H_DATA      CHAR(24) NOT NULL
)
IN ts_history_047
INDEX IN ts_history_047;
ALTER TABLE HISTORY47 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY48;
CREATE TABLE HISTORY48
(
  H_C_ID      INTEGER NOT NULL,
  H_C_D_ID    SMALLINT NOT NULL,
  H_C_W_ID    INTEGER NOT NULL,
  H_D_ID      SMALLINT NOT NULL,
  H_W_ID      INTEGER NOT NULL,
  H_DATE      TIMESTAMP NOT NULL,
  H_AMOUNT    DECIMAL(6,2) NOT NULL,
  H_DATA      CHAR(24) NOT NULL
)
IN ts_history_048
INDEX IN ts_history_048;
ALTER TABLE HISTORY48 APPEND ON;
connect reset;

DDL/CRTB_ITEM.ddl

connect to TPCC in share mode;
DROP TABLE ITEM;
CREATE TABLE ITEM
(
  I_NAME      CHAR(24) NOT NULL,
  I_PRICE     DECIMAL(5,2) NOT NULL,
  I_DATA      VARCHAR(50) NOT NULL,
  I_IM_ID     INTEGER NOT NULL,
  I_ID        INTEGER NOT NULL
)
IN ts_item
INDEX IN ts_item
ORGANIZE BY KEY SEQUENCE (
  I_ID STARTING FROM 1 ENDING AT 100000
)
ALLOW OVERFLOW;
ALTER TABLE ITEM LOCKSIZE TABLE;
connect reset;

```

DDL/CRTB NEW_ORDERA.ddl

```
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA1;
CREATE TABLE NEW_ORDERA1
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_001
INDEX IN ts_newordA_001
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 1 ENDING AT 584,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA2;
CREATE TABLE NEW_ORDERA2
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_002
INDEX IN ts_newordA_002
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 585 ENDING AT 1168,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA3;
CREATE TABLE NEW_ORDERA3
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_003
INDEX IN ts_newordA_003
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 1169 ENDING AT 1752,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA4;
CREATE TABLE NEW_ORDERA4
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_004
INDEX IN ts_newordA_004
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 1753 ENDING AT 2336,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
```

```
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA5;
CREATE TABLE NEW_ORDERA5
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_005
INDEX IN ts_newordA_005
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 2337 ENDING AT 2920,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA6;
CREATE TABLE NEW_ORDERA6
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_006
INDEX IN ts_newordA_006
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 2921 ENDING AT 3504,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA7;
CREATE TABLE NEW_ORDERA7
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_007
INDEX IN ts_newordA_007
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 3505 ENDING AT 4088,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA8;
CREATE TABLE NEW_ORDERA8
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_008
INDEX IN ts_newordA_008
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 4089 ENDING AT 4672,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
```

```
DROP TABLE NEW_ORDERA9;
CREATE TABLE NEW_ORDERA9
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_009
INDEX IN ts_newordA_009
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 4673 ENDING AT 5256,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA10;
CREATE TABLE NEW_ORDERA10
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_010
INDEX IN ts_newordA_010
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 5257 ENDING AT 5840,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA11;
CREATE TABLE NEW_ORDERA11
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_011
INDEX IN ts_newordA_011
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 5841 ENDING AT 6424,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA12;
CREATE TABLE NEW_ORDERA12
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_012
INDEX IN ts_newordA_012
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 6425 ENDING AT 7008,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA13;
CREATE TABLE NEW_ORDERA13
```

```

(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT   NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_013
INDEX IN ts_newordA_013
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 7009 ENDING AT 7592,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA14;
CREATE TABLE NEW_ORDERA14
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT   NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_014
INDEX IN ts_newordA_014
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 7593 ENDING AT 8176,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA15;
CREATE TABLE NEW_ORDERA15
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT   NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_015
INDEX IN ts_newordA_015
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 8177 ENDING AT 8760,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA16;
CREATE TABLE NEW_ORDERA16
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT   NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_016
INDEX IN ts_newordA_016
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 8761 ENDING AT 9344,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA17;
CREATE TABLE NEW_ORDERA17
(
NO_O_ID    INTEGER    NOT NULL,

```

```

NO_D_ID    SMALLINT   NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_017
INDEX IN ts_newordA_017
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 9345 ENDING AT 9928,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA18;
CREATE TABLE NEW_ORDERA18
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT   NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_018
INDEX IN ts_newordA_018
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 9929 ENDING AT 10512,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA19;
CREATE TABLE NEW_ORDERA19
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT   NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_019
INDEX IN ts_newordA_019
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 10513 ENDING AT 11096,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA20;
CREATE TABLE NEW_ORDERA20
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT   NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_020
INDEX IN ts_newordA_020
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 11097 ENDING AT 11680,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA21;
CREATE TABLE NEW_ORDERA21
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT   NOT NULL,
NO_W_ID    INTEGER    NOT NULL

```

```

)
IN ts_newordA_021
INDEX IN ts_newordA_021
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 11681 ENDING AT 12264,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA22;
CREATE TABLE NEW_ORDERA22
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT   NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_022
INDEX IN ts_newordA_022
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 12265 ENDING AT 12848,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA23;
CREATE TABLE NEW_ORDERA23
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT   NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_023
INDEX IN ts_newordA_023
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 12849 ENDING AT 13432,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA24;
CREATE TABLE NEW_ORDERA24
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT   NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_024
INDEX IN ts_newordA_024
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 13433 ENDING AT 14016,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA25;
CREATE TABLE NEW_ORDERA25
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT   NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_025

```



```

ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA38;
CREATE TABLE NEW_ORDERA38
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT  NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_038
INDEX IN ts_newordA_038
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 21609 ENDING AT 22192,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA39;
CREATE TABLE NEW_ORDERA39
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT  NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_039
INDEX IN ts_newordA_039
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 22193 ENDING AT 22776,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA40;
CREATE TABLE NEW_ORDERA40
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT  NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_040
INDEX IN ts_newordA_040
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 22777 ENDING AT 23360,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA41;
CREATE TABLE NEW_ORDERA41
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT  NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_041
INDEX IN ts_newordA_041
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 23361 ENDING AT 23944,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;

```

```

connect to TPCC in share mode;
DROP TABLE NEW_ORDERA42;
CREATE TABLE NEW_ORDERA42
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT  NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_042
INDEX IN ts_newordA_042
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 23945 ENDING AT 24528,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA43;
CREATE TABLE NEW_ORDERA43
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT  NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_043
INDEX IN ts_newordA_043
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 24529 ENDING AT 25112,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA44;
CREATE TABLE NEW_ORDERA44
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT  NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_044
INDEX IN ts_newordA_044
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 25113 ENDING AT 25696,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA45;
CREATE TABLE NEW_ORDERA45
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT  NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_045
INDEX IN ts_newordA_045
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 25697 ENDING AT 26280,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA46;

```

```

CREATE TABLE NEW_ORDERA46
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT  NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_046
INDEX IN ts_newordA_046
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 26281 ENDING AT 26864,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA47;
CREATE TABLE NEW_ORDERA47
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT  NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_047
INDEX IN ts_newordA_047
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 26865 ENDING AT 27448,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERA48;
CREATE TABLE NEW_ORDERA48
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT  NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordA_048
INDEX IN ts_newordA_048
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 27449 ENDING AT 28032,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 1900 ENDING AT 3675
)
ALLOW OVERFLOW;
connect reset;
DDL/CRTB NEW ORDERB.ddl
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB1;
CREATE TABLE NEW_ORDERB1
(
NO_O_ID    INTEGER    NOT NULL,
NO_D_ID    SMALLINT  NOT NULL,
NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordB_001
INDEX IN ts_newordB_001
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 1 ENDING AT 584,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;

```

```

DROP TABLE NEW_ORDERB2;
CREATE TABLE NEW_ORDERB2
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordB_002
INDEX IN ts_newordB_002
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 585 ENDING AT 1168,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB3;
CREATE TABLE NEW_ORDERB3
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordB_003
INDEX IN ts_newordB_003
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 1169 ENDING AT 1752,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB4;
CREATE TABLE NEW_ORDERB4
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordB_004
INDEX IN ts_newordB_004
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 1753 ENDING AT 2336,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB5;
CREATE TABLE NEW_ORDERB5
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordB_005
INDEX IN ts_newordB_005
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 2337 ENDING AT 2920,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB6;
CREATE TABLE NEW_ORDERB6

```

```

(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordB_006
INDEX IN ts_newordB_006
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 2921 ENDING AT 3504,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB7;
CREATE TABLE NEW_ORDERB7
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordB_007
INDEX IN ts_newordB_007
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 3505 ENDING AT 4088,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB8;
CREATE TABLE NEW_ORDERB8
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordB_008
INDEX IN ts_newordB_008
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 4089 ENDING AT 4672,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB9;
CREATE TABLE NEW_ORDERB9
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordB_009
INDEX IN ts_newordB_009
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 4673 ENDING AT 5256,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB10;
CREATE TABLE NEW_ORDERB10
(
  NO_O_ID    INTEGER    NOT NULL,

```

```

  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordB_010
INDEX IN ts_newordB_010
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 5257 ENDING AT 5840,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB11;
CREATE TABLE NEW_ORDERB11
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordB_011
INDEX IN ts_newordB_011
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 5841 ENDING AT 6424,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB12;
CREATE TABLE NEW_ORDERB12
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordB_012
INDEX IN ts_newordB_012
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 6425 ENDING AT 7008,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB13;
CREATE TABLE NEW_ORDERB13
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL
)
IN ts_newordB_013
INDEX IN ts_newordB_013
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 7009 ENDING AT 7592,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB14;
CREATE TABLE NEW_ORDERB14
(
  NO_O_ID    INTEGER    NOT NULL,
  NO_D_ID    SMALLINT   NOT NULL,
  NO_W_ID    INTEGER    NOT NULL

```

```

)
IN ts_newordB_014
INDEX IN ts_newordB_014
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 7593 ENDING AT 8176,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;
)
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB15;
CREATE TABLE NEW_ORDERB15
(
NO_O_ID INTEGER NOT NULL,
NO_D_ID SMALLINT NOT NULL,
NO_W_ID INTEGER NOT NULL
)
IN ts_newordB_015
INDEX IN ts_newordB_015
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 8177 ENDING AT 8760,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;
)
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB16;
CREATE TABLE NEW_ORDERB16
(
NO_O_ID INTEGER NOT NULL,
NO_D_ID SMALLINT NOT NULL,
NO_W_ID INTEGER NOT NULL
)
IN ts_newordB_016
INDEX IN ts_newordB_016
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 8761 ENDING AT 9344,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;
)
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB17;
CREATE TABLE NEW_ORDERB17
(
NO_O_ID INTEGER NOT NULL,
NO_D_ID SMALLINT NOT NULL,
NO_W_ID INTEGER NOT NULL
)
IN ts_newordB_017
INDEX IN ts_newordB_017
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 9345 ENDING AT 9928,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;
)
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB18;
CREATE TABLE NEW_ORDERB18
(
NO_O_ID INTEGER NOT NULL,
NO_D_ID SMALLINT NOT NULL,
NO_W_ID INTEGER NOT NULL
)
IN ts_newordB_018

```

```

INDEX IN ts_newordB_018
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 9929 ENDING AT 10512,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;
)
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB19;
CREATE TABLE NEW_ORDERB19
(
NO_O_ID INTEGER NOT NULL,
NO_D_ID SMALLINT NOT NULL,
NO_W_ID INTEGER NOT NULL
)
IN ts_newordB_019
INDEX IN ts_newordB_019
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 10513 ENDING AT 11096,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;
)
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB20;
CREATE TABLE NEW_ORDERB20
(
NO_O_ID INTEGER NOT NULL,
NO_D_ID SMALLINT NOT NULL,
NO_W_ID INTEGER NOT NULL
)
IN ts_newordB_020
INDEX IN ts_newordB_020
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 11097 ENDING AT 11680,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;
)
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB21;
CREATE TABLE NEW_ORDERB21
(
NO_O_ID INTEGER NOT NULL,
NO_D_ID SMALLINT NOT NULL,
NO_W_ID INTEGER NOT NULL
)
IN ts_newordB_021
INDEX IN ts_newordB_021
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 11681 ENDING AT 12264,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;
)
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB22;
CREATE TABLE NEW_ORDERB22
(
NO_O_ID INTEGER NOT NULL,
NO_D_ID SMALLINT NOT NULL,
NO_W_ID INTEGER NOT NULL
)
IN ts_newordB_022
INDEX IN ts_newordB_022
ORGANIZE BY KEY SEQUENCE (

```

```

NO_W_ID STARTING FROM 12265 ENDING AT 12848,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;
)
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB23;
CREATE TABLE NEW_ORDERB23
(
NO_O_ID INTEGER NOT NULL,
NO_D_ID SMALLINT NOT NULL,
NO_W_ID INTEGER NOT NULL
)
IN ts_newordB_023
INDEX IN ts_newordB_023
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 12849 ENDING AT 13432,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;
)
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB24;
CREATE TABLE NEW_ORDERB24
(
NO_O_ID INTEGER NOT NULL,
NO_D_ID SMALLINT NOT NULL,
NO_W_ID INTEGER NOT NULL
)
IN ts_newordB_024
INDEX IN ts_newordB_024
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 13433 ENDING AT 14016,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;
)
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB25;
CREATE TABLE NEW_ORDERB25
(
NO_O_ID INTEGER NOT NULL,
NO_D_ID SMALLINT NOT NULL,
NO_W_ID INTEGER NOT NULL
)
IN ts_newordB_025
INDEX IN ts_newordB_025
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 14017 ENDING AT 14600,
NO_D_ID STARTING FROM 1 ENDING AT 10,
NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;
)
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB26;
CREATE TABLE NEW_ORDERB26
(
NO_O_ID INTEGER NOT NULL,
NO_D_ID SMALLINT NOT NULL,
NO_W_ID INTEGER NOT NULL
)
IN ts_newordB_026
INDEX IN ts_newordB_026
ORGANIZE BY KEY SEQUENCE (
NO_W_ID STARTING FROM 14601 ENDING AT 15184,
NO_D_ID STARTING FROM 1 ENDING AT 10,

```



```

CREATE TABLE NEW_ORDERB39
(
  NO_O_ID   INTEGER NOT NULL,
  NO_D_ID   SMALLINT NOT NULL,
  NO_W_ID   INTEGER NOT NULL
)
IN ts_newordB_039
INDEX IN ts_newordB_039
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 22193 ENDING AT 22776,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB40;
CREATE TABLE NEW_ORDERB40
(
  NO_O_ID   INTEGER NOT NULL,
  NO_D_ID   SMALLINT NOT NULL,
  NO_W_ID   INTEGER NOT NULL
)
IN ts_newordB_040
INDEX IN ts_newordB_040
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 22777 ENDING AT 23360,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB41;
CREATE TABLE NEW_ORDERB41
(
  NO_O_ID   INTEGER NOT NULL,
  NO_D_ID   SMALLINT NOT NULL,
  NO_W_ID   INTEGER NOT NULL
)
IN ts_newordB_041
INDEX IN ts_newordB_041
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 23361 ENDING AT 23944,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB42;
CREATE TABLE NEW_ORDERB42
(
  NO_O_ID   INTEGER NOT NULL,
  NO_D_ID   SMALLINT NOT NULL,
  NO_W_ID   INTEGER NOT NULL
)
IN ts_newordB_042
INDEX IN ts_newordB_042
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 23945 ENDING AT 24528,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB43;
CREATE TABLE NEW_ORDERB43
(

```

```

  NO_O_ID   INTEGER NOT NULL,
  NO_D_ID   SMALLINT NOT NULL,
  NO_W_ID   INTEGER NOT NULL
)
IN ts_newordB_043
INDEX IN ts_newordB_043
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 24529 ENDING AT 25112,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB44;
CREATE TABLE NEW_ORDERB44
(
  NO_O_ID   INTEGER NOT NULL,
  NO_D_ID   SMALLINT NOT NULL,
  NO_W_ID   INTEGER NOT NULL
)
IN ts_newordB_044
INDEX IN ts_newordB_044
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 25113 ENDING AT 25696,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB45;
CREATE TABLE NEW_ORDERB45
(
  NO_O_ID   INTEGER NOT NULL,
  NO_D_ID   SMALLINT NOT NULL,
  NO_W_ID   INTEGER NOT NULL
)
IN ts_newordB_045
INDEX IN ts_newordB_045
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 25697 ENDING AT 26280,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB46;
CREATE TABLE NEW_ORDERB46
(
  NO_O_ID   INTEGER NOT NULL,
  NO_D_ID   SMALLINT NOT NULL,
  NO_W_ID   INTEGER NOT NULL
)
IN ts_newordB_046
INDEX IN ts_newordB_046
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 26281 ENDING AT 26864,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB47;
CREATE TABLE NEW_ORDERB47
(
  NO_O_ID   INTEGER NOT NULL,
  NO_D_ID   SMALLINT NOT NULL,

```

```

  NO_W_ID   INTEGER NOT NULL
)
IN ts_newordB_047
INDEX IN ts_newordB_047
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 26865 ENDING AT 27448,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDERB48;
CREATE TABLE NEW_ORDERB48
(
  NO_O_ID   INTEGER NOT NULL,
  NO_D_ID   SMALLINT NOT NULL,
  NO_W_ID   INTEGER NOT NULL
)
IN ts_newordB_048
INDEX IN ts_newordB_048
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 27449 ENDING AT 28032,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 3676 ENDING AT 5451
)
ALLOW OVERFLOW;

connect reset;

DDL/CRTB_ORDERS.ddl

connect to TPCC in share mode;
DROP TABLE ORDERS1;
CREATE TABLE ORDERS1
(
  O_C_ID   INTEGER NOT NULL,
  O_ENTRY_D  TIMESTAMP NOT NULL,
  O_CARRIER_ID SMALLINT NOT NULL,
  O_OL_CNT  SMALLINT NOT NULL,
  O_ALL_LOCAL SMALLINT NOT NULL,
  O_ID      INTEGER NOT NULL,
  O_W_ID    INTEGER NOT NULL,
  O_D_ID    SMALLINT NOT NULL
)
IN ts_order_001
INDEX IN is_order_001
ORGANIZE BY KEY SEQUENCE (
  O_ID STARTING FROM 1 ENDING AT 3675,
  O_W_ID STARTING FROM 1 ENDING AT 584,
  O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS2;
CREATE TABLE ORDERS2
(
  O_C_ID   INTEGER NOT NULL,
  O_ENTRY_D  TIMESTAMP NOT NULL,
  O_CARRIER_ID SMALLINT NOT NULL,
  O_OL_CNT  SMALLINT NOT NULL,
  O_ALL_LOCAL SMALLINT NOT NULL,
  O_ID      INTEGER NOT NULL,
  O_W_ID    INTEGER NOT NULL,
  O_D_ID    SMALLINT NOT NULL
)
IN ts_order_002
INDEX IN is_order_002
ORGANIZE BY KEY SEQUENCE (
  O_ID STARTING FROM 1 ENDING AT 3675,

```

```

O_W_ID STARTING FROM 585 ENDING AT 1168,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS3;
CREATE TABLE ORDERS3
(
O_C_ID INTEGER NOT NULL,
O_ENTRY_D TIMESTAMP NOT NULL,
O_CARRIER_ID SMALLINT NOT NULL,
O_OL_CNT SMALLINT NOT NULL,
O_ALL_LOCAL SMALLINT NOT NULL,
O_ID INTEGER NOT NULL,
O_W_ID INTEGER NOT NULL,
O_D_ID SMALLINT NOT NULL
)
IN ts_order_003
INDEX IN is_order_003
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3675,
O_W_ID STARTING FROM 1169 ENDING AT 1752,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS4;
CREATE TABLE ORDERS4
(
O_C_ID INTEGER NOT NULL,
O_ENTRY_D TIMESTAMP NOT NULL,
O_CARRIER_ID SMALLINT NOT NULL,
O_OL_CNT SMALLINT NOT NULL,
O_ALL_LOCAL SMALLINT NOT NULL,
O_ID INTEGER NOT NULL,
O_W_ID INTEGER NOT NULL,
O_D_ID SMALLINT NOT NULL
)
IN ts_order_004
INDEX IN is_order_004
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3675,
O_W_ID STARTING FROM 1753 ENDING AT 2336,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS5;
CREATE TABLE ORDERS5
(
O_C_ID INTEGER NOT NULL,
O_ENTRY_D TIMESTAMP NOT NULL,
O_CARRIER_ID SMALLINT NOT NULL,
O_OL_CNT SMALLINT NOT NULL,
O_ALL_LOCAL SMALLINT NOT NULL,
O_ID INTEGER NOT NULL,
O_W_ID INTEGER NOT NULL,
O_D_ID SMALLINT NOT NULL
)
IN ts_order_005
INDEX IN is_order_005
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3675,
O_W_ID STARTING FROM 2337 ENDING AT 2920,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS6;
CREATE TABLE ORDERS6
(
O_C_ID INTEGER NOT NULL,
O_ENTRY_D TIMESTAMP NOT NULL,
O_CARRIER_ID SMALLINT NOT NULL,
O_OL_CNT SMALLINT NOT NULL,
O_ALL_LOCAL SMALLINT NOT NULL,
O_ID INTEGER NOT NULL,
O_W_ID INTEGER NOT NULL,
O_D_ID SMALLINT NOT NULL
)
IN ts_order_006
INDEX IN is_order_006
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3675,
O_W_ID STARTING FROM 2921 ENDING AT 3504,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS7;
CREATE TABLE ORDERS7
(
O_C_ID INTEGER NOT NULL,
O_ENTRY_D TIMESTAMP NOT NULL,
O_CARRIER_ID SMALLINT NOT NULL,
O_OL_CNT SMALLINT NOT NULL,
O_ALL_LOCAL SMALLINT NOT NULL,
O_ID INTEGER NOT NULL,
O_W_ID INTEGER NOT NULL,
O_D_ID SMALLINT NOT NULL
)
IN ts_order_007
INDEX IN is_order_007
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3675,
O_W_ID STARTING FROM 3505 ENDING AT 4088,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS8;
CREATE TABLE ORDERS8
(
O_C_ID INTEGER NOT NULL,
O_ENTRY_D TIMESTAMP NOT NULL,
O_CARRIER_ID SMALLINT NOT NULL,
O_OL_CNT SMALLINT NOT NULL,
O_ALL_LOCAL SMALLINT NOT NULL,
O_ID INTEGER NOT NULL,
O_W_ID INTEGER NOT NULL,
O_D_ID SMALLINT NOT NULL
)
IN ts_order_008
INDEX IN is_order_008
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3675,
O_W_ID STARTING FROM 4089 ENDING AT 4672,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS9;
CREATE TABLE ORDERS9

```

```

(
O_C_ID INTEGER NOT NULL,
O_ENTRY_D TIMESTAMP NOT NULL,
O_CARRIER_ID SMALLINT NOT NULL,
O_OL_CNT SMALLINT NOT NULL,
O_ALL_LOCAL SMALLINT NOT NULL,
O_ID INTEGER NOT NULL,
O_W_ID INTEGER NOT NULL,
O_D_ID SMALLINT NOT NULL
)
IN ts_order_009
INDEX IN is_order_009
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3675,
O_W_ID STARTING FROM 4673 ENDING AT 5256,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS10;
CREATE TABLE ORDERS10
(
O_C_ID INTEGER NOT NULL,
O_ENTRY_D TIMESTAMP NOT NULL,
O_CARRIER_ID SMALLINT NOT NULL,
O_OL_CNT SMALLINT NOT NULL,
O_ALL_LOCAL SMALLINT NOT NULL,
O_ID INTEGER NOT NULL,
O_W_ID INTEGER NOT NULL,
O_D_ID SMALLINT NOT NULL
)
IN ts_order_010
INDEX IN is_order_010
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3675,
O_W_ID STARTING FROM 5257 ENDING AT 5840,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS11;
CREATE TABLE ORDERS11
(
O_C_ID INTEGER NOT NULL,
O_ENTRY_D TIMESTAMP NOT NULL,
O_CARRIER_ID SMALLINT NOT NULL,
O_OL_CNT SMALLINT NOT NULL,
O_ALL_LOCAL SMALLINT NOT NULL,
O_ID INTEGER NOT NULL,
O_W_ID INTEGER NOT NULL,
O_D_ID SMALLINT NOT NULL
)
IN ts_order_011
INDEX IN is_order_011
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3675,
O_W_ID STARTING FROM 5841 ENDING AT 6424,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS12;
CREATE TABLE ORDERS12
(
O_C_ID INTEGER NOT NULL,
O_ENTRY_D TIMESTAMP NOT NULL,
O_CARRIER_ID SMALLINT NOT NULL,

```



```

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS41;
CREATE TABLE ORDERS41
(
  O_C_ID INTEGER NOT NULL,
  O_ENTRY_D TIMESTAMP NOT NULL,
  O_CARRIER_ID SMALLINT NOT NULL,
  O_OL_CNT SMALLINT NOT NULL,
  O_ALL_LOCAL SMALLINT NOT NULL,
  O_ID INTEGER NOT NULL,
  O_W_ID INTEGER NOT NULL,
  O_D_ID SMALLINT NOT NULL
)
IN ts_order_041
INDEX IN is_order_041
ORGANIZE BY KEY SEQUENCE (
  O_ID STARTING FROM 1 ENDING AT 3675,
  O_W_ID STARTING FROM 23361 ENDING AT 23944,
  O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS42;
CREATE TABLE ORDERS42

```

```

(
  O_C_ID INTEGER NOT NULL,
  O_ENTRY_D TIMESTAMP NOT NULL,
  O_CARRIER_ID SMALLINT NOT NULL,
  O_OL_CNT SMALLINT NOT NULL,
  O_ALL_LOCAL SMALLINT NOT NULL,
  O_ID INTEGER NOT NULL,
  O_W_ID INTEGER NOT NULL,
  O_D_ID SMALLINT NOT NULL
)
IN ts_order_042
INDEX IN is_order_042
ORGANIZE BY KEY SEQUENCE (
  O_ID STARTING FROM 1 ENDING AT 3675,
  O_W_ID STARTING FROM 23945 ENDING AT 24528,
  O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS43;
CREATE TABLE ORDERS43

```

```

(
  O_C_ID INTEGER NOT NULL,
  O_ENTRY_D TIMESTAMP NOT NULL,
  O_CARRIER_ID SMALLINT NOT NULL,
  O_OL_CNT SMALLINT NOT NULL,
  O_ALL_LOCAL SMALLINT NOT NULL,
  O_ID INTEGER NOT NULL,
  O_W_ID INTEGER NOT NULL,
  O_D_ID SMALLINT NOT NULL
)
IN ts_order_043
INDEX IN is_order_043
ORGANIZE BY KEY SEQUENCE (
  O_ID STARTING FROM 1 ENDING AT 3675,
  O_W_ID STARTING FROM 24529 ENDING AT 25112,
  O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS44;
CREATE TABLE ORDERS44

```

```

(
  O_C_ID INTEGER NOT NULL,
  O_ENTRY_D TIMESTAMP NOT NULL,
  O_CARRIER_ID SMALLINT NOT NULL,
  O_OL_CNT SMALLINT NOT NULL,
  O_ALL_LOCAL SMALLINT NOT NULL,
  O_ID INTEGER NOT NULL,
  O_W_ID INTEGER NOT NULL,
  O_D_ID SMALLINT NOT NULL
)
IN ts_order_044
INDEX IN is_order_044
ORGANIZE BY KEY SEQUENCE (
  O_ID STARTING FROM 1 ENDING AT 3675,
  O_W_ID STARTING FROM 25113 ENDING AT 25696,
  O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS45;
CREATE TABLE ORDERS45

```

```

(
  O_C_ID INTEGER NOT NULL,
  O_ENTRY_D TIMESTAMP NOT NULL,
  O_CARRIER_ID SMALLINT NOT NULL,
  O_OL_CNT SMALLINT NOT NULL,
  O_ALL_LOCAL SMALLINT NOT NULL,
  O_ID INTEGER NOT NULL,
  O_W_ID INTEGER NOT NULL,
  O_D_ID SMALLINT NOT NULL
)
IN ts_order_045
INDEX IN is_order_045
ORGANIZE BY KEY SEQUENCE (
  O_ID STARTING FROM 1 ENDING AT 3675,
  O_W_ID STARTING FROM 25697 ENDING AT 26280,
  O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS46;
CREATE TABLE ORDERS46

```

```

(
  O_C_ID INTEGER NOT NULL,
  O_ENTRY_D TIMESTAMP NOT NULL,
  O_CARRIER_ID SMALLINT NOT NULL,
  O_OL_CNT SMALLINT NOT NULL,
  O_ALL_LOCAL SMALLINT NOT NULL,
  O_ID INTEGER NOT NULL,
  O_W_ID INTEGER NOT NULL,
  O_D_ID SMALLINT NOT NULL
)
IN ts_order_046
INDEX IN is_order_046
ORGANIZE BY KEY SEQUENCE (
  O_ID STARTING FROM 1 ENDING AT 3675,
  O_W_ID STARTING FROM 26281 ENDING AT 26864,
  O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS47;
CREATE TABLE ORDERS47

```

```

(
  O_C_ID INTEGER NOT NULL,
  O_ENTRY_D TIMESTAMP NOT NULL,
  O_CARRIER_ID SMALLINT NOT NULL,

```

```

  O_OL_CNT SMALLINT NOT NULL,
  O_ALL_LOCAL SMALLINT NOT NULL,
  O_ID INTEGER NOT NULL,
  O_W_ID INTEGER NOT NULL,
  O_D_ID SMALLINT NOT NULL
)

```

```

IN ts_order_047
INDEX IN is_order_047
ORGANIZE BY KEY SEQUENCE (
  O_ID STARTING FROM 1 ENDING AT 3675,
  O_W_ID STARTING FROM 26865 ENDING AT 27448,
  O_D_ID STARTING FROM 1 ENDING AT 10
)

```

ALLOW OVERFLOW;

```

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS48;
CREATE TABLE ORDERS48

```

```

(
  O_C_ID INTEGER NOT NULL,
  O_ENTRY_D TIMESTAMP NOT NULL,
  O_CARRIER_ID SMALLINT NOT NULL,
  O_OL_CNT SMALLINT NOT NULL,
  O_ALL_LOCAL SMALLINT NOT NULL,
  O_ID INTEGER NOT NULL,
  O_W_ID INTEGER NOT NULL,
  O_D_ID SMALLINT NOT NULL
)

```

```

IN ts_order_048
INDEX IN is_order_048
ORGANIZE BY KEY SEQUENCE (
  O_ID STARTING FROM 1 ENDING AT 3675,
  O_W_ID STARTING FROM 27449 ENDING AT 28032,
  O_D_ID STARTING FROM 1 ENDING AT 10
)

```

ALLOW OVERFLOW;

connect reset;

DDL/CRTB ORDER LINE.ddl

```

connect to TPCC in share mode;
DROP TABLE ORDER_LINE1;
CREATE TABLE ORDER_LINE1

```

```

(
  OL_DELIVERY_D TIMESTAMP NOT NULL,
  OL_AMOUNT DECIMAL(6,2) NOT NULL,
  OL_I_ID INTEGER NOT NULL,
  OL_SUPPLY_W_ID INTEGER NOT NULL,
  OL_QUANTITY SMALLINT NOT NULL,
  OL_DIST_INFO CHAR(24) NOT NULL,
  OL_O_ID INTEGER NOT NULL,
  OL_D_ID SMALLINT NOT NULL,
  OL_W_ID INTEGER NOT NULL,
  OL_NUMBER SMALLINT NOT NULL
)

```

```

IN ts_orderline_001
INDEX IN ts_orderline_001
ORGANIZE BY KEY SEQUENCE (
  OL_W_ID STARTING FROM 1 ENDING AT 584,
  OL_D_ID STARTING FROM 1 ENDING AT 10,
  OL_O_ID STARTING FROM 1 ENDING AT 3675,
  OL_NUMBER STARTING FROM 1 ENDING AT 15
)

```

ALLOW OVERFLOW;

```

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDER_LINE2;
CREATE TABLE ORDER_LINE2

```

```

(
  OL_DELIVERY_D TIMESTAMP NOT NULL,

```



```

OL_AMOUNT DECIMAL(6,2) NOT NULL,
OL_I_ID INTEGER NOT NULL,
OL_SUPPLY_W_ID INTEGER NOT NULL,
OL_QUANTITY SMALLINT NOT NULL,
OL_DIST_INFO CHAR(24) NOT NULL,
OL_O_ID INTEGER NOT NULL,
OL_D_ID SMALLINT NOT NULL,
OL_W_ID INTEGER NOT NULL,
OL_NUMBER SMALLINT NOT NULL
)
IN ts_orderline_044
INDEX IN ts_orderline_044
ORGANIZE BY KEY SEQUENCE (
  OL_W_ID STARTING FROM 25113 ENDING AT 25696,
  OL_D_ID STARTING FROM 1 ENDING AT 10,
  OL_O_ID STARTING FROM 1 ENDING AT 3675,
  OL_NUMBER STARTING FROM 1 ENDING AT 15
)

```

ALLOW OVERFLOW;

```

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDER_LINE45;
CREATE TABLE ORDER_LINE45

```

```

(
  OL_DELIVERY_D TIMESTAMP NOT NULL,
  OL_AMOUNT DECIMAL(6,2) NOT NULL,
  OL_I_ID INTEGER NOT NULL,
  OL_SUPPLY_W_ID INTEGER NOT NULL,
  OL_QUANTITY SMALLINT NOT NULL,
  OL_DIST_INFO CHAR(24) NOT NULL,
  OL_O_ID INTEGER NOT NULL,
  OL_D_ID SMALLINT NOT NULL,
  OL_W_ID INTEGER NOT NULL,
  OL_NUMBER SMALLINT NOT NULL
)

```

```

IN ts_orderline_045
INDEX IN ts_orderline_045
ORGANIZE BY KEY SEQUENCE (
  OL_W_ID STARTING FROM 25697 ENDING AT 26280,
  OL_D_ID STARTING FROM 1 ENDING AT 10,
  OL_O_ID STARTING FROM 1 ENDING AT 3675,
  OL_NUMBER STARTING FROM 1 ENDING AT 15
)

```

ALLOW OVERFLOW;

```

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDER_LINE46;
CREATE TABLE ORDER_LINE46

```

```

(
  OL_DELIVERY_D TIMESTAMP NOT NULL,
  OL_AMOUNT DECIMAL(6,2) NOT NULL,
  OL_I_ID INTEGER NOT NULL,
  OL_SUPPLY_W_ID INTEGER NOT NULL,
  OL_QUANTITY SMALLINT NOT NULL,
  OL_DIST_INFO CHAR(24) NOT NULL,
  OL_O_ID INTEGER NOT NULL,
  OL_D_ID SMALLINT NOT NULL,
  OL_W_ID INTEGER NOT NULL,
  OL_NUMBER SMALLINT NOT NULL
)

```

```

IN ts_orderline_046
INDEX IN ts_orderline_046
ORGANIZE BY KEY SEQUENCE (
  OL_W_ID STARTING FROM 26281 ENDING AT 26864,
  OL_D_ID STARTING FROM 1 ENDING AT 10,
  OL_O_ID STARTING FROM 1 ENDING AT 3675,
  OL_NUMBER STARTING FROM 1 ENDING AT 15
)

```

ALLOW OVERFLOW;

connect reset;

```

connect to TPCC in share mode;
DROP TABLE ORDER_LINE47;
CREATE TABLE ORDER_LINE47

```

```

(
  OL_DELIVERY_D TIMESTAMP NOT NULL,
  OL_AMOUNT DECIMAL(6,2) NOT NULL,
  OL_I_ID INTEGER NOT NULL,
  OL_SUPPLY_W_ID INTEGER NOT NULL,
  OL_QUANTITY SMALLINT NOT NULL,
  OL_DIST_INFO CHAR(24) NOT NULL,
  OL_O_ID INTEGER NOT NULL,
  OL_D_ID SMALLINT NOT NULL,
  OL_W_ID INTEGER NOT NULL,
  OL_NUMBER SMALLINT NOT NULL
)

```

```

IN ts_orderline_047
INDEX IN ts_orderline_047
ORGANIZE BY KEY SEQUENCE (
  OL_W_ID STARTING FROM 26865 ENDING AT 27448,
  OL_D_ID STARTING FROM 1 ENDING AT 10,
  OL_O_ID STARTING FROM 1 ENDING AT 3675,
  OL_NUMBER STARTING FROM 1 ENDING AT 15
)

```

ALLOW OVERFLOW;

```

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDER_LINE48;
CREATE TABLE ORDER_LINE48

```

```

(
  OL_DELIVERY_D TIMESTAMP NOT NULL,
  OL_AMOUNT DECIMAL(6,2) NOT NULL,
  OL_I_ID INTEGER NOT NULL,
  OL_SUPPLY_W_ID INTEGER NOT NULL,
  OL_QUANTITY SMALLINT NOT NULL,
  OL_DIST_INFO CHAR(24) NOT NULL,
  OL_O_ID INTEGER NOT NULL,
  OL_D_ID SMALLINT NOT NULL,
  OL_W_ID INTEGER NOT NULL,
  OL_NUMBER SMALLINT NOT NULL
)

```

```

IN ts_orderline_048
INDEX IN ts_orderline_048
ORGANIZE BY KEY SEQUENCE (
  OL_W_ID STARTING FROM 27449 ENDING AT 28032,
  OL_D_ID STARTING FROM 1 ENDING AT 10,
  OL_O_ID STARTING FROM 1 ENDING AT 3675,
  OL_NUMBER STARTING FROM 1 ENDING AT 15
)

```

ALLOW OVERFLOW;

connect reset;

DDL/CRTB_STOCK.ddl

```

connect to TPCC in share mode;
DROP TABLE STOCK1;
CREATE TABLE STOCK1

```

```

(
  S_REMOTE_CNT INTEGER NOT NULL,
  S_QUANTITY INTEGER NOT NULL,
  S_ORDER_CNT INTEGER NOT NULL,
  S_YTD INTEGER NOT NULL,
  S_DATA VARCHAR(50) NOT NULL,
  S_DIST_01 CHAR(24) NOT NULL,
  S_DIST_02 CHAR(24) NOT NULL,
  S_DIST_03 CHAR(24) NOT NULL,
  S_DIST_04 CHAR(24) NOT NULL,
  S_DIST_05 CHAR(24) NOT NULL,
  S_DIST_06 CHAR(24) NOT NULL,
  S_DIST_07 CHAR(24) NOT NULL,
  S_DIST_08 CHAR(24) NOT NULL,

```

```

S_DIST_09 CHAR(24) NOT NULL,
S_DIST_10 CHAR(24) NOT NULL,
S_I_ID INTEGER NOT NULL,
S_W_ID INTEGER NOT NULL
)

```

```

IN ts_stock_001
INDEX IN ts_stock_001
ORGANIZE BY KEY SEQUENCE (
  S_I_ID STARTING FROM 1 ENDING AT 100000,
  S_W_ID STARTING FROM 1 ENDING AT 584
)

```

ALLOW OVERFLOW;

```

connect reset;
connect to TPCC in share mode;
DROP TABLE STOCK2;
CREATE TABLE STOCK2

```

```

(
  S_REMOTE_CNT INTEGER NOT NULL,
  S_QUANTITY INTEGER NOT NULL,
  S_ORDER_CNT INTEGER NOT NULL,
  S_YTD INTEGER NOT NULL,
  S_DATA VARCHAR(50) NOT NULL,
  S_DIST_01 CHAR(24) NOT NULL,
  S_DIST_02 CHAR(24) NOT NULL,
  S_DIST_03 CHAR(24) NOT NULL,
  S_DIST_04 CHAR(24) NOT NULL,
  S_DIST_05 CHAR(24) NOT NULL,
  S_DIST_06 CHAR(24) NOT NULL,
  S_DIST_07 CHAR(24) NOT NULL,
  S_DIST_08 CHAR(24) NOT NULL,
  S_DIST_09 CHAR(24) NOT NULL,
  S_DIST_10 CHAR(24) NOT NULL,
  S_I_ID INTEGER NOT NULL,
  S_W_ID INTEGER NOT NULL
)

```

```

IN ts_stock_002
INDEX IN ts_stock_002
ORGANIZE BY KEY SEQUENCE (
  S_I_ID STARTING FROM 1 ENDING AT 100000,
  S_W_ID STARTING FROM 585 ENDING AT 1168
)

```

ALLOW OVERFLOW;

```

connect reset;
connect to TPCC in share mode;
DROP TABLE STOCK3;
CREATE TABLE STOCK3

```

```

(
  S_REMOTE_CNT INTEGER NOT NULL,
  S_QUANTITY INTEGER NOT NULL,
  S_ORDER_CNT INTEGER NOT NULL,
  S_YTD INTEGER NOT NULL,
  S_DATA VARCHAR(50) NOT NULL,
  S_DIST_01 CHAR(24) NOT NULL,
  S_DIST_02 CHAR(24) NOT NULL,
  S_DIST_03 CHAR(24) NOT NULL,
  S_DIST_04 CHAR(24) NOT NULL,
  S_DIST_05 CHAR(24) NOT NULL,
  S_DIST_06 CHAR(24) NOT NULL,
  S_DIST_07 CHAR(24) NOT NULL,
  S_DIST_08 CHAR(24) NOT NULL,
  S_DIST_09 CHAR(24) NOT NULL,
  S_DIST_10 CHAR(24) NOT NULL,
  S_I_ID INTEGER NOT NULL,
  S_W_ID INTEGER NOT NULL
)

```

```

IN ts_stock_003
INDEX IN ts_stock_003
ORGANIZE BY KEY SEQUENCE (
  S_I_ID STARTING FROM 1 ENDING AT 100000,
  S_W_ID STARTING FROM 1169 ENDING AT 1752
)

```



```

)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE STOCK46;
CREATE TABLE STOCK46
(
S_REMOTE_CNT INTEGER NOT NULL,
S_QUANTITY INTEGER NOT NULL,
S_ORDER_CNT INTEGER NOT NULL,
S_YTD INTEGER NOT NULL,
S_DATA VARCHAR(50) NOT NULL,
S_DIST_01 CHAR(24) NOT NULL,
S_DIST_02 CHAR(24) NOT NULL,
S_DIST_03 CHAR(24) NOT NULL,
S_DIST_04 CHAR(24) NOT NULL,
S_DIST_05 CHAR(24) NOT NULL,
S_DIST_06 CHAR(24) NOT NULL,
S_DIST_07 CHAR(24) NOT NULL,
S_DIST_08 CHAR(24) NOT NULL,
S_DIST_09 CHAR(24) NOT NULL,
S_DIST_10 CHAR(24) NOT NULL,
S_I_ID INTEGER NOT NULL,
S_W_ID INTEGER NOT NULL
)
IN ts_stock_046
INDEX IN ts_stock_046
ORGANIZE BY KEY SEQUENCE (
S_I_ID STARTING FROM 1 ENDING AT 100000,
S_W_ID STARTING FROM 26281 ENDING AT 26864
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE STOCK47;
CREATE TABLE STOCK47
(
S_REMOTE_CNT INTEGER NOT NULL,
S_QUANTITY INTEGER NOT NULL,
S_ORDER_CNT INTEGER NOT NULL,
S_YTD INTEGER NOT NULL,
S_DATA VARCHAR(50) NOT NULL,
S_DIST_01 CHAR(24) NOT NULL,
S_DIST_02 CHAR(24) NOT NULL,
S_DIST_03 CHAR(24) NOT NULL,
S_DIST_04 CHAR(24) NOT NULL,
S_DIST_05 CHAR(24) NOT NULL,
S_DIST_06 CHAR(24) NOT NULL,
S_DIST_07 CHAR(24) NOT NULL,
S_DIST_08 CHAR(24) NOT NULL,
S_DIST_09 CHAR(24) NOT NULL,
S_DIST_10 CHAR(24) NOT NULL,
S_I_ID INTEGER NOT NULL,
S_W_ID INTEGER NOT NULL
)
IN ts_stock_047
INDEX IN ts_stock_047
ORGANIZE BY KEY SEQUENCE (
S_I_ID STARTING FROM 1 ENDING AT 100000,
S_W_ID STARTING FROM 26865 ENDING AT 27448
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE STOCK48;
CREATE TABLE STOCK48
(
S_REMOTE_CNT INTEGER NOT NULL,
S_QUANTITY INTEGER NOT NULL,
S_ORDER_CNT INTEGER NOT NULL,

```

```

S_YTD INTEGER NOT NULL,
S_DATA VARCHAR(50) NOT NULL,
S_DIST_01 CHAR(24) NOT NULL,
S_DIST_02 CHAR(24) NOT NULL,
S_DIST_03 CHAR(24) NOT NULL,
S_DIST_04 CHAR(24) NOT NULL,
S_DIST_05 CHAR(24) NOT NULL,
S_DIST_06 CHAR(24) NOT NULL,
S_DIST_07 CHAR(24) NOT NULL,
S_DIST_08 CHAR(24) NOT NULL,
S_DIST_09 CHAR(24) NOT NULL,
S_DIST_10 CHAR(24) NOT NULL,
S_I_ID INTEGER NOT NULL,
S_W_ID INTEGER NOT NULL
)
IN ts_stock_048
INDEX IN ts_stock_048
ORGANIZE BY KEY SEQUENCE (
S_I_ID STARTING FROM 1 ENDING AT 100000,
S_W_ID STARTING FROM 27449 ENDING AT 28032
)
ALLOW OVERFLOW;
connect reset;

DDL/CRTB WAREHOUSE.ddl

connect to TPCC in share mode;
DROP TABLE WAREHOUSE1;
CREATE TABLE WAREHOUSE1
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_001
INDEX IN ts_wh_001
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 1 ENDING AT 584
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE2;
CREATE TABLE WAREHOUSE2
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_002
INDEX IN ts_wh_002
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 585 ENDING AT 1168
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE3;

```

```

CREATE TABLE WAREHOUSE3
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_003
INDEX IN ts_wh_003
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 1169 ENDING AT 1752
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE4;
CREATE TABLE WAREHOUSE4
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_004
INDEX IN ts_wh_004
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 1753 ENDING AT 2336
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE5;
CREATE TABLE WAREHOUSE5
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_005
INDEX IN ts_wh_005
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 2337 ENDING AT 2920
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE6;
CREATE TABLE WAREHOUSE6
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,

```



```

W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_006
INDEX IN ts_wh_006
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 2921 ENDING AT 3504
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE7;
CREATE TABLE WAREHOUSE7
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_007
INDEX IN ts_wh_007
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 3505 ENDING AT 4088
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE8;
CREATE TABLE WAREHOUSE8
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_008
INDEX IN ts_wh_008
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 4089 ENDING AT 4672
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE9;
CREATE TABLE WAREHOUSE9
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_009
INDEX IN ts_wh_009

```

```

ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 4673 ENDING AT 5256
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE10;
CREATE TABLE WAREHOUSE10
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_010
INDEX IN ts_wh_010
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 5257 ENDING AT 5840
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE11;
CREATE TABLE WAREHOUSE11
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_011
INDEX IN ts_wh_011
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 5841 ENDING AT 6424
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE12;
CREATE TABLE WAREHOUSE12
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_012
INDEX IN ts_wh_012
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 6425 ENDING AT 7008
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE13;

```

```

CREATE TABLE WAREHOUSE13
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_013
INDEX IN ts_wh_013
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 7009 ENDING AT 7592
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE14;
CREATE TABLE WAREHOUSE14
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_014
INDEX IN ts_wh_014
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 7593 ENDING AT 8176
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE15;
CREATE TABLE WAREHOUSE15
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_015
INDEX IN ts_wh_015
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 8177 ENDING AT 8760
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE16;
CREATE TABLE WAREHOUSE16
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,

```

```

W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_016
INDEX IN ts_wh_016
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 8761 ENDING AT 9344
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE17;
CREATE TABLE WAREHOUSE17

```

```

(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)

```

```

IN ts_wh_017
INDEX IN ts_wh_017
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 9345 ENDING AT 9928
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE18;
CREATE TABLE WAREHOUSE18

```

```

(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)

```

```

IN ts_wh_018
INDEX IN ts_wh_018
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 9929 ENDING AT 10512
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE19;
CREATE TABLE WAREHOUSE19

```

```

(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)

```

```

IN ts_wh_019
INDEX IN ts_wh_019

```

```

ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 10513 ENDING AT 11096
)

```

```

ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE20;
CREATE TABLE WAREHOUSE20

```

```

(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)

```

```

IN ts_wh_020
INDEX IN ts_wh_020
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 11097 ENDING AT 11680
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE21;
CREATE TABLE WAREHOUSE21

```

```

(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)

```

```

IN ts_wh_021
INDEX IN ts_wh_021
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 11681 ENDING AT 12264
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE22;
CREATE TABLE WAREHOUSE22

```

```

(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)

```

```

IN ts_wh_022
INDEX IN ts_wh_022
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 12265 ENDING AT 12848
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE23;

```

```

CREATE TABLE WAREHOUSE23

```

```

(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)

```

```

IN ts_wh_023
INDEX IN ts_wh_023
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 12849 ENDING AT 13432
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE24;
CREATE TABLE WAREHOUSE24

```

```

(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)

```

```

IN ts_wh_024
INDEX IN ts_wh_024
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 13433 ENDING AT 14016
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE25;
CREATE TABLE WAREHOUSE25

```

```

(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)

```

```

IN ts_wh_025
INDEX IN ts_wh_025
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 14017 ENDING AT 14600
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE26;
CREATE TABLE WAREHOUSE26

```

```

(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,

```

```

W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_026
INDEX IN ts_wh_026
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 14601 ENDING AT 15184
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE27;
CREATE TABLE WAREHOUSE27
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_027
INDEX IN ts_wh_027
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 15185 ENDING AT 15768
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE28;
CREATE TABLE WAREHOUSE28
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_028
INDEX IN ts_wh_028
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 15769 ENDING AT 16352
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE29;
CREATE TABLE WAREHOUSE29
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_029
INDEX IN ts_wh_029

```

```

ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 16353 ENDING AT 16936
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE30;
CREATE TABLE WAREHOUSE30
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_030
INDEX IN ts_wh_030
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 16937 ENDING AT 17520
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE31;
CREATE TABLE WAREHOUSE31
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_031
INDEX IN ts_wh_031
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 17521 ENDING AT 18104
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE32;
CREATE TABLE WAREHOUSE32
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_032
INDEX IN ts_wh_032
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 18105 ENDING AT 18688
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE33;

```

```

CREATE TABLE WAREHOUSE33
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_033
INDEX IN ts_wh_033
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 18689 ENDING AT 19272
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE34;
CREATE TABLE WAREHOUSE34
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_034
INDEX IN ts_wh_034
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 19273 ENDING AT 19856
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE35;
CREATE TABLE WAREHOUSE35
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_035
INDEX IN ts_wh_035
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 19857 ENDING AT 20440
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE36;
CREATE TABLE WAREHOUSE36
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,

```

```

W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_036
INDEX IN ts_wh_036
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 20441 ENDING AT 21024
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE37;
CREATE TABLE WAREHOUSE37
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_037
INDEX IN ts_wh_037
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 21025 ENDING AT 21608
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE38;
CREATE TABLE WAREHOUSE38
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_038
INDEX IN ts_wh_038
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 21609 ENDING AT 22192
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE39;
CREATE TABLE WAREHOUSE39
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_039
INDEX IN ts_wh_039

```

```

ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 22193 ENDING AT 22776
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE40;
CREATE TABLE WAREHOUSE40
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_040
INDEX IN ts_wh_040
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 22777 ENDING AT 23360
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE41;
CREATE TABLE WAREHOUSE41
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_041
INDEX IN ts_wh_041
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 23361 ENDING AT 23944
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE42;
CREATE TABLE WAREHOUSE42
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_042
INDEX IN ts_wh_042
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 23945 ENDING AT 24528
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE43;

```

```

CREATE TABLE WAREHOUSE43
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_043
INDEX IN ts_wh_043
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 24529 ENDING AT 25112
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE44;
CREATE TABLE WAREHOUSE44
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_044
INDEX IN ts_wh_044
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 25113 ENDING AT 25696
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE45;
CREATE TABLE WAREHOUSE45
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_045
INDEX IN ts_wh_045
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 25697 ENDING AT 26280
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE46;
CREATE TABLE WAREHOUSE46
(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,

```

```

W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_046
INDEX IN ts_wh_046
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 26281 ENDING AT 26864
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE47;
CREATE TABLE WAREHOUSE47

```

```

(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_047
INDEX IN ts_wh_047
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 26865 ENDING AT 27448
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE48;
CREATE TABLE WAREHOUSE48

```

```

(
W_NAME CHAR(10) NOT NULL,
W_STREET_1 CHAR(20) NOT NULL,
W_STREET_2 CHAR(20) NOT NULL,
W_CITY CHAR(20) NOT NULL,
W_STATE CHAR(2) NOT NULL,
W_ZIP CHAR(9) NOT NULL,
W_TAX REAL NOT NULL,
W_YTD DECIMAL(12,2) NOT NULL,
W_ID INTEGER NOT NULL
)
IN ts_wh_048
INDEX IN ts_wh_048
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 27449 ENDING AT 28032
)
ALLOW OVERFLOW;

```

```
connect reset;
```

DDL/CRVW CUSTOMER.ddl

```

connect to TPCC in share mode;
DROP VIEW CUSTOMER;
CREATE VIEW CUSTOMER

```

```

(C_ID,
C_STATE,
C_ZIP,
C_PHONE,
C_SINCE,
C_CREDIT_LIM,
C_MIDDLE,
C_CREDIT,
C_DISCOUNT,
C_DATA,

```

```

C_LAST,
C_FIRST,
C_STREET_1,
C_STREET_2,
C_CITY,
C_D_ID,
C_W_ID,
C_DELIVERY_CNT,
C_BALANCE,
C_YTD_PAYMENT,
C_PAYMENT_CNT
) AS SELECT * FROM CUSTOMER1 UNION ALL

```

```

SELECT * FROM CUSTOMER2 UNION ALL
SELECT * FROM CUSTOMER3 UNION ALL
SELECT * FROM CUSTOMER4 UNION ALL
SELECT * FROM CUSTOMER5 UNION ALL
SELECT * FROM CUSTOMER6 UNION ALL
SELECT * FROM CUSTOMER7 UNION ALL
SELECT * FROM CUSTOMER8 UNION ALL
SELECT * FROM CUSTOMER9 UNION ALL
SELECT * FROM CUSTOMER10 UNION ALL
SELECT * FROM CUSTOMER11 UNION ALL
SELECT * FROM CUSTOMER12 UNION ALL
SELECT * FROM CUSTOMER13 UNION ALL
SELECT * FROM CUSTOMER14 UNION ALL
SELECT * FROM CUSTOMER15 UNION ALL
SELECT * FROM CUSTOMER16 UNION ALL
SELECT * FROM CUSTOMER17 UNION ALL
SELECT * FROM CUSTOMER18 UNION ALL
SELECT * FROM CUSTOMER19 UNION ALL
SELECT * FROM CUSTOMER20 UNION ALL
SELECT * FROM CUSTOMER21 UNION ALL
SELECT * FROM CUSTOMER22 UNION ALL
SELECT * FROM CUSTOMER23 UNION ALL
SELECT * FROM CUSTOMER24 UNION ALL
SELECT * FROM CUSTOMER25 UNION ALL
SELECT * FROM CUSTOMER26 UNION ALL
SELECT * FROM CUSTOMER27 UNION ALL
SELECT * FROM CUSTOMER28 UNION ALL
SELECT * FROM CUSTOMER29 UNION ALL
SELECT * FROM CUSTOMER30 UNION ALL
SELECT * FROM CUSTOMER31 UNION ALL
SELECT * FROM CUSTOMER32 UNION ALL
SELECT * FROM CUSTOMER33 UNION ALL
SELECT * FROM CUSTOMER34 UNION ALL
SELECT * FROM CUSTOMER35 UNION ALL
SELECT * FROM CUSTOMER36 UNION ALL
SELECT * FROM CUSTOMER37 UNION ALL
SELECT * FROM CUSTOMER38 UNION ALL
SELECT * FROM CUSTOMER39 UNION ALL
SELECT * FROM CUSTOMER40 UNION ALL
SELECT * FROM CUSTOMER41 UNION ALL
SELECT * FROM CUSTOMER42 UNION ALL
SELECT * FROM CUSTOMER43 UNION ALL
SELECT * FROM CUSTOMER44 UNION ALL
SELECT * FROM CUSTOMER45 UNION ALL
SELECT * FROM CUSTOMER46 UNION ALL
SELECT * FROM CUSTOMER47 UNION ALL
SELECT * FROM CUSTOMER48

```

```

WITH ROW MOVEMENT;
COMMIT WORK;
connect reset;

```

DDL/CRVW DISTRICT.ddl

```

connect to TPCC in share mode;
DROP VIEW DISTRICT;
CREATE VIEW DISTRICT
(D_NEXT_O_ID,
D_TAX,

```

```

D_YTD,
D_NAME,
D_STREET_1,
D_STREET_2,
D_CITY,
D_STATE,
D_ZIP,
D_ID,
D_W_ID
) AS SELECT * FROM DISTRICT1 UNION ALL

```

```

SELECT * FROM DISTRICT2 UNION ALL
SELECT * FROM DISTRICT3 UNION ALL
SELECT * FROM DISTRICT4 UNION ALL
SELECT * FROM DISTRICT5 UNION ALL
SELECT * FROM DISTRICT6 UNION ALL
SELECT * FROM DISTRICT7 UNION ALL
SELECT * FROM DISTRICT8 UNION ALL
SELECT * FROM DISTRICT9 UNION ALL
SELECT * FROM DISTRICT10 UNION ALL
SELECT * FROM DISTRICT11 UNION ALL
SELECT * FROM DISTRICT12 UNION ALL
SELECT * FROM DISTRICT13 UNION ALL
SELECT * FROM DISTRICT14 UNION ALL
SELECT * FROM DISTRICT15 UNION ALL
SELECT * FROM DISTRICT16 UNION ALL
SELECT * FROM DISTRICT17 UNION ALL
SELECT * FROM DISTRICT18 UNION ALL
SELECT * FROM DISTRICT19 UNION ALL
SELECT * FROM DISTRICT20 UNION ALL
SELECT * FROM DISTRICT21 UNION ALL
SELECT * FROM DISTRICT22 UNION ALL
SELECT * FROM DISTRICT23 UNION ALL
SELECT * FROM DISTRICT24 UNION ALL
SELECT * FROM DISTRICT25 UNION ALL
SELECT * FROM DISTRICT26 UNION ALL
SELECT * FROM DISTRICT27 UNION ALL
SELECT * FROM DISTRICT28 UNION ALL
SELECT * FROM DISTRICT29 UNION ALL
SELECT * FROM DISTRICT30 UNION ALL
SELECT * FROM DISTRICT31 UNION ALL
SELECT * FROM DISTRICT32 UNION ALL
SELECT * FROM DISTRICT33 UNION ALL
SELECT * FROM DISTRICT34 UNION ALL
SELECT * FROM DISTRICT35 UNION ALL
SELECT * FROM DISTRICT36 UNION ALL
SELECT * FROM DISTRICT37 UNION ALL
SELECT * FROM DISTRICT38 UNION ALL
SELECT * FROM DISTRICT39 UNION ALL
SELECT * FROM DISTRICT40 UNION ALL
SELECT * FROM DISTRICT41 UNION ALL
SELECT * FROM DISTRICT42 UNION ALL
SELECT * FROM DISTRICT43 UNION ALL
SELECT * FROM DISTRICT44 UNION ALL
SELECT * FROM DISTRICT45 UNION ALL
SELECT * FROM DISTRICT46 UNION ALL
SELECT * FROM DISTRICT47 UNION ALL
SELECT * FROM DISTRICT48

```

```

WITH ROW MOVEMENT;
COMMIT WORK;
connect reset;

```

DDL/CRVW HISTORY.ddl

```

connect to TPCC in share mode;
DROP VIEW HISTORY;
CREATE VIEW HISTORY
(H_C_ID,
H_C_D_ID,
H_C_W_ID,

```



```

SELECT * FROM ORDERS33 UNION ALL
SELECT * FROM ORDERS34 UNION ALL
SELECT * FROM ORDERS35 UNION ALL
SELECT * FROM ORDERS36 UNION ALL
SELECT * FROM ORDERS37 UNION ALL
SELECT * FROM ORDERS38 UNION ALL
SELECT * FROM ORDERS39 UNION ALL
SELECT * FROM ORDERS40 UNION ALL
SELECT * FROM ORDERS41 UNION ALL
SELECT * FROM ORDERS42 UNION ALL
SELECT * FROM ORDERS43 UNION ALL
SELECT * FROM ORDERS44 UNION ALL
SELECT * FROM ORDERS45 UNION ALL
SELECT * FROM ORDERS46 UNION ALL
SELECT * FROM ORDERS47 UNION ALL
SELECT * FROM ORDERS48
WITH ROW MOVEMENT;
COMMIT WORK;
connect reset;

```

DDL/CRVW ORDER_LINE.ddl

```

connect to TPCC in share mode;
DROP VIEW ORDER_LINE;
CREATE VIEW ORDER_LINE
(OL_DELIVERY_D,
OL_AMOUNT,
OL_I_ID,
OL_SUPPLY_W_ID,
OL_QUANTITY,
OL_DIST_INFO,
OL_O_ID,
OL_D_ID,
OL_W_ID,
OL_NUMBER
) AS SELECT * FROM ORDER_LINE1 UNION ALL
SELECT * FROM ORDER_LINE2 UNION ALL
SELECT * FROM ORDER_LINE3 UNION ALL
SELECT * FROM ORDER_LINE4 UNION ALL
SELECT * FROM ORDER_LINE5 UNION ALL
SELECT * FROM ORDER_LINE6 UNION ALL
SELECT * FROM ORDER_LINE7 UNION ALL
SELECT * FROM ORDER_LINE8 UNION ALL
SELECT * FROM ORDER_LINE9 UNION ALL
SELECT * FROM ORDER_LINE10 UNION ALL
SELECT * FROM ORDER_LINE11 UNION ALL
SELECT * FROM ORDER_LINE12 UNION ALL
SELECT * FROM ORDER_LINE13 UNION ALL
SELECT * FROM ORDER_LINE14 UNION ALL
SELECT * FROM ORDER_LINE15 UNION ALL
SELECT * FROM ORDER_LINE16 UNION ALL
SELECT * FROM ORDER_LINE17 UNION ALL
SELECT * FROM ORDER_LINE18 UNION ALL
SELECT * FROM ORDER_LINE19 UNION ALL
SELECT * FROM ORDER_LINE20 UNION ALL
SELECT * FROM ORDER_LINE21 UNION ALL
SELECT * FROM ORDER_LINE22 UNION ALL
SELECT * FROM ORDER_LINE23 UNION ALL
SELECT * FROM ORDER_LINE24 UNION ALL
SELECT * FROM ORDER_LINE25 UNION ALL
SELECT * FROM ORDER_LINE26 UNION ALL
SELECT * FROM ORDER_LINE27 UNION ALL
SELECT * FROM ORDER_LINE28 UNION ALL
SELECT * FROM ORDER_LINE29 UNION ALL
SELECT * FROM ORDER_LINE30 UNION ALL
SELECT * FROM ORDER_LINE31 UNION ALL
SELECT * FROM ORDER_LINE32 UNION ALL
SELECT * FROM ORDER_LINE33 UNION ALL
SELECT * FROM ORDER_LINE34 UNION ALL
SELECT * FROM ORDER_LINE35 UNION ALL

```

```

SELECT * FROM ORDER_LINE36 UNION ALL
SELECT * FROM ORDER_LINE37 UNION ALL
SELECT * FROM ORDER_LINE38 UNION ALL
SELECT * FROM ORDER_LINE39 UNION ALL
SELECT * FROM ORDER_LINE40 UNION ALL
SELECT * FROM ORDER_LINE41 UNION ALL
SELECT * FROM ORDER_LINE42 UNION ALL
SELECT * FROM ORDER_LINE43 UNION ALL
SELECT * FROM ORDER_LINE44 UNION ALL
SELECT * FROM ORDER_LINE45 UNION ALL
SELECT * FROM ORDER_LINE46 UNION ALL
SELECT * FROM ORDER_LINE47 UNION ALL
SELECT * FROM ORDER_LINE48
WITH ROW MOVEMENT;
COMMIT WORK;
connect reset;

```

DDL/CRVW STOCK.ddl

```

connect to TPCC in share mode;
DROP VIEW STOCK;
CREATE VIEW STOCK
(S_REMOTE_CNT,
S_QUANTITY,
S_ORDER_CNT,
S_YTD,
S_DATA,
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_I_ID,
S_W_ID
) AS SELECT * FROM STOCK1 UNION ALL
SELECT * FROM STOCK2 UNION ALL
SELECT * FROM STOCK3 UNION ALL
SELECT * FROM STOCK4 UNION ALL
SELECT * FROM STOCK5 UNION ALL
SELECT * FROM STOCK6 UNION ALL
SELECT * FROM STOCK7 UNION ALL
SELECT * FROM STOCK8 UNION ALL
SELECT * FROM STOCK9 UNION ALL
SELECT * FROM STOCK10 UNION ALL
SELECT * FROM STOCK11 UNION ALL
SELECT * FROM STOCK12 UNION ALL
SELECT * FROM STOCK13 UNION ALL
SELECT * FROM STOCK14 UNION ALL
SELECT * FROM STOCK15 UNION ALL
SELECT * FROM STOCK16 UNION ALL
SELECT * FROM STOCK17 UNION ALL
SELECT * FROM STOCK18 UNION ALL
SELECT * FROM STOCK19 UNION ALL
SELECT * FROM STOCK20 UNION ALL
SELECT * FROM STOCK21 UNION ALL
SELECT * FROM STOCK22 UNION ALL
SELECT * FROM STOCK23 UNION ALL
SELECT * FROM STOCK24 UNION ALL
SELECT * FROM STOCK25 UNION ALL
SELECT * FROM STOCK26 UNION ALL
SELECT * FROM STOCK27 UNION ALL
SELECT * FROM STOCK28 UNION ALL
SELECT * FROM STOCK29 UNION ALL
SELECT * FROM STOCK30 UNION ALL
SELECT * FROM STOCK31 UNION ALL

```

```

SELECT * FROM STOCK32 UNION ALL
SELECT * FROM STOCK33 UNION ALL
SELECT * FROM STOCK34 UNION ALL
SELECT * FROM STOCK35 UNION ALL
SELECT * FROM STOCK36 UNION ALL
SELECT * FROM STOCK37 UNION ALL
SELECT * FROM STOCK38 UNION ALL
SELECT * FROM STOCK39 UNION ALL
SELECT * FROM STOCK40 UNION ALL
SELECT * FROM STOCK41 UNION ALL
SELECT * FROM STOCK42 UNION ALL
SELECT * FROM STOCK43 UNION ALL
SELECT * FROM STOCK44 UNION ALL
SELECT * FROM STOCK45 UNION ALL
SELECT * FROM STOCK46 UNION ALL
SELECT * FROM STOCK47 UNION ALL
SELECT * FROM STOCK48
WITH ROW MOVEMENT;
COMMIT WORK;
connect reset;

```

DDL/CRVW WAREHOUSE.ddl

```

connect to TPCC in share mode;
DROP VIEW WAREHOUSE;
CREATE VIEW WAREHOUSE
(W_NAME,
W_STREET_1,
W_STREET_2,
W_CITY,
W_STATE,
W_ZIP,
W_TAX,
W_YTD,
W_ID
) AS SELECT * FROM WAREHOUSE1 UNION ALL
SELECT * FROM WAREHOUSE2 UNION ALL
SELECT * FROM WAREHOUSE3 UNION ALL
SELECT * FROM WAREHOUSE4 UNION ALL
SELECT * FROM WAREHOUSE5 UNION ALL
SELECT * FROM WAREHOUSE6 UNION ALL
SELECT * FROM WAREHOUSE7 UNION ALL
SELECT * FROM WAREHOUSE8 UNION ALL
SELECT * FROM WAREHOUSE9 UNION ALL
SELECT * FROM WAREHOUSE10 UNION ALL
SELECT * FROM WAREHOUSE11 UNION ALL
SELECT * FROM WAREHOUSE12 UNION ALL
SELECT * FROM WAREHOUSE13 UNION ALL
SELECT * FROM WAREHOUSE14 UNION ALL
SELECT * FROM WAREHOUSE15 UNION ALL
SELECT * FROM WAREHOUSE16 UNION ALL
SELECT * FROM WAREHOUSE17 UNION ALL
SELECT * FROM WAREHOUSE18 UNION ALL
SELECT * FROM WAREHOUSE19 UNION ALL
SELECT * FROM WAREHOUSE20 UNION ALL
SELECT * FROM WAREHOUSE21 UNION ALL
SELECT * FROM WAREHOUSE22 UNION ALL
SELECT * FROM WAREHOUSE23 UNION ALL
SELECT * FROM WAREHOUSE24 UNION ALL
SELECT * FROM WAREHOUSE25 UNION ALL
SELECT * FROM WAREHOUSE26 UNION ALL
SELECT * FROM WAREHOUSE27 UNION ALL
SELECT * FROM WAREHOUSE28 UNION ALL
SELECT * FROM WAREHOUSE29 UNION ALL
SELECT * FROM WAREHOUSE30 UNION ALL
SELECT * FROM WAREHOUSE31 UNION ALL
SELECT * FROM WAREHOUSE32 UNION ALL
SELECT * FROM WAREHOUSE33 UNION ALL
SELECT * FROM WAREHOUSE34 UNION ALL
SELECT * FROM WAREHOUSE35 UNION ALL

```



```
alter bufferpool NEW4 size 95000;
alter bufferpool NEW5 size 95000;
alter bufferpool NEW6 size 95000;
alter bufferpool NEW7 size 95000;
alter bufferpool NEW8 size 95000;
```

```
alter bufferpool OLNORDIORD1 size 190000;
alter bufferpool OLNORDIORD2 size 190000;
alter bufferpool OLNORDIORD3 size 190000;
alter bufferpool OLNORDIORD4 size 190000;
alter bufferpool OLNORDIORD5 size 190000;
alter bufferpool OLNORDIORD6 size 190000;
alter bufferpool OLNORDIORD7 size 190000;
alter bufferpool OLNORDIORD8 size 190000;
```

```
alter bufferpool HST1 size 132;
alter bufferpool HST2 size 132;
alter bufferpool HST3 size 132;
alter bufferpool HST4 size 132;
alter bufferpool HST5 size 132;
alter bufferpool HST6 size 132;
alter bufferpool HST7 size 132;
alter bufferpool HST8 size 132;
```

```
alter bufferpool CSTI1 size 31000;
alter bufferpool CSTI2 size 31000;
alter bufferpool CSTI3 size 31000;
alter bufferpool CSTI4 size 31000;
alter bufferpool CSTI5 size 31000;
alter bufferpool CSTI6 size 31000;
alter bufferpool CSTI7 size 31000;
alter bufferpool CSTI8 size 31000;
```

```
connect reset;
terminate;
```

db/create_database.ddl

```
drop database tpcc;
create database tpcc on /db_home collate using identity
catalog tablespace
managed by system using ('/db_home/db1catalog');
```

ts/create_tablespace.ddl

```
connect to tpcc;
create regular tablespace ts_wh_001 pagesize 4K
managed by database
using
(
    device '/dev/raw/raw1' 220
)
extentsize 32
bufferpool IBMDEFAULTBP
prefetchsize 4096;
commit;
drop tablespace ts_wh_002;
create regular tablespace ts_wh_002 pagesize 4K
managed by database
using
(
    device '/dev/raw/raw2' 220
)
extentsize 32
bufferpool IBMDEFAULTBP
prefetchsize 4096;
commit;
```

```
drop tablespace ts_wh_003;
create regular tablespace ts_wh_003 pagesize 4K
managed by database
drop tablespace ts_wh_001;
using
(
    device '/dev/raw/raw3' 220
)
extentsize 32
bufferpool IBMDEFAULTBP
prefetchsize 4096;
commit;
drop tablespace ts_wh_004;
create regular tablespace ts_wh_004 pagesize 4K
managed by database
using
(
    device '/dev/raw/raw4' 220
)
extentsize 32
bufferpool IBMDEFAULTBP
prefetchsize 4096;
commit;
drop tablespace ts_wh_005;
create regular tablespace ts_wh_005 pagesize 4K
managed by database
using
(
    device '/dev/raw/raw5' 220
)
extentsize 32
bufferpool IBMDEFAULTBP
prefetchsize 4096;
commit;
drop tablespace ts_wh_006;
create regular tablespace ts_wh_006 pagesize 4K
managed by database
using
(
    device '/dev/raw/raw6' 220
)
extentsize 32
bufferpool IBMDEFAULTBP
prefetchsize 4096;
commit;
drop tablespace ts_wh_007;
create regular tablespace ts_wh_007 pagesize 4K
managed by database
using
(
    device '/dev/raw/raw7' 220
)
extentsize 32
bufferpool IBMDEFAULTBP
prefetchsize 4096;
commit;
drop tablespace ts_wh_008;
create regular tablespace ts_wh_008 pagesize 4K
managed by database
using
(
    device '/dev/raw/raw8' 220
)
extentsize 32
bufferpool IBMDEFAULTBP
prefetchsize 4096;
commit;
drop tablespace ts_wh_009;
create regular tablespace ts_wh_009 pagesize 4K
managed by database
```

```
using
(
    device '/dev/raw/raw9' 220
)
extentsize 32
bufferpool IBMDEFAULTBP
prefetchsize 4096;
commit;
drop tablespace ts_wh_010;
create regular tablespace ts_wh_010 pagesize 4K
managed by database
using
(
    device '/dev/raw/raw10' 220
)
extentsize 32
bufferpool IBMDEFAULTBP
prefetchsize 4096;
commit;
drop tablespace ts_wh_011;
create regular tablespace ts_wh_011 pagesize 4K
managed by database
using
(
    device '/dev/raw/raw11' 220
)
extentsize 32
bufferpool IBMDEFAULTBP
prefetchsize 4096;
commit;
drop tablespace ts_wh_012;
create regular tablespace ts_wh_012 pagesize 4K
managed by database
using
(
    device '/dev/raw/raw12' 220
)
extentsize 32
bufferpool IBMDEFAULTBP
prefetchsize 4096;
commit;
drop tablespace ts_wh_013;
create regular tablespace ts_wh_013 pagesize 4K
managed by database
using
(
    device '/dev/raw/raw13' 220
)
extentsize 32
bufferpool IBMDEFAULTBP
prefetchsize 4096;
commit;
drop tablespace ts_wh_014;
create regular tablespace ts_wh_014 pagesize 4K
managed by database
using
(
    device '/dev/raw/raw14' 220
)
extentsize 32
bufferpool IBMDEFAULTBP
prefetchsize 4096;
commit;
drop tablespace ts_wh_015;
create regular tablespace ts_wh_015 pagesize 4K
managed by database
using
(
    device '/dev/raw/raw15' 220
)
)
```



```

(
    device '/dev/raw/raw1143' 62000
)
extentsize 256
bufferpool IBMDEFAULTBP
prefetchsize 4096;
commit;
drop tablespace ts_newordB_044;
create regular tablespace ts_newordB_044 pagesize 4K
    managed by database
    using
    (
        device '/dev/raw/raw1144' 62000
    )
    extentsize 256
    bufferpool IBMDEFAULTBP
    prefetchsize 4096;
commit;
drop tablespace ts_newordB_045;
create regular tablespace ts_newordB_045 pagesize 4K
    managed by database
    using
    (
        device '/dev/raw/raw1145' 62000
    )
    extentsize 256
    bufferpool IBMDEFAULTBP
    prefetchsize 4096;
commit;
drop tablespace ts_newordB_046;
create regular tablespace ts_newordB_046 pagesize 4K
    managed by database
    using
    (
        device '/dev/raw/raw1146' 62000
    )
    extentsize 256
    bufferpool IBMDEFAULTBP
    prefetchsize 4096;
commit;
drop tablespace ts_newordB_047;
create regular tablespace ts_newordB_047 pagesize 4K
    managed by database
    using
    (
        device '/dev/raw/raw1147' 62000
    )
    extentsize 256
    bufferpool IBMDEFAULTBP
    prefetchsize 4096;
commit;
drop tablespace ts_newordB_048;
create regular tablespace ts_newordB_048 pagesize 4K
    managed by database
    using
    (
        device '/dev/raw/raw1148' 62000
    )
    extentsize 256
    bufferpool IBMDEFAULTBP
    prefetchsize 4096;
commit;
connect reset;

```

11.2. Data Generation

Makefile.config

```

#####
####
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp. 1996 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use, duplication or
## disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#####

#
# Makefile.config - Linux 64-bit
#
#

# Make Configuration
MAKE=make

# Compiler Configuration.
# CFLAGS_DEBUG may be set to "-g", "-DDEBUGIT" "-g -DDEBUGIT" or left blank
CC=cc
CFLAGS_OS=-DSQLUNIX -DSQLLinux -O2 -fpic -m64
CFLAGS_OUT=-o
CFLAGS_DEBUG=

# Linker Configuration
LD_EXEC=gcc
LD_STORP=gcc
LD_FLAGS_EXEC=
LD_FLAGS_SHLIB=-shared
LD_FLAGS_STORP=$(LD_FLAGS_SHLIB)
LD_FLAGS_LIB=-L$(TPCC_SQLLIB)/lib -ldb2 -m64
LD_FLAGS_OUT=-o

# Library Configuration
AR=ar
AR_FLAGS=-rv
AR_FLAGS_LIB=
AR_FLAGS_OUT=

# OS Commands
ERASE=rm -f
ERASEDIR=$(ERASE) -R
MOVE=mv
COPY=cp

# OS File Extensions & Path Separators
OBJEXT=.o
LIBEXT=.a
SHLIBEXT=.so
BINEXT=
SLASH=/
CMDSEP=;

```

Src.Common/Makefile

```

#####
####
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use, duplication or
## disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#####

#
# Makefile - Makefile for Src.Common
#
#

include $(TPCC_ROOT)/Makefile.config

#
#####
# Preprocessor, Compiler and Linker Flags
#
#####

BND_OPTS = GRANT PUBLIC \
    MESSAGES $*.bnd.msg
PRP_OPTS = BINDFILE \
    OPTLEVEL 1 \
    ISOLATION RR \
    MESSAGES $*.prep.msg \
    LEVEL $(TPCC_VERSION) \
    NOLINEMACRO

INCLUDE = -I$(TPCC_SQLLIB)/include -I$(TPCC_ROOT)/include

CFLAGS = $(CFLAGS_OS) $(CFLAGS_DEBUG) $(INCLUDE) \
    -DSQLA_NOLINES -D$(DB2EDITION) -D$(DB2VERSION) \
    -D$(TPCC_SPTYPE)

UTIL_OBJ_DBG = tpccdbg$(OBJEXT)
UTIL_OBJ_GEN = tpccmisc$(OBJEXT)
UTIL_OBJ_DB2 = tpccctx$(OBJEXT)

#
#####
# User Targets
#
#####

all: $(UTIL_OBJ_DBG) $(UTIL_OBJ_GEN) connect $(UTIL_OBJ_DB2) disconnect

dbgen: $(UTIL_OBJ_GEN)

clean:
    - $(ERASE) *$(OBJEXT) *.bnd *.msg tpccctx.c

#
#####
# Helper Targets
#
#####

connect:
    - db2 connect to $(TPCC_DBNAME)

disconnect:

```

```
- db2 connect reset
- db2 terminate

rebind: connect
db2 bind tpccctx.bnd $(BND_OPTS)
```

```
#
#####
# Build Rules
#
#####
```

```
.SUFFIXES:
.SUFFIXES: $(OBJEXT) .c .sqc
```

```
.sqc.c:
@echo "Prepping $*.sqc"
-db2 prep $*.sqc $(PRP_OPTS)
@echo "Binding $*.bnd"
db2 bind $*.bnd $(BND_OPTS)
```

```
#
#####
# Dependencies
#
#####
```

```
# Source
tpccdbg$(OBJEXT): tpccdbg.c
tpccctx$(OBJEXT): tpccctx.c
tpccmisc$(OBJEXT): tpccmisc.c
```

```
# Headers
tpccdbg.c: $(TPCC_ROOT)/include/db2tpcc.h
```

```
UTIL_OBJ_DB2 = tpccctx$(OBJEXT)
```

```
#
#####
# User Targets
```

Src.Common/tpccmisc.c

```
/*
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*/
```

```
/*
*
* tpccmisc.c - Miscellaneous routines
*
*/
```

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

```
double current_time_ms(void);
double current_time(void);
```

```
/* Current time in SECONDS, precision SECONDS */
```

```
double current_time(void)
{
/* use time() to get seconds */
return(time(NULL));
}
```

```
/* Current time in SECONDS, precision MILLISECONDS */
double current_time_ms(void)
{
/* gettimeofday() returns seconds and microseconds */
/* convert to fractional seconds */
struct timeval t;
gettimeofday(&t,NULL);
return (t.tv_sec + (double)t.tv_usec/(1000*1000));
}
```

dbgen/Makefile

```
#####
### Licensed Materials - Property of IBM
###
### Governed under the terms of the International
### License Agreement for Non-Warranted Sample Code.
###
### (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
### All Rights Reserved.
###
### US Government Users Restricted Rights - Use, duplication or
### disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#####
```

```
# Makefile - Build gendata tool
#
```

```
include $(TPCC_ROOT)/Makefile.config
```

```
#
#####
# Preprocessor, Compiler and Linker Flags
#
#####
```

```
INCLUDE = -I$(TPCC_SQLLIB)/include -I$(TPCC_ROOT)/include
```

```
CFLAGS = $(INCLUDE) $(CFLAGS_OS) -DLINT_ARGS -DSQLA_NOLINES \
-D$(DB2EDITION) -D$(DB2VERSION) $(CFLAGS_DEBUG)
```

```
LDFLAGS = $(LDFLAGS_EXEC) $(LDFLAGS_LIB)
```

```
#
#####
# File Collections
#
#####
```

```
OBJS = tpccrnd$(OBJEXT) \
$(TPCC_ROOT)/Src.Common/tpccmisc$(OBJEXT)
OBJ_EEE = $(TPCC_ROOT)/Src.Common/tpccwh$(OBJEXT)
```

```
EXEC = gendata$(BINEXT)
```

```
#
#####
# End-User Targets
#
#####
```

```
all: $(EXEC)
```

```
clean:
- $(ERASE) *$(OBJEXT) $(EXEC)
```

```
#
#####
# Build Rules
#
#####
```

```
.SUFFIXES:
.SUFFIXES: $(OBJEXT) .c
```

```
# We use $$$(OBJEXT) here so that the UNIX makefiles work with both
# 'traditional' make and GNU make
$(EXEC):
$(LD_EXEC) $(LDFLAGS) $(OBJS) $$$(OBJEXT) $(LDFLAGS_OUT)$$
```

```
#
#####
# Dependencies
#
#####
```

```
# Link Dependencies
gendata$(BINEXT): $(OBJS) gendata$(OBJEXT)
```

```
# Build Dependencies
# Source
gendata$(OBJEXT): gendata.c
```

```
# Headers
gendata.c:$(TPCC_ROOT)/include/tpccrnd.h $(TPCC_ROOT)/include/lval.h
```

dbgen/gendata.c

```
/*
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*/
```

```
/*
* gendata.c - Generate data for TPC-C database
*
*/
```

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sqlutil.h>
/* UNIX named pipe support */
#include <sys/stat.h>
#include <errno.h>
#include <fcntl.h>
#include <time.h>
```

```
#include "platform.h"
#include "db2tpcc.h"
#include "tpccrnd.h"
#include "tpccmisc.h"
#include "lval.h"
```



```

/* PROTOTYPES */
void gen_dist_tbl( void );
void gen_cust_tbl( void );
void gen_hist_tbl( void );
void gen_nu_ord_tbl( void );
void gen_ordr_tbl( void );
void gen_item_tbl( void );
void gen_stock_tbl( void );
void gen_ware_tbl( void );

int i, j;
double timestamp1, timestamp2, elapse;
int rc, rc1, rc2;

int using_range = 0;
int using_npipe = 0;
int using_rctload = 0;
int quiet_mode = 0;
sqlint32 ware_start=-1, ware_end=-1;

char fmtWare[] = "%s%s%s%s%s%04.4f%02f%d\n";
char fmtDist[] = "%d%04.4f%02f%s%s%s%04.4f%02f%d\n";
char fmtItem[] = "%s%02f%s%04.4f%02f%d\n";
char fmtStock[] = "%d%04.4f%02f%s%s%s%04.4f%02f%d\n";
char fmtCust[] = "%d%04.4f%02f%s%s%04.4f%02f%d\n";
char fmtHist[] = "%d%04.4f%02f%s%s%04.4f%02f%d\n";
char fmtOrdr[] = "%d%04.4f%02f%s%s%04.4f%02f%d\n";
char fmtOLine[] = "%s%02f%04.4f%02f%d\n";
char fmtNewOrd[] = "%d%04.4f%02f%d\n";
void InitFormatStrings(char delim);
void ScalingReport(void);

int outtype1 = 0;
int outtype2 = 0;
char *outname1 = NULL;
char *outname2 = NULL;

/*-----*/
/* main */
/*-----*/
int main (int argc, char *argv[])
{
    int option = -1;
    char *delim = NULL;

    /*-----*/
    /* Compute Warehouse Ranges */
    /*-----*/
    ware_start = 1;
    ware_end = WAREHOUSES;

    /*-----*/
    /* Process Command Line Arguments */
    /*-----*/

    /* Valid Command Line Options
    * -----
    * Table Option: -t <table> (-t 3 for warehouse)
    * Output Column Delimiter: -d <char> (-d ' ', -d '|', etc)
    * Output to File: -f[n] <file> (-f customer.dat)
    * Output to Pipe: -p[n] <pipename> (-p tpccpipe.000)
    * Warehouse Range: -r <start> <end> (-r 1 100)
    * Scaling Report: -s
    * Quiet Mode: -q
    *
    * The -f[n] and/or -p[n] options are required.
    * The -t, -d, -r, -s and -q options are optional.

```

```

*
* If -d is omitted, the vertical bar (pipe) symbol (|) will be used.
* If -r is omitted, the range [1..WAREHOUSES] will be used.
*
* Due to the TPC-C spec requiring that orders and orderline be
* generated at the same time, there is an extension to the -f and -p
* options to specify one of the two output streams for each argument.
*
* -f1 orders.dat -f2 orderline.dat will output to two files
* -f1 orders.dat -p2 tpccpipe.000 will output to a file and a pipe
*
* -f1/-p1 specifies the destination for the orders table
* -f2/-p2 specifies the destination for the orderline table
*
*/

/* Read Arguments */
for (i=1; i<argc; i++)
{
    if (strcmp(argv[i], "-t") == 0) {
        option = atoi(argv[i+1]);
        i++;
    } else if (strcmp(argv[i], "-r") == 0) {
        ware_start = atoi(argv[i+1]);
        ware_end = atoi(argv[i+2]);
        i += 2;
    } else if (strcmp(argv[i], "-d") == 0) {
        delim = argv[i+1];
        i++;
    } else if ((strcmp(argv[i], "-f") == 0) ||
                (strcmp(argv[i], "-p1") == 0)) {
        outtype1 = IOH_FILE;
        outname1 = argv[i+1];
        i++;
    } else if (strcmp(argv[i], "-f2") == 0) {
        outtype2 = IOH_FILE;
        outname2 = argv[i+1];
        i++;
    } else if ((strcmp(argv[i], "-p") == 0) ||
                (strcmp(argv[i], "-p1") == 0)) {
        outtype1 = IOH_PIPE;
        outname1 = argv[i+1];
        i++;
    } else if (strcmp(argv[i], "-p2") == 0) {
        outtype2 = IOH_PIPE;
        outname2 = argv[i+1];
        i++;
    } else if (strcmp(argv[i], "-s") == 0) {
        ScalingReport();
        exit(0);
    } else if (strcmp(argv[i], "-q") == 0) {
        quiet_mode = 1;
    } else {
        fprintf(stderr, "gendata: Don't understand argument: %s\n", argv[i]);
        exit(-1);
    }
}

/*-----*/
/* Validate Command Line Arguments */
/*-----*/

/* Validate Table Argument */
if (option < 3 || option > 11 || option == 10)
{
    fprintf(stderr, "gendata: Invalid table selected: %d\n", option);
    exit(-1);
}

/* Validate Delimiter Argument */

```

```

if (delim == NULL) {
    // default delimiter is used for IMPORT & LOAD, no changes necessary
    using_rctload = 0;
} else if (strlen(delim) == 1 && !isalnum(delim[0]) &&
           delim[0] != '.' && delim[0] != '%')
{
    // user-supplied delimiter used for rctload
    InitFormatStrings(delim[0]);
    using_rctload = 1;
} else {
    fprintf(stderr, "gendata: Invalid delimiter specified: %s\n", delim);
    exit(-1);
}

/* Validate File/Pipe Arguments */
if (option != 9 && outtype1 > 0 && outtype2 > 0)
{
    fprintf(stderr, "gendata: Specifying two output file/pipes allowed only when
    generating\norders/orderline.\n");
    exit(-1);
}
if (option == 9 && ((outtype1 == 0) || (outtype2 == 0)))
{
    fprintf(stderr, "gendata: Must specify two output file/pipes when generating
    orders/orderline.\n");
    exit(-1);
}
if (outtype1 == 0 || outname1 == NULL || strcmp(outname1, "") == 0)
{
    fprintf(stderr, "gendata: Invalid 1st output file/pipe specified.\n");
    exit(-1);
}
if (option == 9 && (outtype2 == 0 || outname2 == NULL || strcmp(outname2, "") == 0))
{
    fprintf(stderr, "gendata: Invalid 2nd output file/pipe specified.\n");
    exit(-1);
}

/* Ensure O/OL flat files are opened in append mode. This is required */
/* because we generate O/OL concurrently. See comments in genload.pl */
/* for further details on why this is necessary. */
if (option == 9)
{
    if (outtype1 == IOH_FILE) outtype1 = IOH_FILE_APPEND;
    if (outtype2 == IOH_FILE) outtype2 = IOH_FILE_APPEND;
}

/* Validate Range Arguments */
if (ware_start <= 0 || ware_start > WAREHOUSES) {
    fprintf(stderr, "gendata: Invalid range starting value: %d\n", ware_start);
    exit(-1);
}
if (ware_end <= 0 || ware_end > WAREHOUSES || ware_end < ware_start) {
    fprintf(stderr, "gendata: Invalid range ending value: %d\n", ware_end);
    exit(-1);
}

initialize_random();

/*-----*/
/* Generate Data */
/*-----*/

switch (option) {
    case 3: /* WAREHOUSE */
        gen_ware_tbl();
        break;
    case 4: /* DISTRICT */
        gen_dist_tbl();
        break;
    case 5: /* ITEM */
        gen_item_tbl();

```

```

break;
case 6: /* STOCK */
gen_stock_tbl();
break;
case 7: /* CUSTOMER */
gen_cust_tbl();
break;
case 8: /* HISTORY */
gen_hist_tbl();
break;
case 9: /* ORDERS + ORDER_LINE */
gen_ordr_tbl();
break;
case 11: /* NEW_ORDER */
gen_nu_ord_tbl();
break;
case 2:
case 10:
default:
fprintf(stderr, "Error: invalid option = %d\n", (option));
break;
}
return 0;
}

/*-----*/
/* generate item table */
/*-----*/

void gen_item_tbl( void )
{
    sqlint32 item_num = 0 ;
    sqlint32 item_im_id ;
    char item_name[25] ;
    double item_price ;
    char item_data[51] ;

    IOH_NUM numBytes;
    ioHandle hnd;
    char Buffer[1024];

    timestamp1 = current_time();

    rc = GenericOpen(&hnd, outtype1, outname1);
    if (rc != 0) { goto item_done; }

    for(item_num = 1; item_num <= ITEMS; item_num++)
    {
        /* create image id field */
        item_im_id = rand_integer( 1, 10000 ) ;
        /* create name field */
        create_random_a_string( item_name, 14, 24);
        /* create price field */
        item_price = rand_decimal( 100, 10000, 2 ) ;
        /* create ORIGINAL field */
        create_a_string_with_original( item_data, 26, 50, 10 ) ;

        numBytes = sprintf(Buffer, fmtItem,
            item_name,
            item_price,
            item_data,
            item_im_id,
            item_num);

        rc = GenericWrite(&hnd, Buffer, numBytes);
        if (rc != 0) { goto item_done; }

    } /* end for... */

    rc = GenericClose(&hnd);

```

```

item_done:

    timestamp2 = current_time();
    elapse = timestamp2 - timestamp1;
    if (rc == 0) {
        if (!quiet_mode) {
            fprintf(stdout, "\nITEM table generated in %8.2f seconds.\n\n", elapse);
            fflush(stdout);
        }
    } else {
        fprintf(stderr, "\nITEM table FAILED at (I %d) after %8.2f
seconds.\n\n", item_num, elapse);
        fflush(stderr);
    }
}

/*-----*/
/* generate stock table */
/*-----*/

void gen_stock_tbl( void )
{
    sqlint32 ware_num = 0 ;
    sqlint32 stock_num = 0 ;
    sqlint32 stock_quant;
    sqlint32 s_ytd;
    sqlint32 s_order_cnt, s_remote_cnt;
    char stock_dist_01[25] ;
    char stock_dist_02[25] ;
    char stock_dist_03[25] ;
    char stock_dist_04[25] ;
    char stock_dist_05[25] ;
    char stock_dist_06[25] ;
    char stock_dist_07[25] ;
    char stock_dist_08[25] ;
    char stock_dist_09[25] ;
    char stock_dist_10[25] ;
    char stock_data[51] ;

    IOH_NUM numBytes;
    ioHandle hnd;
    char Buffer[1024];

    timestamp1 = current_time();

    rc = GenericOpen(&hnd, outtype1, outname1);
    if (rc != 0) { goto stock_done; }

    for (stock_num = 1; stock_num <= STOCK_PER_WAREHOUSE; stock_num++)
    {
        if (!quiet_mode && (stock_num%500 == 0))
        {
            fprintf(stdout, "STOCK for Item #%d\n", stock_num);
            fflush(stdout);
        }
        for (ware_num = ware_start; ware_num <= ware_end; ware_num++)
        {
            stock_quant = rand_integer( 10, 100 ) ;
            create_random_a_string( stock_dist_01, 24, 24);
            create_random_a_string( stock_dist_02, 24, 24);
            create_random_a_string( stock_dist_03, 24, 24);
            create_random_a_string( stock_dist_04, 24, 24);
            create_random_a_string( stock_dist_05, 24, 24);
            create_random_a_string( stock_dist_06, 24, 24);
            create_random_a_string( stock_dist_07, 24, 24);
            create_random_a_string( stock_dist_08, 24, 24);
            create_random_a_string( stock_dist_09, 24, 24);
            create_random_a_string( stock_dist_10, 24, 24);

            /* create ORIGINAL field */

```

```

create_a_string_with_original( stock_data, 26, 50, 10 );
s_ytd = s_order_cnt = s_remote_cnt = 0;

    numBytes = sprintf(Buffer, fmtStock,
        s_remote_cnt,
        stock_quant,
        s_order_cnt,
        s_ytd,
        stock_data,
        stock_dist_01,
        stock_dist_02,
        stock_dist_03,
        stock_dist_04,
        stock_dist_05,
        stock_dist_06,
        stock_dist_07,
        stock_dist_08,
        stock_dist_09,
        stock_dist_10,
        stock_num,
        ware_num);

    rc = GenericWrite(&hnd, Buffer, numBytes);
    if (rc != 0) { goto stock_done; }

} /* end for... */
} /* end for... */

rc = GenericClose(&hnd);

stock_done:

    timestamp2 = current_time();
    elapse = timestamp2 - timestamp1;
    if (rc == 0) {
        if (!quiet_mode) {
            fprintf(stdout, "\nSTOCK table generated in %8.2f seconds.\n\n", elapse);
            fflush(stdout);
        }
    } else {
        fprintf(stderr, "\nSTOCK table FAILED at (S %d W %d) after %8.2f
seconds.\n\n", stock_num, ware_num, elapse);
        fflush(stderr);
    }
}

/*-----*/
/* generate warehouse table */
/*-----*/

void gen_ware_tbl( void )
{
    sqlint32 ware_num = 0 ;
    char ware_name[11] ;
    char ware_street_1[21] ;
    char ware_street_2[21] ;
    char ware_city[21] ;
    char ware_state[3] ;
    char ware_zip[10] ;
    double ware_tax ;
    double ware_YTD ;

    IOH_NUM numBytes;
    ioHandle hnd;
    char Buffer[1024];

    timestamp1 = current_time();

    rc = GenericOpen(&hnd, outtype1, outname1);
    if (rc != 0) { goto ware_done; }

```

```

for (ware_num = ware_start; ware_num <= ware_end; ware_num++)
{
    if (!quiet_mode && ((ware_num % 500) == 0)) {
        fprintf(stdout, "Warehouse #%d\n", ware_num);
        fflush(stdout);
    }

    create_random_a_string( ware_name, 6,10); /* create name */
    create_random_a_string( ware_street_1, 10,20); /* create street 1 */
    create_random_a_string( ware_street_2, 10,20); /* create street 2 */
    create_random_a_string( ware_city, 10,20); /* create city */
    create_random_a_string( ware_state, 2,2); /* create state */
    create_random_n_string( ware_zip, 4,4); /* create zip */
    strcat(ware_zip, "11111");

    ware_tax = rand_decimal(0, 2000,4);
    ware_YTD = 300000.00;

    numBytes = sprintf(Buffer, fmtWare,
        ware_name,
        ware_street_1,
        ware_street_2,
        ware_city,
        ware_state,
        ware_zip,
        ware_tax,
        ware_YTD,
        ware_num);

    rc = GenericWrite(&hnd, Buffer, numBytes);
    if (rc != 0) { goto ware_done; }

} /* end for */

rc = GenericClose(&hnd);

ware_done:

timestamp2 = current_time();
elapsed = timestamp2 - timestamp1;
if (rc == 0) {
    if (!quiet_mode) {
        fprintf(stdout, "\nWAREHOUSE table generated in %8.2f seconds.\n\n", elapsed);
        fflush(stdout);
    }
} else {
    fprintf(stderr, "\nWAREHOUSE table FAILED at (W %d) after %8.2f
seconds.\n\n", ware_num, elapsed);
    fflush(stderr);
}

}

/*-----*/
/* generate dist table */
/*-----*/
void gen_dist_tbl( void )
{
    sqlint32 ware_num = 0 ;
    sqlint32 dist_num = 0 ;
    char dist_name[11];
    char dist_street_1[21];
    char dist_street_2[21];
    char dist_city[21];
    char dist_state[3];
    char dist_zip[10];
    double dist_tax;
    sqlint32 next_o_id;
    double dist_YTD;

    IOH_NUM numBytes;

```

```

ioHandle hnd;
char Buffer[1024];

next_o_id = CUSTOMERS_PER_DISTRICT + 1;
timestamp1 = current_time();

rc = GenericOpen(&hnd, outtype1, outname1);
if (rc != 0) { goto dist_done; }

for (ware_num = ware_start; ware_num <= ware_end; ware_num++)
{
    if (!quiet_mode) {
        fprintf(stdout, "DISTRICT for Warehouse #%d\n", ware_num);
        fflush(stdout);
    }
    for (dist_num = 1; dist_num <= DISTRICTS_PER_WAREHOUSE; dist_num++)
    {
        create_random_a_string( dist_name, 6,10); /* create name */
        create_random_a_string( dist_street_1, 10,20); /* create street 1 */
        create_random_a_string( dist_street_2, 10,20); /* create street 2 */
        create_random_a_string( dist_city, 10,20); /* create city */
        create_random_a_string( dist_state, 2,2); /* create state */
        create_random_n_string( dist_zip, 4,4); /* create zip */
        strcat(dist_zip, "11111");
        dist_tax = rand_decimal(0, 2000,4);
        dist_YTD = 300000.00;

        numBytes = sprintf(Buffer, fmtDist,
            next_o_id,
            dist_tax,
            dist_YTD,
            dist_name,
            dist_street_1,
            dist_street_2,
            dist_city,
            dist_state,
            dist_zip,
            dist_num,
            ware_num);

        rc = GenericWrite(&hnd, Buffer, numBytes);
        if (rc != 0) { goto dist_done; }

    } /* end for... */
} /* end for... */

rc = GenericClose(&hnd);

dist_done:

timestamp2 = current_time();
elapsed = timestamp2 - timestamp1;
if (rc == 0) {
    if (!quiet_mode) {
        fprintf(stdout, "\nDISTRICT table generated in %8.2f seconds.\n\n", elapsed);
        fflush(stdout);
    }
} else {
    fprintf(stderr, "\nDISTRICT table FAILED at (W %d D %d) after %8.2f
seconds.\n\n", ware_num, dist_num, elapsed);
    fflush(stderr);
}

}

/*-----*/
/* generate customer table */
/*-----*/
void gen_cust_tbl( void )
{
    sqlint32 ware_num = 0 ;

```

```

    sqlint32 dist_num = 0 ;
    sqlint32 cust_num = 0 ;
    char cust_last[17];
    char cust_middle[3];
    char cust_first[17];
    char cust_street_1[21];
    char cust_street_2[21];
    char cust_city[21];
    char cust_state[3];
    char cust_zip[10];
    char cust_phone[17];
    char cust_credit[3];
    char cust_data[501];
    char cust_since[27];
    double cust_discount;
    double cust_balance;
    double cust_YTD_payment;
    double cust_credit_lim;

    IOH_NUM numBytes;
    ioHandle hnd;
    char Buffer[1024];
    int len, pos;

    timestamp1 = current_time();

    rc = GenericOpen(&hnd, outtype1, outname1);
    if (rc != 0) { goto cust_done; }

    strcpy(cust_middle, "OE");

    createTimestampString(cust_since);

    for (cust_num = 1; cust_num <= CUSTOMERS_PER_DISTRICT; cust_num++)
    {
        if (!quiet_mode) {
            fprintf(stdout, "CUSTOMER #%d\n", cust_num);
            fflush(stdout);
        }

        for (ware_num = ware_start; ware_num <= ware_end; ware_num++)
        {
            for (dist_num = 1; dist_num <= DISTRICTS_PER_WAREHOUSE; dist_num++)
            {
                if (cust_num <= 1000) /* create last name */
                    create_random_last_name( cust_last, cust_num);
                else /* create last name */
                    create_random_last_name( cust_last, 0);
                create_random_a_string( cust_first, 8,16); /* create first name */
                create_random_a_string( cust_street_1, 10,20); /* create street 1 */
                create_random_a_string( cust_street_2, 10,20); /* create street 2 */
                create_random_a_string( cust_city, 10,20); /* create city */
                create_random_a_string( cust_state, 2,2); /* create state */
                create_random_n_string( cust_zip, 4,4); /* create zip */
                strcat(cust_zip, "11111");

                /* create phone number */
                create_random_n_string( cust_phone, 16,16);
                if ( rand_integer( 1, 100 ) <= 10 )
                    strcpy( cust_credit, "BC" );
                else
                    strcpy( cust_credit, "GC" );

                /* create discount rate */
                cust_discount = rand_decimal(0,5000,4);

                /* create customer data */
                create_random_a_string(cust_data, 300, 500);

                /* pad customer data (only for non-rctload) */

```

```

if (using_rctload == 0) {
    for (pos=strlen(cust_data); pos<500; pos++)
        cust_data[pos] = ' ';
    cust_data[500] = '\0';
}

cust_credit_lim = 50000.00;
cust_balance = -10.00;
cust_YTD_payment = 10.00;

if (cust_num == 1 && dist_num == 1 && ware_num == 1)
{
    sprintf(cust_first,"C_LAST_LOAD=%d",C_C_LAST_LOAD);
}

numBytes = sprintf(Buffer, fmtCust,
    cust_num,
    cust_state,
    cust_zip,
    cust_phone,
    cust_since,
    cust_credit_lim,
    cust_middle,
    cust_credit,
    cust_discount,
    cust_data,
    cust_last,
    cust_first,
    cust_street_1,
    cust_street_2,
    cust_city,
    dist_num,
    ware_num,
    0,
    cust_balance,
    cust_YTD_payment,
    1);

rc = GenericWrite(&hnd, Buffer, numBytes);
if (rc != 0) { goto cust_done; }

} /* end for district... */
} /* end for warehouse... */
} /* end for customer... */

rc = GenericClose(&hnd);

cust_done:

timestamp2 = current_time();
elapsed = timestamp2 - timestamp1;
if (rc == 0) {
    if (!quiet_mode) {
        fprintf(stdout,"nCUSTOMER table generated in %8.2f seconds.\n\n",elapsed);
        fflush(stdout);
    }
} else {
    fprintf(stderr,"nCUSTOMER table FAILED at (W %d D %d C %d) after %8.2f
seconds.\n\n",ware_num, dist_num, cust_num, elapsed);
    fflush(stderr);
}
}

/*-----*/
/* generate hist table */
/*-----*/
void gen_hist_tbl( void )
{
    sqlint32 ware_num = 0 ;
    sqlint32 dist_num = 0 ;

```

```

sqlint32 cust_num = 0 ;
char hist_data[25] ;
char h_date[27] ;

IOH_NUM numBytes;
ioHandle hnd;
char Buffer[1024];

timestamp1 = current_time();

rc = GenericOpen(&hnd, outtype1, outname1);
if (rc != 0) { goto hist_done; }

createTimestampString(h_date);

for (ware_num = ware_start; ware_num <= ware_end; ware_num++)
{
    if (!quiet_mode) {
        fprintf(stdout, "HISTORY for Warehouse #%d:\n", ware_num);
        fflush(stdout);
    }
    for (dist_num = 1; dist_num <= DISTRICTS_PER_WAREHOUSE; dist_num++)
    {
        for (cust_num = 1; cust_num <= CUSTOMERS_PER_DISTRICT; cust_num++)
        {
            /* create history data */
            create_random_a_string( hist_data, 12,24 ) ;

            numBytes = sprintf(Buffer, fmtHist,
                cust_num,
                dist_num,
                ware_num,
                dist_num,
                ware_num,
                h_date,
                10.00,
                hist_data);

            rc = GenericWrite(&hnd, Buffer, numBytes);
            if (rc != 0) { goto hist_done; }

        } /* end for customer... */
    } /* end for district... */
} /* end for warehouse... */

rc = GenericClose(&hnd);

hist_done:

timestamp2 = current_time();
elapsed = timestamp2 - timestamp1;
if (rc == 0) {
    if (!quiet_mode) {
        fprintf(stdout,"nHISTORY table generated in %8.2f seconds.\n\n",elapsed);
        fflush(stdout);
    }
} else {
    fprintf(stderr,"nHISTORY table FAILED at (W %d D %d C %d) after %8.2f
seconds.\n\n",ware_num, dist_num, cust_num, elapsed);
    fflush(stderr);
}
}

/*-----*/
/* generate nu_ord table */
/*-----*/
void gen_nu_ord_tbl( void )
{
    sqlint32 ware_num = 0 ;
    sqlint32 dist_num = 0 ;

```

```

sqlint32 nu_ord_id = 0 ;
int nu_ord_hi ;

IOH_NUM numBytes;
ioHandle hnd;
char Buffer[1024];

/* compute maximum and minimum
order numbers for this
district */
nu_ord_hi = CUSTOMERS_PER_DISTRICT - NU_ORDERS_PER_DISTRICT + 1;
if (nu_ord_hi < 0) {
    nu_ord_hi = CUSTOMERS_PER_DISTRICT - (CUSTOMERS_PER_DISTRICT / 3) + 1
    fprintf(stderr,"n**** WARNING **** NU_ORDERS_PER_DISTRICT is >
CUSTOMERS_PER_DISTRICT\n");
    fprintf(stderr,"          Check the values in file lval.h\n");
    fprintf(stderr,"          Loading New-Order with 1/3 of
CUSTOMERS_PER_DISTRICT\n");
}

timestamp1 = current_time();

rc = GenericOpen(&hnd, outtype1, outname1);
if (rc != 0) { goto neword_done; }

/* We generate in O/W/D order for non-RCT tables. With the
* data clustered on O_ID, this gives us good bufferpool
* characteristics. We also create a btree index in W/D/O
* order, to satisfy MIN(O_ID) queries.
*
* For RCT tables *with* RCT Jump Cache, we *should* generate
* the data in W/D/O order (to match the table definition.)
* We don't since it would push schema decisions into flat file
* generation (and I don't want to do that.) It's just as easy
* to sort the flat files afterwards.
*/

for (nu_ord_id = nu_ord_hi;
    nu_ord_id <= CUSTOMERS_PER_DISTRICT;
    nu_ord_id++)
{
    if (!quiet_mode) {
        fprintf(stdout, "NEW_ORDER for Customer #%d:\n", nu_ord_id);
        fflush(stdout);
    }
    for (ware_num = ware_start; ware_num <= ware_end; ware_num++)
    {
        for (dist_num = 1; dist_num <= DISTRICTS_PER_WAREHOUSE; dist_num++)
        {
            numBytes = sprintf(Buffer, fmtNewOrd,
                nu_ord_id,
                dist_num,
                ware_num);

            rc = GenericWrite(&hnd, Buffer, numBytes);
            if (rc != 0) { goto neword_done; }

        } /* end for... */
    } /* end for... */
} /* end for... */

rc = GenericClose(&hnd);

neword_done:

timestamp2 = current_time();
elapsed = timestamp2 - timestamp1;
if (rc == 0) {
    if (!quiet_mode) {
        fprintf(stdout,"nNEW_ORDER table generated in %8.2f seconds.\n\n",elapsed);
    }
}

```

```

        fflush(stdout);
    }
} else {
    fprintf(stderr, "\nNEW_ORDER table FAILED at (W %d D %d O %d) after %8.2f
seconds.\n\n", ware_num, dist_num, nu_ord_id, elapse);
    fflush(stderr);
}
}
}

/*-----*/
/* generate order and order_line tables */
/*-----*/
void gen_ordr_tbl( void )
{
    sqlint32 ware_num = 0 ;
    sqlint32 dist_num = 0 ;
    sqlint32 cust_num = 0 ;
    sqlint32 ord_num = 0 ;
    sqlint32 ordr_carrier_id;
    sqlint32 ordr_ol_cnt;
    sqlint32 oline_ol_num;
    sqlint32 oline_item_num;

    double oline_amount;
    char oline_dist_info[25];

    IOH_NUM numBytes;
    ioHandle hnd1, hnd2;
    char Buffer[1024];

    char currtmstmp[27];
    char nulltmstmp[27] = "0001-01-01 00:00:00";

    oline_dist_info[24] = '\0';

    timestamp1 = current_time();

    rc1 = GenericOpen(&hnd1, outtype1, outname1);
    if (rc1 != 0) { goto ool_done; }
    rc2 = GenericOpen(&hnd2, outtype2, outname2);
    if (rc2 != 0) { goto ool_done; }

    createTimestampString(currtmstmp);

    for (ware_num = ware_start; ware_num <= ware_end; ware_num++)
    {
        if (!quiet_mode) {
            fprintf(stdout, "ORDERS & ORDER_LINE for Warehouse #%d\n", ware_num);
            fflush(stdout);
        }
        for (dist_num = 1; dist_num <= DISTRICTS_PER_WAREHOUSE; dist_num++)
        {
            if (!quiet_mode) {
                fprintf(stdout, "District #%d", dist_num);
                fflush(stdout);
            }
        }

        seed_1_3000();

        for (ord_num = 1; ord_num <= CUSTOMERS_PER_DISTRICT; ord_num++)
        {
            if (ord_num < 2101)
                ordr_carrier_id = rand_integer( 1, 10 );
            else
                ordr_carrier_id = 0;

            cust_num = random_1_3000();
            ordr_ol_cnt = rand_integer(MIN_OL_PER_ORDER, MAX_OL_PER_ORDER);

            numBytes = sprintf(Buffer, fmtOrdr,

```

```

                cust_num,
                currtmstmp,
                ordr_carrier_id,
                ordr_ol_cnt,
                1,
                ord_num,
                ware_num,
                dist_num);

            rc1 = GenericWrite(&hnd1, Buffer, numBytes);
            if (rc1 != 0) { goto ool_done; }

            for ( oline_ol_num = 1; oline_ol_num <= ordr_ol_cnt; oline_ol_num++ )
            {
                oline_item_num = rand_integer(1, ITEMS);
                create_random_a_string( oline_dist_info, 24, 24);

                numBytes = sprintf(Buffer, fmtOLine,
                    ((ord_num < 2101) ? currtmstmp : nulltmstmp),
                    ((ord_num < 2101) ? 0.00 : rand_decimal(1,999999,2)),
                    oline_item_num,
                    ware_num,
                    5,
                    oline_dist_info,
                    ord_num,
                    dist_num,
                    ware_num,
                    oline_ol_num);

                rc2 = GenericWrite(&hnd2, Buffer, numBytes);
                if (rc2 != 0) { goto ool_done; }

                } /* for order_line */
            } /* for order */
        } /* for dist */
    } /* for ware */

    rc1 = GenericClose(&hnd2);
    rc2 = GenericClose(&hnd1);

ool_done:

    timestamp2 = current_time();
    elapse = timestamp2 - timestamp1;
    if (rc1 == 0 && rc2 == 0) {
        if (!quiet_mode) {
            fprintf(stdout, "\nORDERS & ORDER_LINE tables generated in %8.2f
seconds.\n\n", elapse);
            fflush(stdout);
        }
    } else {
        fprintf(stderr, "\nORDERS & ORDER_LINE tables FAILED at (W %d D %d O %d OL %d)
after %8.2f seconds.\n\n", ware_num, dist_num, ord_num, oline_ol_num, elapse);
        fflush(stderr);
    }
}

// This routine will initialize the printf format strings and replace the
// delimiter with the one provided. The pipe symbol is the default.
void InitFormatStrings(char delim)
{
    char *p;

    // Check if Using Default Delimiter
    if (delim == '|') return;

    // Replace Delimiters
    while (p = strchr(fmtWare, '|')) { *p = delim; }
    while (p = strchr(fmtDist, '|')) { *p = delim; }
    while (p = strchr(fmtItem, '|')) { *p = delim; }

```

```

    while (p = strchr(fmtStock, '|')) { *p = delim; }
    while (p = strchr(fmtCust, '|')) { *p = delim; }
    while (p = strchr(fmtHist, '|')) { *p = delim; }
    while (p = strchr(fmtOrdr, '|')) { *p = delim; }
    while (p = strchr(fmtOLine, '|')) { *p = delim; }
    while (p = strchr(fmtNewOrd, '|')) { *p = delim; }
}

void ScalingReport(void)
{
    /* Print Scaling Values */
    fprintf(stdout, "Scaling Values in Use\n");
    fprintf(stdout, "-----\n");
    fprintf(stdout, "Warehouses: %d\n", WAREHOUSES);
    fprintf(stdout, "Districts/Warehouse: %d\n", DISTRICTS_PER_WAREHOUSE);
    fprintf(stdout, "Customers/District: %d\n", CUSTOMERS_PER_DISTRICT);
    fprintf(stdout, "Items: %d\n", ITEMS);
    fprintf(stdout, "Stock/Warehouse: %d\n", STOCK_PER_WAREHOUSE);
    fprintf(stdout, "Min Order Lines/Order: %d\n", MIN_OL_PER_ORDER);
    fprintf(stdout, "Max Order Lines/Order: %d\n", MAX_OL_PER_ORDER);
    fprintf(stdout, "New Orders/District: %d\n", NU_ORDERS_PER_DISTRICT);
    fprintf(stdout, "-----\n");
}

dbgen/tpccrnd.c

/*-----*/
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
/*-----*/

/*
 * tpccrnd.c - Random generation functions for TPC-C
 */

#include <stdio.h>
#include <string.h>
#include <math.h>
#include "db2tpcc.h"
#include "tpccmisc.h"
#include "lval.h"

static char tbl_cust[CUSTOMERS_PER_DISTRICT];

static char alnum[] =
    "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

static char *last_name_parts[] =
{
    "BAR",
    "OUGHT",
    "ABLE",
    "PRI",
    "PRES",
    "ESE",
    "ANTI",
    "CALLY",
    "ATION",
    "ING"
};
}

```

```

/*
*****
* rand_integer
*
* create a uniform random numeric value of type integer, of random
* value between lo and hi. Number is NOT placed in BUFFER, and IS
* simply RETURNED.
*
* Routine RETURNS the VALUE.
*
* parameters
* -----
* lo end of acceptable value range
* hi end of acceptable value range
*
* output
* -----
* random integer value RETURNED
*****
*/

int rand_integer ( int val_lo, int val_hi )
{
    return((random()%(val_hi-val_lo+1))+val_lo);
}

/*
*****
* rand_decimal
*
* create a uniform random numeric value of type double, of random
* value between lo and hi with val_dec fractional digits.
* Number is NOT placed in BUFFER, and IS simply RETURNED.
*
* Routine RETURNS the VALUE.
*
* parameters
* -----
* lo end of acceptable value range
* hi end of acceptable value range
* number of fractional digits
*
* output
* -----
* random double value RETURNED
*****
*/

double rand_decimal ( int val_lo, int val_hi, int val_dec )
{
    return(rand_integer(val_lo,val_hi)/pow(10.0,(double)val_dec));
}

/*
*****
* seed_1_3000
*
*****
*/

void seed_1_3000( void )
{
    int i;

    for (i = 0; i < CUSTOMERS_PER_DISTRICT; i++) {
        tbl_cust[i] = 0;
    }
}

```

```

}
/*
*****
* random_1_3000
*
*****
*/

int random_1_3000( void )
{
    static int i;
    static int x;

    x = rand_integer(0, CUSTOMERS_PER_DISTRICT - 1);

    for (i = 0; i < CUSTOMERS_PER_DISTRICT; i++)
    {
        if (tbl_cust[x] == 0)
        {
            tbl_cust[x] = 1;
            return(x+1);
        } else {
            x++;
        }
        if (x == CUSTOMERS_PER_DISTRICT)
            x=0;
    }

    printf("\nfatal error in random_1_3000\n");
    abort();
}

/*
*****
* initialize_random
*****
*/

void initialize_random(void)
{
    int t = current_time();

    srand(t);
    srandom(t);
}

/*
*****
* create_random_a_string
*
* create a random alphanumeric string, of random length between lo and
* hi and place them in designated buffer. Routine returns the actual
* length.
*
* parameters
* -----
* lo end of acceptable length range
* hi end of acceptable length range
*
* output
* -----
* actual length
* random alphanumeric string
*****
*/

```

```

int create_random_a_string( char *out_buffer, int length_lo, int length_hi )
{
    int i, actual_length ;

    actual_length = rand_integer( length_lo, length_hi ) ;

    for (i = 0; i < actual_length; i++)
    {
        out_buffer[i] = alnum[rand_integer( 0, 61 )] ;
    }
    out_buffer[actual_length] = '\0' ;

    return (actual_length);
}

/*
*****
* create_random_n_string
*
* create a random numeric string, of random length between lo and
* hi and place them in designated buffer. Routine returns the actual
* length.
*
* parameters
* -----
* lo end of acceptable length range
* hi end of acceptable length range
*
* output
* -----
* actual length
* random numeric string
*****
*/

int create_random_n_string( char *out_buffer, int length_lo, int length_hi )
{
    int i, actual_length ;

    actual_length = rand_integer( length_lo, length_hi ) ;

    for (i = 0; i < actual_length; i++)
    {
        out_buffer[i] = (char)rand_integer( 48,57 ) ;
    }
    out_buffer[actual_length] = '\0' ;

    return (actual_length);
}

/*
*****
* NUrand_val
*
* create a non-uniform random numeric value of type integer, of random
* value between lo and hi. Number is NOT placed in BUFFER, and IS
* simply RETURNED.
*
* Routine RETURNS the VALUE.
*
* parameters
* -----
* lo end of acceptable value range
* hi end of acceptable value range
*
* output
* -----

```

```

* random integer value RETURNED
*
*****
*/

int NUrand_val ( int A, int x, int y, int C )
{
    return((((rand_integer(0,A)|rand_integer(x,y))+C)%(y-x+1))+x);
}

/*
*****
* create_a_string_with_original
*
* create a random alphanumeric string, of random length between lo and
* hi and place them in designated buffer. Routine returns the actual
* length.
*
* the word "ORIGINAL" is placed at a random location in the buffer at
* random, for a given percent of the records.
*
* percent_to_set must be an integer value from 0 to 100.
* if 0, no records will be set. If 100, all records will be set.
*
* CANNOT USE ON STRINGS OF LENGTH LESS THAN 8 ! LOWER LIMIT MUST BE >
8 !
*
* parameters
* -----
* lo end of acceptable length range
* hi end of acceptable length range
* percentage of records to set to ORIGINAL
*
* output
* -----
* actual length
* random alphanumeric string with the word "ORIGINAL" is placed at a
* random location
*
*****
*/

int create_a_string_with_original( char *out_buffer, int length_lo,
                                int length_hi, int percent_to_set )
{
    int actual_length, start_pos ;

    actual_length = create_random_a_string( out_buffer, length_lo, length_hi ) ;

    if ( rand_integer( 1, 100 ) <= percent_to_set )
    {
        start_pos = rand_integer( 0, actual_length-8 ) ;
        strncpy(out_buffer+start_pos,"ORIGINAL",8) ;
    }

    return (actual_length);
}

/*
*****
*
* create_random_last_name
*
* parameters:
*   out_buffer - target buffer for the generated last name
*
* description:
*   create_random_last_name 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. the three strings are

```

```

* concatenated to generate the name
*
*****
*/

int create_random_last_name(char *out_buffer, int cust_num)
{
    int random_num;

    if (cust_num == 0)
        random_num = NUrand_val( A_C_LAST, 0, 999, C_C_LAST_LOAD );
    else
        random_num = cust_num - 1;

    strcpy(out_buffer, last_name_parts[random_num / 100]);
    random_num %= 100;
    strcat(out_buffer, last_name_parts[random_num / 10]);
    random_num %= 10;
    strcat(out_buffer, last_name_parts[random_num]);

    return(strlen(out_buffer));
}

include/db2tpcc.h

/*
*****
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****
*/

/*
* db2tpcc.h - Macros and Miscellany
*
*/

#ifndef __DB2TPCC_H
#define __DB2TPCC_H

#include <sys/types.h>

#include "lval.h"

/*
*****
*/
/* Transaction Return Codes (s_transtatus) */
/*
*****
*/

#define INVALID_ITEM 100
#define TRAN_OK 0
#define FATAL_SQLERROR -1

/*
*****
*/
/* Definition of Unused and Bad Items */
/*
*****
*/
/* Define unused item ID to be 0. This allows the SUT to determine the
/* number of items in the order as required by 2.4.1.3 and 2.4.2.2 since
/* the assumption that any item with OL_I_ID = 0 is unused will be true.
/* This in turn requires that the value used for an invalid item is
/* equal to ITEMS + 1.
/*
*****
*/

#define INVALID_ITEM_ID (2 * ITEMS) + 1
#define UNUSED_ITEM_ID 0

```

```

#define MIN_WAREHOUSE 1
#define MAX_WAREHOUSE WAREHOUSES

/*
*****
*/
/* NURand Constants */
/* C_C_LAST_RUN and C_C_LAST_LOAD must adhere to clause 2.1.6.
/* Analysis indicates that a C_LAST delta of 85 is optimal.
/*
*****
*/

#define C_C_LAST_RUN 88
#define C_C_LAST_LOAD 173
#define C_C_ID 319
#define C_OL_I_ID 3849
#define A_C_LAST 255
#define A_C_ID 1023
#define A_OL_I_ID 8191

/*
*****
*/
/* Transaction Type Identifiers */
/*
*****
*/

#define CLIENT_SQL 0
#define NEWORD_SQL 1
#define PAYMENT_SQL 2
#define ORDSTAT_SQL 3
#define DELIVERY_SQL 4
#define STOCKLEV_SQL 5

#define SPGENERAL_PAD 3
#define SPGENERAL_ADJUST sizeof(int16_t)

struct in_neword_struct {
    int16_t len;
    int16_t pad[SPGENERAL_PAD];
    struct in_items_struct {
        int32_t s_OL_I_ID;
        int32_t s_OL_SUPPLY_W_ID;
        int16_t s_OL_QUANTITY;
        int16_t pad1[3];
    } in_item[15];
    int32_t s_C_ID;
    int32_t s_W_ID;
    int16_t s_D_ID;
    int16_t s_O_OL_CNT; /* init by SUT */
    int16_t s_all_local;
    int16_t duplicate_items;
};

struct out_neword_struct {
    int16_t len;
    int16_t pad[SPGENERAL_PAD];
    struct items_struct {
        float s_I_PRICE;
        float s_OL_AMOUNT;
        int16_t s_S_QUANTITY;
        int16_t pad2;
        char s_I_NAME[25];
        char s_brand_generic;
    } item[15];
    float s_W_TAX;
    float s_D_TAX;
    float s_C_DISCOUNT;
    float s_total_amount;
    int32_t s_O_ID;
    int16_t s_O_OL_CNT;
    int16_t s_transtatus;
    int16_t deadlocks;
    char s_C_LAST[17];
    char s_C_CREDIT[3];
    char s_O_ENTRY_D_time[27];
};

```

```
};
```

```
struct in_payment_struct {
    int16_t len;
    int16_t pad[SPGENERAL_PAD];
    float s_H_AMOUNT;
    int32_t s_W_ID;
    int32_t s_C_W_ID;
    int32_t s_C_ID;
    int16_t s_C_D_ID;
    int16_t s_D_ID;
    char s_C_LAST[17];
};
```

```
struct out_payment_struct {
    int16_t len;
    int16_t pad[SPGENERAL_PAD];
    double s_C_CREDIT_LIM;
    double s_C_BALANCE;
    float s_C_DISCOUNT;
    int32_t s_C_ID;
    int16_t s_transtatus;
    int16_t deadlocks;
    char s_W_STREET_1[21];
    char s_W_STREET_2[21];
    char s_W_CITY[21];
    char s_W_STATE[3];
    char s_W_ZIP[10];
    char s_D_STREET_1[21];
    char s_D_STREET_2[21];
    char s_D_CITY[21];
    char s_D_STATE[3];
    char s_D_ZIP[10];
    char s_C_FIRST[17];
    char s_C_MIDDLE[3];
    char s_C_LAST[17];
    char s_C_STREET_1[21];
    char s_C_STREET_2[21];
    char s_C_CITY[21];
    char s_C_STATE[3];
    char s_C_ZIP[10];
    char s_C_PHONE[17];
    char s_C_CREDIT[3];
    char s_C_DATA[201];
    char s_H_DATE_time[27];
    char s_C_SINCE_time[27];
};
```

```
struct in_ordstat_struct {
    int16_t len;
    int16_t pad[SPGENERAL_PAD];
    int32_t s_C_ID;
    int32_t s_W_ID;
    int16_t s_D_ID;
    int16_t pad1[3];
    char s_C_LAST[17];
};
```

```
struct out_ordstat_struct {
    int16_t len;
    int16_t pad[SPGENERAL_PAD];
    double s_C_BALANCE;
    int32_t s_C_ID;
    int32_t s_O_ID;
    int16_t s_O_CARRIER_ID;
    int16_t s_ol_cnt;
    int16_t pad1[2];
    struct oitems_struct {
        double s_OL_AMOUNT;
        int32_t s_OL_I_ID;
    };
};
```

```
int32_t s_OL_SUPPLY_W_ID;
int16_t s_OL_QUANTITY;
int16_t pad2;
char s_OL_DELIVERY_D_time[27];
} item[15];
int16_t s_transtatus;
int16_t deadlocks;
char s_C_FIRST[17];
char s_C_MIDDLE[3];
char s_C_LAST[17];
char s_O_ENTRY_D_time[27];
int16_t pad3[2];
};
```

```
struct in_delivery_struct {
    int16_t len;
    int16_t pad[SPGENERAL_PAD];
    int32_t s_W_ID;
    int16_t s_O_CARRIER_ID;
};
```

```
struct out_delivery_struct {
    int16_t len;
    int16_t pad[SPGENERAL_PAD];
    int32_t s_O_ID[10];
    int16_t s_transtatus;
    int16_t deadlocks;
};
```

```
struct in_stocklev_struct {
    int16_t len;
    int16_t pad[SPGENERAL_PAD];
    int32_t s_threshold;
    int32_t s_W_ID;
    int16_t s_D_ID;
};
```

```
struct out_stocklev_struct {
    int16_t len;
    int16_t pad[SPGENERAL_PAD];
    int32_t s_low_stock;
    int16_t s_transtatus;
    int16_t deadlocks;
};
```

```
/* ***** */
/* Transaction Prototypes */
/* ***** */
```

```
#ifdef __cplusplus
extern "C" {
#endif
```

```
extern int neword_sql(struct in_neword_struct*, struct out_neword_struct*);
extern int payment_sql(struct in_payment_struct*, struct out_payment_struct*);
extern int ordstat_sql(struct in_ordstat_struct*, struct out_ordstat_struct*);
extern int delivery_sql(struct in_delivery_struct*, struct out_delivery_struct*);
extern int stocklev_sql(struct in_stocklev_struct*, struct out_stocklev_struct*);
```

```
#ifdef __cplusplus
}
#endif
```

```
/* ***** */
/* DB2 Connect/Disconnect & Thread Context Wrappers */
/* ***** */
```

```
#ifdef __cplusplus
extern "C" {
#endif
```

```
extern int connect_to_TM(char*);
extern int connect_to_TM_auth(char*, char*, char*);
extern int disconnect_from_TM(void);
```

```
#ifndef __cplusplus
}
#endif
```

```
#endif // __DB2TPCC_H
```

include/lval.h

```
#ifndef __LVAL_H
#define __LVAL_H
#define WAREHOUSES 28032
#define DISTRICTS_PER_WAREHOUSE 10
#define CUSTOMERS_PER_DISTRICT 3000
#define ITEMS 100000
#define STOCK_PER_WAREHOUSE 100000
#define MIN_OL_PER_ORDER 5
#define MAX_OL_PER_ORDER 15
#define NU_ORDERS_PER_DISTRICT 900
#endif // __LVAL_H
```

include/platform.h

```
/* *****
** Licensed Materials - Property of IBM
**
** Governed under the terms of the International
** License Agreement for Non-Warranted Sample Code.
**
** (C) COPYRIGHT International Business Machines Corp. 1996 - 2005
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
***** */
```

```
/*
 * platform.h - Platform Isolation Layer
 */
```

```
#ifndef __PLATFORM_H
#define __PLATFORM_H
```

```
/* *****
/* Generic Macros */
/* *****
#define GEN_ERRCODE errno
```

```
/* *****
/* Windows I/O Macros */
/* *****
```

```
/* *****
/* UNIX I/O Macros */
/* *****
#include <fcntl.h>
```

```
#define IOH_INIT(hnd, type, name) \
    hnd->fd = -1; \
    hnd->type = type; \
    hnd->name = name;
```

```
#define IOH_CREATE(hnd)
```



```

if (hnd->type == IOH_PIPE) {
    rc = mkfifo(hnd->name, 0666);
} else {
    rc = 0;
}

#define IOH_OPEN(hnd)
if (hnd->type == IOH_FILE_APPEND) {
    hnd->fd = open(hnd->name, O_WRONLY | O_CREAT | O_APPEND, 0666);
} else {
    hnd->fd = open(hnd->name, O_WRONLY | O_CREAT | O_TRUNC, 0666);
}
if (hnd->fd == -1) {
    rc = -1;
} else {
    rc = 0;
}

#define IOH_WRITE(hnd, buff, num, num2)
rc = write(hnd->fd, buff, num);
if (rc >= 0) {
    num2 = rc;
    rc = 0;
}

#define IOH_FLUSH(hnd) rc = 0;
#define IOH_CLOSE(hnd) rc = close(hnd->fd);
#define IOH_DELETE(hnd) if (hnd->type == IOH_PIPE) { rc = unlink(hnd->name); }

```

```

typedef unsigned int IOH_NUM;
typedef int IOH_HND;

```

```

/*****
 * UNIX Semaphore Macros
 *****/

```

```

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>

```

```

union semun {
    int val;
    struct semid_ds *buf;
    unsigned short int *array;
} semUnion;

```

```

struct sembuf semBuf;

```

```

#define SEM_HANDLE int

```

```

#define SEM_INIT(hnd, x, name)
if ( (hnd = semget(IPC_PRIVATE, 1, IPC_CREAT | IPC_EXCL | S_IRUSR | S_IWUSR |
S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH)) == -1)
    API_ERROR(__LINE__, "semget", (rc=GEN_ERRCODE));
semUnion.val = x;
if ( semctl(hnd, 0, SETVAL, semUnion) < 0 )
    API_ERROR(__LINE__, "semctl SETVAL", (rc=GEN_ERRCODE));

```

```

#define SEM_WAIT(hnd)
semBuf.sem_num = 0;
semBuf.sem_op = -1;
semBuf.sem_flg = SEM_UNDO;
if ( semop(hnd, &semBuf, 1) < 0 )
    API_ERROR(__LINE__, "semop wait", (rc=GEN_ERRCODE));

```

```

#define SEM_FREE(hnd)
semBuf.sem_num = 0;
semBuf.sem_op = 1;
semBuf.sem_flg = SEM_UNDO;
if ( semop(hnd, &semBuf, 1) < 0 )
    API_ERROR(__LINE__, "semop free", (rc=GEN_ERRCODE));

```

```

#define SEM_DESTROY(hnd)
if ( semctl(hnd, 0, IPC_RMID, 0) )
    API_ERROR(__LINE__, "semctl IPC_RMID", (rc=GEN_ERRCODE));

```

```

/*****
 * Common I/O Macros and Definitions
 *****/

```

```

#define IOH_FILE 1
#define IOH_PIPE 2
#define IOH_FILE_APPEND 3

```

```

#define IOH_ERRMSG(hnd, msg)
if (rc != 0) {
    fprintf(stderr, "Error %d %s fd %d (%d, %s)\n", GEN_ERRCODE, msg,
        hnd->fd, hnd->type, hnd->name);
    return rc;
}

```

```

struct _ioh {
    IOH_HND fd;
    int type;
    char *name;
};

```

```

typedef struct _ioh ioHandle;

```

```

/*****
 * Generic I/O Routine Prototypes
 *****/

```

```

int GenericOpen(ioHandle *hnd, int type, char *name);
int GenericWrite(ioHandle *hnd, char *Buffer, unsigned int numBytes);
int GenericClose(ioHandle *hnd);

```

```

/*****
 * Generic I/O Routines
 *****/

```

```

int GenericOpen(ioHandle *hnd, int type, char *name)

```

```

{
    int rc = 0;

    IOH_INIT(hnd, type, name)

```

```

    IOH_CREATE(hnd)
    IOH_ERRMSG(hnd, "creating")

```

```

    IOH_OPEN(hnd)
    IOH_ERRMSG(hnd, "opening")

```

```

    return rc;
}

```

```

int GenericWrite(ioHandle *hnd, char *Buffer, unsigned int numBytes)

```

```

{
    int rc = 0;
    int numBytesWritten = -1;

```

```

    IOH_WRITE(hnd, Buffer, numBytes, numBytesWritten)
    IOH_ERRMSG(hnd, "writing")

```

```

    if (numBytes != numBytesWritten) {
        fprintf(stderr, "Truncated data writing to fd %d (%d, %s)\n", hnd->fd, hnd->type, hnd->name);
        rc = -1;
    }

```

```

    return rc;
}

```

```

int GenericClose(ioHandle *hnd)

```

```

{

```

```

int rc = 0;

```

```

    IOH_FLUSH(hnd)
    IOH_ERRMSG(hnd, "flushing")

```

```

    IOH_CLOSE(hnd)
    IOH_ERRMSG(hnd, "closing")

```

```

    IOH_DELETE(hnd)
    IOH_ERRMSG(hnd, "deleting")

```

```

    return rc;
}

```

```

#endif // __PLATFORM_H

```

include/tpccmisc.h

```

/*****

```

```

** Licensed Materials - Property of IBM

```

```

**

```

```

** Governed under the terms of the International

```

```

** License Agreement for Non-Warranted Sample Code.

```

```

**

```

```

** (C) COPYRIGHT International Business Machines Corp. 1996 - 2005

```

```

** All Rights Reserved.

```

```

**

```

```

** US Government Users Restricted Rights - Use, duplication or

```

```

** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

```

```

*****/

```

```

/*
 * tpccmisc.h - Miscellaneous Routines
 */

```

```

#ifndef __TPCCMISC_H
#define __TPCCMISC_H

```

```

extern double current_time_ms(void);
extern double current_time(void);

```

```

#include <time.h>
#define createTimestampString(buf)

```

```

{
    time_t now;
    struct tm *tm;
    time(&now);
    tm = localtime(&now);
    sprintf(buf,
        "%4.4d-%2.2d-%2.2d %2.2d:%2.2d:%2.2d",
        tm->tm_year + 1900, tm->tm_mon + 1, tm->tm_mday,
        tm->tm_hour, tm->tm_min, tm->tm_sec);
}

```

```

#endif // __TPCCMISC_H

```

include/tpccrnd.h

```

/*****

```

```

** Licensed Materials - Property of IBM

```

```

**

```

```

** Governed under the terms of the International

```

```

** License Agreement for Non-Warranted Sample Code.

```

```

**

```

```

** (C) COPYRIGHT International Business Machines Corp. 1996 - 2006

```

```

** All Rights Reserved.

```

```
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****/
```

```
/*
 * tpcrnd.h - Random generation functions for TPC-C
 */
```

```
#ifndef __TPCCRD_H
#define __TPCCRD_H
```

```
void initialize_random(void);
int rand_integer( int val_lo, int val_hi );
double rand_decimal( int val_lo, int val_hi, int val_dec );
int NUrand_val( int A, int val_lo, int val_hi, int C );
```

```
void seed_1_3000( void );
int random_1_3000( void );
```

```
int create_random_a_string( char *out_buffer,
                           int length_lo,
                           int length_hi );
```

```
int create_random_n_string( char *out_buffer,
                           int length_lo,
                           int length_hi );
```

```
int create_a_string_with_original( char *out_buffer,
                                  int length_lo,
                                  int length_hi,
                                  int percent_to_set );
```

```
int create_random_last_name(char *out_buffer, int cust_num);
```

```
#endif // __TPCCRD_H
```

tpccenv.sh

```
#####
####
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp. 1996 - 2006
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use, duplication or
## disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#####
```

```
#
# tpccenv.sh - UNIX Environment Setup
#
```

```
# The Kit Version
export TPCC_VERSION=CK060815
```

```
# The DB2 Instance Name (for DB2)
export DB2INSTANCE=${USER}
```

```
# The OS being used (i.e. "UNIX", "LINUX", "WINDOWS")
export PLATFORM=LINUX
```

```
# The type of make command and slash used by the OS.
# (i.e. UNIX - "/", WINDOWS - "\").
# These are referenced all over the kit.
export SLASH="/";
export MAKE=make
```

```
# Specifies whether or not to use dari stored proc's for the TPC-C driver. Set to either
DARIVERSION or NONDARI:
#export TPCC_SPTYPE=NOSP
#export TPCC_SPTYPE=SPGENERAL2
export TPCC_SPTYPE=SPGENERAL
#export TPCC_SPTYPE=DARI2SQLDA
```

```
export DB2VERSION=v8
```

```
# The schema name is typically the SQL authorization ID (or username).
# This is required for runstats and EEE.
export TPCC_SCHEMA=${USER}
```

```
# DB2 EE/EEE Configuration
export DB2EDITION=EE
#export DB2EDITION=EEE
export DB2NODE=0
export DB2NODES=1; # set to the number of nodes you have. Set to 1 for EE.
```

```
# TPCC General Configuration
export TPCC_DBNAME=TPCC
export TPCC_ROOT=${HOME}/tpc-c.ibm
export TPCC_SQLLIB=${HOME}/sqllib
export TPCC_RUNDATA=${HOME}/tpccdata
```

```
# TPCC Debug Configuration
# This is the path where all error and debug logs are placed.
# To get debugging from within the stored procedures, you must
# set DB2ENVLIST="TPCC_DEBUGDIR" in tpcc.config.
export TPCC_DEBUGDIR=/tmp
```

```
# Specifies where stored procedures should be placed and if they should
# be fenced.
export TPCC_SPDIR=${TPCC_SQLLIB}/function
export TPCC_FENCED=NO
```

12 Appendix D: Pricing

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Tel 425 882 8080
Fax 425 936 7329
<http://www.microsoft.com/>

Microsoft

November 21, 2006

IBM Corporation
Chris King
3039 Cornwallis Road
Research Triangle Park, NC 27709

Ms. King:

Here is the information you requested regarding pricing for several Microsoft products to be used in conjunction with your TPC-C benchmark testing.

All pricing shown is in US Dollars (\$).

Part Number	Description	Unit Price	Quantity	Price
C11-00821	Windows 2000 Server Server License Only - No CALs Discount Schedule: No Level Unit Price reflects a 8% discount from the retail unit price of \$799.	\$738	16	\$11,808
254-00170	Visual C++ Standard Edition No Discounts Applied	\$109	1	\$109
N/A	Microsoft Problem Resolution Services Professional Support (1 Incident)	\$245	1	\$245

All products are currently orderable through Microsoft's normal distribution channels. A list of Microsoft's authorized resellers can be found at: <http://www.microsoft.com/products/info/render.aspx?view=22&type=mnpc&content=22/licensing>.

Defect support is included in the purchase price. Additional support is available from Microsoft PSS on an incident by incident basis at \$245 per call.

This quote is valid for the next 90 days.

If we can be of any further assistance, please contact Jamie Reding at (425) 703-0510 or jamiere@microsoft.com.

Reference ID: PCchki0622016404.

Please include this Reference ID in any correspondence regarding this price quote.



November 27, 2006

IBM Corporation
Ms. Celia Schreiber
xSeries Performance

Dear Celia:

The table shown below lists the U.S. pricing for DB2 9 Data server product that has been used in the TPC-C Benchmark.

All prices shown are in U.S. Dollars.

DB2 Enterprise Server Edition (ESE)	VUs	Reference Price per value unit	Total Reference price
DB2 Enterprise Proc 9 Lic/1 year Maintenance	400	262.65	105,060
SW Maintenance Renewal - 1 year	800	12.51	10,008
		Sub-total reference price for DB2 ESE:	115,068
		TOTAL REFERENCE PRICE:	115,068

Any and all prices herein are suggested prices only and are subject to change at IBM's sole discretion. Products listed herein are subject to withdrawal or modification by IBM at any time at IBM's sole discretion.

Sincerely,

Richard Hughes
IBM Sales & Distribution, Software Sales
Americas Sales Executive DB2 and Informix
212-493-2065
rhughes@us.ibm.com



[Home](#) | [About Us](#) | [Customer Support](#) | [View Cart](#) | [Log On](#)


[Brands](#) | [Hardware](#) | [Software](#) | [Networking](#) | [Accessories](#) | [Services](#) | [Product Finders](#) | **800.750.4239**

[\\$200 mail-in rebate w/ Autodesk AutoCAD 2007](#)

Filter YOUR Search Results

Your selected filters have been applied to the results displayed on this page.

To modify your results, click the appropriate criteria line item or click to remove the entire line.



 **Take Our Survey**
Was our search engine helpful?

Keyword: 430446

Show ready to ship products only Hide Product images

Sort By: **BEST MATCH** | [GROUP](#) | [BRAND](#) | [LOWEST PRICE](#) | [HIGHEST PRICE](#)

Showing 1 - 1 of 1 Products

<input type="button" value="Compare"/>	Product Details	CDW#	Availability	Advertised Price
<input type="checkbox"/>	 <p><u>Linksys ProConnect KVM switch - 2 ports</u> KVM switch - 2 ports - 1 local user MFG# : KVM2KIT</p>	430446	In Stock	\$39.99 

Showing 1 - 1 of 1 Products

The Right Technology. Right Away.™

SEARCH: [Go](#)

PHONE ORDERS: 800.287.2323

[HOME](#) | [ABOUT US](#) | [PRODUCT RETURN](#) | [CUSTOMER SERVICE](#) | [CONTACT US](#) | [PHONE ORDERS](#) | [REBATES](#)

Home > Networking > Gigabit

Select a Brand

[GO](#)

- [AUDIO](#)
- [AUTO ELECTRONICS](#)
- [CAMCORDERS](#)
- [CAMERAS](#)
- [CASES](#)
- [CELLULAR](#)
- [COMPUTER SYSTEM](#)
- [CONTROLLERS](#)
- [DVD & CD](#)
- [FAX MACHINES](#)
- [GAMING](#)
- [GPS NAVIGATION](#)
- [HEADSETS](#)
- [HOUSEWARES](#)
- [INPUT DEVICES](#)
- [MEMORY](#)
- [MODEMS](#)
- [MONITORS](#)
- [MOTHERBOARDS](#)
- [MP3](#)
- [NETWORKING](#)
- [PDA HANDHELDS](#)
- [POWER](#)
- [PRINTERS](#)
- [PROCESSORS](#)
- [PROJECTORS](#)
- [RADIOS](#)
- [ROUTERS](#)
- [SCANNERS](#)
- [SHAVERS](#)
- [SOFTWARE](#)

D-LINK DGS-1024D 24-PORT 10/100/1000 UNMANAGED RACKMOUNTABLE DESKTOP SWITCH

[See Full Description](#)

Holiday Special--Extended Return Policy: Returns accepted on gift purchases through January 6th! Click [here](#) for details.

Item is brand new and in Stock

Usually ships in 1-2 Business Days

Buy with Confidence

2 million satisfied customers since 1993

13 years of internet sales

100% customer satisfaction guarantee

All items factory fresh with a full USA warranty

30 day money back



Selling Elsewhere for : ~~\$279.00~~

Our Price Only: \$235.00

Buy Now

[Tell a friend](#)



FEATURED ACCESSORIES

Customers who purchased this also bought:

Cables

Category 5 E Gray Networking Patch Cable - 14 Foot

\$4.99 [More Info](#)

Category 5 E Gray Networking Patch Cable - 7 Foot

\$4.99 [More Info](#)

Category 5 E Gray Networking Patch Cable - 3 Foot

\$3.25 [More Info](#)

[MORE Cables](#)



Home > Categories > Wired Networking Devices > Network - Cables > AMC (ARROWMICRO) > Item #: N82E16812105305



AMC CC5E-B14B 14 FT Cat 5E Blue Cat 5E Blue Cable - OEM



Image Viewer



\$2.99

3 Business Day Shipping \$4.99
(Not available in HI, AK and PR)

In Stock

ADD TO CART

ADD TO WISH LIST

EMAIL THIS PAGE

PRINT THIS PAGE

PRICE ALERT

CUSTOMER REVIEWS SPECIFICATIONS

Model	
Brand	AMC
Model	CC5E-B14B
Spec	
Color	Blue
Cat	5E
Length	14 FT

Quantity Pricing

Units	Price	Shipping
1	\$2.99	\$4.99
2+	\$2.39	\$3.47 ea.
5+	\$2.29	\$2.43 ea.
10+	\$1.99	\$2.09 ea.

Special Offers



No Payments for 6 Months on purchases of \$500 or more with your Newegg.com Preferred Account! [Click here for important disclosures](#)



No Payments for 90 Days On purchases over \$250 with Bill Me Later®! [Click here for important disclosures](#)

MANUFACTURER INFO RETURNS & REBATES

Manufacturer Contact Info

- [Manufacturer Product Page](#)
- Website: <http://www.arrowmicro.com>
- Support Phone Number: 714-522-0232
- [View other products from AMC \(ARROWMICRO\)](#)

