
IBM x3650 M4 with KVM

Using

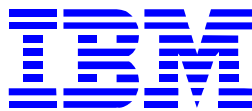
DB2 9.7 Enterprise Edition

and

Red Hat Enterprise Linux Server 6.4 with KVM

TPC BenchmarkTM C

Full Disclosure Report



First Edition, February 25, 2013

First Edition: February 25, 2013

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, 2013 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.

Trademarks

The following terms used in this publication are trademarks or registered trademarks of **International Business Machines Corporation** in the United States and/or other countries: IBM, System x, System Storage, ServeRAID and 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 2008 server and COM+ are registered trademarks of Microsoft Corporation.

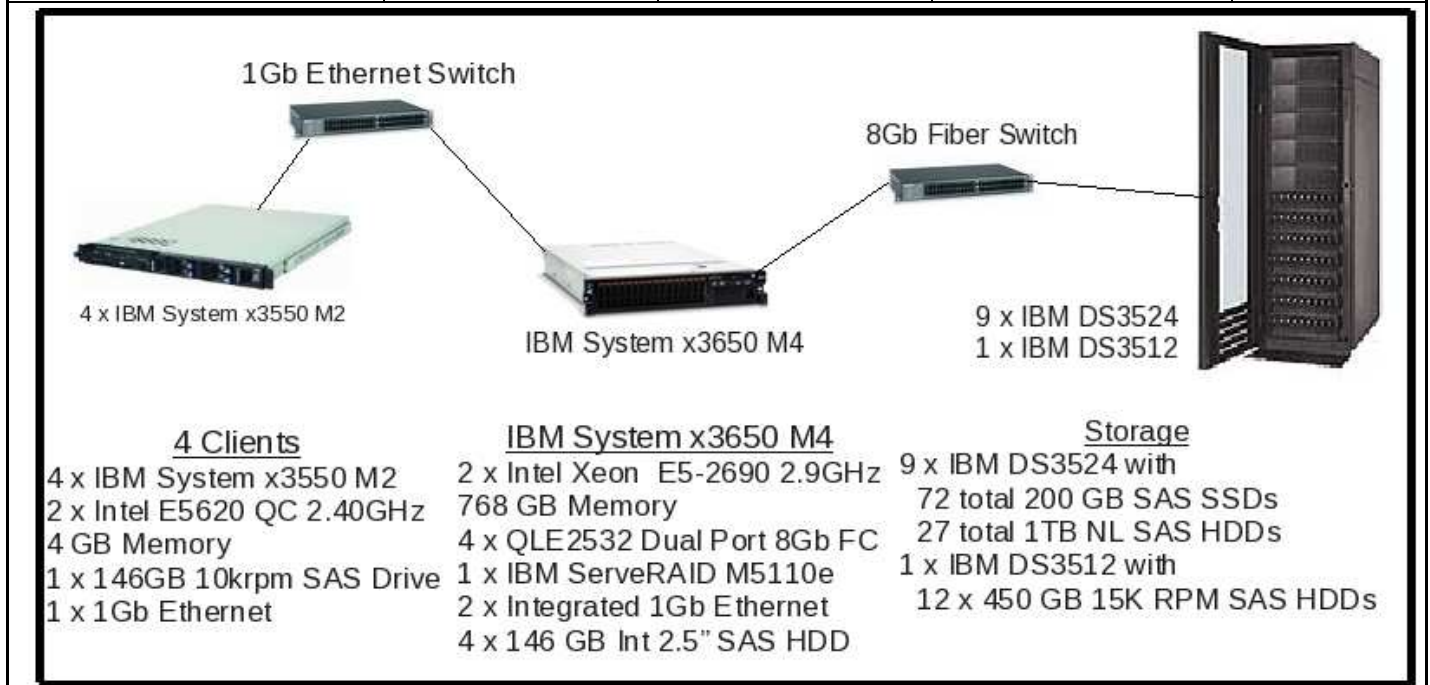
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

RedHat is a registered trademarks of RedHat, Inc.

Intel and Xeon are registered trademarks of Intel Corporation.

	IBM x3650 M4 with KVM DB2® 9.7		TPC-C Rev. 5.11
			Report Date: February 25, 2013

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date	
\$667,882 USD	1,320,082	\$0.51 USD	February 25, 2013	
Database Processors/Cores/Threads	Database Manager	Operating System	Other Software	No. Users
2/16/32 2.90 GHz Intel Xeon E5-2690	DB2 9.7	RHEL 6.4 with KVM	Microsoft Visual C++ Microsoft COM+	1,040,400



System Components	Each of the 4 Clients		Server	
	Quantity	Description	Quantity	Description
Processors/Cores/Threads	2/8/16	2.4GHz Intel Xeon E5620 4 x 256KB L2, 1 x 12MB L3	2/16/32	2.90 GHz Intel Xeon E5-2690 8 x 256KB L2, 1 x 20MB L3
Memory	2	2 GB	24	32 GB
Disk Controllers	1	SAS	4 1	QLogic 8Gb FC Dual-port HBA IBM ServeRAID M5110e
Disk Drives	1	146 GB	27 12 72 4	1 TB 7200 RPM NL SAS 450 GB 15K RPM SAS 200GB SAS 2.5" SSD drives 146GB 15K RPM SAS HDD
Total Storage		146 GB		40.33 TB
Terminals	1	System Console	1	System Console

IBM Corporation		IBM x3650 M4 with KVM and DB2 9.7				TPC-C Revision 5.11			
Description		Part Number	Brand	Price Source	Unit Price	Qty	Extended Price	3-Yr. Maint. Price	
Server Hardware									
IBM System x3650 M4	7915AC1	IBM	1	59275	1	59275			
x3650 M4 PCIe Riser Card 1 (1 x8 FH/FL + 2 x8 FH/HL Slots)	A1JT		1		1				
1.5m Blue Cat5e Cable	3802		1		2				
x3650 M4 8x 2.5" H5 HDD Assembly Kit	A1JX		1		1				
IBM UltraSlim Enhanced SATA DVD-ROM	4161		1		1				
x3650 M4 PCIe Gen-III Riser Card 2 (1 x8 FH/FL + 2 x8 FH/HL Slots)	A1JU		1		1				
IBM System x3650 M4 2.5" Base without Power Supply	A1KF		1		1				
QLogic 8Gb FC Dual-port HBA for IBM System x	3579		1		4				
IBM System x 750W High Efficiency Platinum AC Power Supply	A1H5		1		2				
Addl Intel Xeon Processor E5-2690 8C 2.9GHz 20MB 135W W/Fan	A2QL		1		1				
ServeRAID M5110e SAS/SATA Controller for IBM System x	A2N2		1		1				
IBM System x Gen-III CMA	A229		1		1				
IBM System x Advanced Lightpath Kit	A2U6		1		1				
32GB (1x32GB, 1.5V) PC3-8500 CL7 ECC DDR3 1066MHz LP HyperCloud DIMM	A3EJ		1		24				
Intel Xeon Processor E5-2690 8C 2.9GHz 20MB Cache 1600MHz 135W	A2VN		1		1				
ServeRAID M5100 Series 1GB Flash/RAID 5 Upgrade for IBM System x	A1WY		1		1				
4.3m, 10A/100-250V, C13 to IEC 320-C14 Rack Power Cable	6263		1		2				
IBM 146GB 15K 6Gbps SAS 2.5" SFF G2H5 HDD	A2XB		1		4				
IBM System x Gen-III Slides Kit	A228		1		1				
x3650 M4 ODD Cable	A22N		1		1				
IBM System x3650 M4 Planar	A1KH		1	790	1				
3 Year Onsite Repair 24x7 4 Hour Response	67567XR	IBM	1		3			790	
Acer S181HLGb Black 18.5" 5ms Widescreen LED Backlight LCD Monitor (2 spares)	N82E1682400939	Newegg	2	95.99	3	287.97			
Kensington K72436AM Black USB Wired Standard Keyboard for Life Desktop Set (2 spares)	N82E1682315506	Newegg	2	19.99	3	59.97			
						Subtotal	59,623	790	
Server Storage									
IBM System Storage DS3524 Express	1746C4A	IBM	1	54322	9	488898			
Turbo Performance	4400		1		9				
Fiber Cable, 5 meter multimode LC-LC	5605		1		18				
IBM System Storage DS3524 Express Dual Controller Storage System	2812		1		9				
8Gb FC 4 Port Daughter Card (2)	3622		1		18				
United States 10A line C13 to NEMA 5-15P (2.8M)	6313		1		18				
200GB 2.5" 3Gb SAS SSD	5500		1		72				
1TB 7,200 rpm 6Gb SAS NL 2.5" HDD	5270		1		27				
3 Year Onsite Repair 24x7 4 Hour Response (DS3524)	67567DT	IBM	1	1350	9			12150	
IBM System Storage DS3512 Express	1746C2A	IBM	1	19241	1	19241			
Fiber Cable, 5 meter multimode LC-LC	5605		1		2				
IBM System Storage DS3512 Express Dual Controller Storage System	2802		1		1				
8Gb FC 4 Port Daughter Card (2)	3622		1		2				
United States 10A line C13 to NEMA 5-15P (2.8M)	6313		1		2				
450GB 15,000 rpm 6Gb SAS 3.5" HDD	5105		1		12				
3 Year Onsite Repair 24x7 4 Hour Response (DS3512)	67567DN	IBM	1	1350	1			1350	
IBM System Storage SAN24B-4 Express	2498B24	IBM	1	15850	1	15850			
8 Port Activation	7200		1		2				
Fiber Cable LC/LC 5M Multimode	5605		1		6				
SFP Transceiver 8 Gbps SW	2801		1		24				
3 Year Onsite Repair 24x7 4 Hour Response (SAN23B-4)	6756692	IBM	1	1785	1			1785	
NetBAY S2 42U Rack Cabinet	9307RC4	IBM	1	1459	1	1459			
						Subtotal	525,448	15,285	
Server Software									
DB2 ESE 9.7 5W License and Maintenance – 12 Months		IBM	1	451	1120	505,120			
DB2 ESE 9.7 5W Maintenance Renewal – 1 Year		IBM	1	90	2240			201,600	
Red Hat Enterprise Linux Server, Premium (1-2 sockets) (Up to 1 guest)		Red Hat	3	1299	3	3897			
						Subtotal	509,017	201,600	
Client Hardware									
IBM System x3550 M3	7944AC1	IBM	1	3924	4	15696			
PCI-Express (1 x16 slot; LP) Riser Card 1	5076		1		4				
2GB (1x2GB, 1Rx4, 1.5V) PC3-10600 CL9 ECC DDR3 1333MHz LP RDIMM	8935		1		8				
1.5m Blue Cat5e Cable	3802		1		4				
IBM System x3550 M3 4 HDD and Optical Drive Kit	1788		1		4				
IBM UltraSlim Enhanced SATA DVD-ROM	4161		1		4				
PCI-Express (1 x16; FH/HL) Riser Card 2	4375		1		4				
Base with 460W Redundant AC Power Supply	A0ZX		1		4				
IBM 460W Redundant Power Supply Unit	A0ZH		1		4				
Addl Intel Xeon Proc. E5620 4C 2.40GHz 12MB Cache 1066MHz 80w W/Fan	4600		15		4				
ServeRAID-BR10i SAS/SATA Controller	3577		1		4				
IBM 146GB 15K 6Gbps SAS 2.5" SFF Slim-HS HDD	5536		1		4				
Intel Xeon Processor E5620 4C 2.40GHz 12MB Cache 1066MHz 80w	4586		15		4				
2.8m, 10A/100-250V, C13 to IEC 320-C14 Rack Power Cable	6311		1		8				
System Common Planar for 1U/2U	5663		1		4				
3 Year Onsite Repair 24x7 4 Hour Response	6756298	IBM	1	495	4			1980	
						Subtotal	15,696	1,980	
Client Software									
Windows Server Web Edition 2008 R2	LWA-00984	Microsoft	4	469	4	1,876			
Microsoft Visual Studio 2008 Professional	127-00166	Microsoft	4	250	1	250			
Microsoft Problem Resolution Services	NA	Microsoft	4	259	1			259	
						Subtotal	2,126	259	
Third-Party Components									
UPS – powercom KIN-1500AP 1500 VA 900 Watts (2 spares)	N82E1684210611	Newegg	2	149.99	3	449.97			
NETGEAR 48 Port Gigabit Smart Switch w/ 4 Combo ports – GS748T (2 spares)	N82E1683312214	Newegg	2	499.99	3	1499.97			
						Subtotal	1,950	0	
						Total	1,113,860	219,914	
IBM Dollar Volume Discount (See Note 1)							Note 1 Discount	665,951	
Pricing: 1 - IBM - 1-800-656-0833 x35334; 2 – newegg.com; 3 – Red Hat; 4 – Microsoft							Three-Year Cost of Ownership USD: 667,822		
Note 1: Discount based on IBM Direct guidance applies to all line items where Pricing=1.							tpmC: 1,320,082		
Pricing is for this system and software or one of similar size.							\$ USD/tpmC: 0.51		
S - One or more components of the measured configuration have been substituted in the Priced Configuration. See the FDR for details									
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. Thank you.									

Numerical Quantities Summary for the IBM System x 3650 M4 with KVM Virtualization

MQTH, computed Maximum Qualified Throughput: 1,320,082 tpmC

<u>Response Times (in seconds)</u>	<u>90th %</u>	<u>Average</u>	<u>Maximum</u>
New Order	0.160	0.124	1.875
Payment	0.160	0.121	3.078
Order-Status	0.160	0.122	0.953
Delivery (interactive)	0.160	0.117	0.703
Delivery (deferred)	0.03	0.02	1.01
Stock-Level	0.160	0.123	1.110
Menu	0.141	0.117	1.046

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	44.953%
Payment	43.018%
Order-Status	4.011%
Delivery	4.008%
Stock-Level	4.010%

<u>Keying/Think Times (in seconds)</u>	<u>Min.</u>	<u>Average</u>	<u>Max.</u>
New Order	18.000/0.00	18.000/12.035	18.062/120.343
Payment	3.000/0.00	3.000/12.037	3.062/120.344
Order-Status	2.000/0.00	2.000/10.033	2.047/100.329
Delivery	2.000/0.00	2.000/5.036	2.047/50.328
Stock-Level	2.000/0.00	2.000/5.035	2.047/50.328

Test Duration

Ramp-up Time	41 minutes 51 seconds
Measurement interval	2 hours 0 minutes
Transactions during measurement interval (all types)	352,391,356

Checkpoints

Number of checkpoints	N/A
Checkpoint interval	N/A

Table of Contents

0.	General Items.....	- 10 -
0.1.	Application Code Disclosure.....	- 10 -
0.2.	Benchmark Sponsor.....	- 10 -
0.3.	Parameter Settings.....	- 10 -
0.4.	Configuration Diagrams.....	- 10 -
1.	Clause 1: Logical Data Base Design Related Items.....	- 12 -
1.1.	Table Definitions.....	- 12 -
1.2.	Database Organization.....	- 12 -
1.3.	Insert and/or Delete Operations.....	- 12 -
1.4.	Horizontal or Vertical Partitioning.....	- 12 -
2.	Clause 2: Transaction & Terminal Profiles Related Items.....	- 13 -
2.1.	Verification for the Random Number Generator.....	- 13 -
2.2.	Input/Output Screens.....	- 13 -
2.3.	Priced Terminal Features.....	- 13 -
2.4.	Presentation Managers.....	- 13 -
2.5.	Home and Remote Order-lines.....	- 13 -
2.6.	New-Order Rollback Transactions.....	- 13 -
2.7.	Number of Items per Order.....	- 14 -
2.8.	Home and Remote Payment Transactions.....	- 14 -
2.9.	Non-Primary Key Transactions.....	- 14 -
2.10.	Skipped Delivery Transactions.....	- 14 -
2.11.	Mix of Transaction Types.....	- 15 -
2.12.	Queuing Mechanism of Delivery.....	- 15 -
3.	Clause 3: Transaction and System Properties.....	- 16 -
3.1.	Atomicity Requirements.....	- 16 -
3.2.	Consistency Requirements.....	- 17 -
3.3.	Isolation Requirements.....	- 17 -
3.4.	Durability Requirements.....	- 18 -
4.	Clause 4: Scaling and Data Base Population Related Items.....	- 21 -
4.1.	Cardinality of Tables.....	- 21 -
4.2.	Distribution of Tables and Logs.....	- 21 -
4.3.	Data Base Model Implemented.....	- 22 -
4.4.	Partitions/Replications Mapping.....	- 22 -
4.5.	60-Day Space Calculations.....	- 24 -
5.	Clause 5: Performance Metrics and Response Time Related Items.....	- 25 -
5.1.	Response Times.....	- 25 -
5.2.	Keying and Think Times.....	- 25 -
5.3.	Response Time Frequency Distribution.....	- 26 -
5.4.	Performance Curve for Response Time versus Throughput.....	- 28 -
5.5.	Think Time Frequency Distribution.....	- 29 -
5.6.	Throughput versus Elapsed Time.....	- 30 -
5.7.	Steady State Determination.....	- 30 -
5.8.	Work Performed During Steady State.....	- 30 -
5.9.	Measurement Interval.....	- 32 -
6.	Clause 6: SUT, Driver, and Communication Definition Related Items.....	- 33 -
6.1.	RTE Availability.....	- 33 -

6.2.	Functionality and Performance of Emulated Components	- 33 -
6.3.	Network Bandwidth.....	- 33 -
6.4.	Operator Intervention.....	- 33 -
7.	Clause 7: Pricing Related Items	- 34 -
7.1.	Hardware and Programs Used	- 34 -
7.2.	Three Year Cost of System Configuration.....	- 34 -
7.3.	Availability Dates	- 34 -
7.4.	Statement of tpmC and Price/Performance.....	- 35 -
8.	Clause 9: Audit Related Items	- 36 -
9.	Appendix A: Client Server Code.....	- 38 -
9.1.	Client/Terminal Handler Code.....	- 38 -
9.2.	Client Transaction Code	- 49 -
10.	Appendix B: Tunable Parameters	- 79 -
10.1.	Database Parameters	- 79 -
10.2.	Transaction Monitor Parameters	- 81 -
10.3.	Linux KVM Host Parameters	- 85 -
10.4.	Linux KVM Guest Parameters.....	- 87 -
11.	Appendix C: Database Setup Code	- 90 -
11.1.	Database Creation Scripts	- 90 -
11.2.	Data Generation	- 256 -
12.	Appendix D: Pricing.....	- 267 -

Abstract

This report documents the full disclosure information required by the TPC Benchmark™ C Standard Specification Revision 5.11 dated Feb, 2010, for measurements on the IBM System x 3650 M4 with KVM Virtualization. The software used on the IBM System x 3650 M4 with KVM Virtualization includes Red Hat Enterprise Linux Server 6.4 with KVM operating system and DB2 9.7 data server. Microsoft COM+ is used as the transaction manager.

IBM System x 3650 M4 with KVM Virtualization

Company Name	System Name	Database Software	Operating System Software
IBM Corporation	IBM x3650 M4 with KVM	DB2 9.7	Red Hat Enterprise Linux Server 6.4 with KVM

Total System Cost	TPC-C Throughput	Price/Performance
<input type="checkbox"/> Hardware <input type="checkbox"/> Software <input type="checkbox"/> 3 Years Maintenance	Sustained maximum throughput of system running TPC-C expressed in transactions per minute	Total system cost/tpmC
\$667,882 USD	1,320,082	\$0.51 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 5.11 in Feb. 2010.

This is the full disclosure report for benchmark testing of the IBM System x 3650 M4 with KVM Virtualization and DB2 9.7 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.*

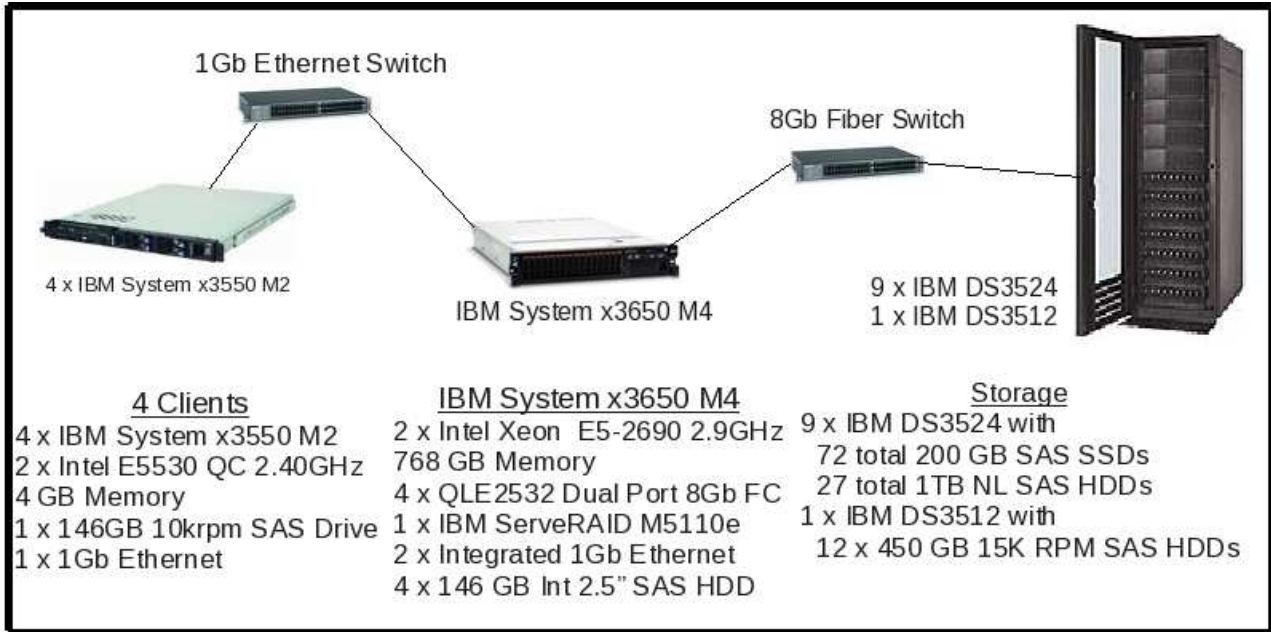
Appendix B contains the system, data base, and application parameters changed from their default values used in these TPC Benchmark™ C tests.

0.4. Configuration Diagrams

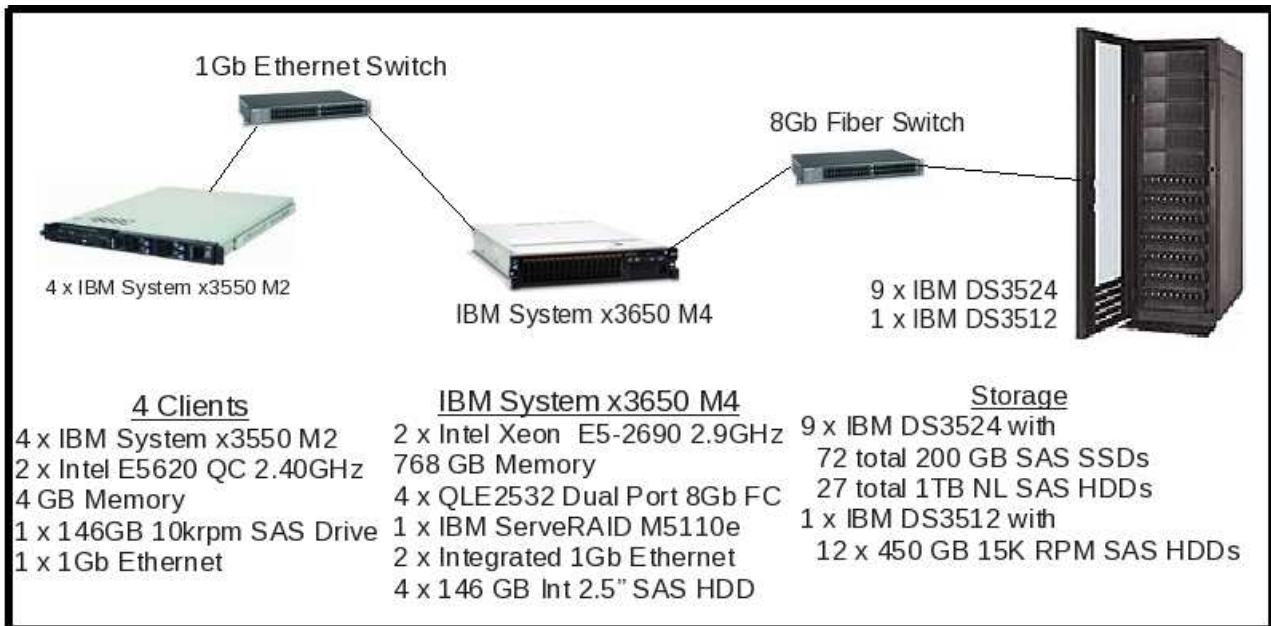
Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:

- *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 x 3650 M4 with KVM Virtualization Benchmark Configuration



IBM System x 3650 M4 with KVM Virtualization Priced Configuration



A total of 8 Intel Xeon E5530 quad core processors running at 2.40 GHz were used by the client System x 3550 M2 systems for the benchmark; 8 Intel Xeon E5620 quad core processors running at 2.40 GHz were priced for these processors and substituted on a one for one basis. The auditor reviewed the specifications of the processors and certified that the pricing upgrade was compliant with the TPC-C specification.

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.

For each partitioned table set, 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 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 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).

Microsoft Internet Explorer was used to verify the compliance with 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 x3650 M4 with KVM
Percentage of Home order lines	99.000%
Percentage of Remote order lines	1.000%
Percentage of Rolled Back Transactions	1.001%
Average Number of Items per order	10.000
Payment	
Percentage of Home transactions	85.003%
Percentage of Remote transactions	14.997%
Non-Primary Key Access	
Percentage of Payment using C_LAST	60.001%
Percentage of Order-Status using C_LAST	60.005%
Delivery	
Delivery transactions skipped	0
Transaction Mix	
New-Order	44.953%
Payment	43.018%
Order-Status	4.011%
Delivery	4.008%
Stock-Level	4.010%

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:

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

$$\text{where } (d_w_id = o_w_id = no_w_id) \text{ and } (d_id = o_d_id = no_d_id)$$

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 Controller:

This test was conducted on a 12.5% 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 12.5% load test was started and allowed to run and steady state was reached and maintained for over 5 minutes.
3. 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. The battery attached to the controller cache was removed and and let sit for 5 minutes.
4. The system was subsequently powered off.
5. The storage controller from step 3 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.
6. The system was powered back on and DB2 was allowed to recover.
7. 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
8. Consistency condition 3 was verified.

Failure of Log Disk:

This test was conducted on a 12.5% scaled database. The following steps were successfully performed:

9. 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.
10. A 12.5% load test was started and allowed to run and steady state was reached and maintained for over 5 minutes.
11. One of the disks containing the transaction log was removed. Since the log was implemented as a RAID-10 array, DB2 continued to process the transactions successfully.
12. The test continued for at least another 5 minutes.
13. The test was ended and the database was deactivated in a controlled manner.
14. 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
15. Consistency condition 3 was verified.

Failure of Durable Medium Containing TPC-C Database Tables:

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 and allowed to run and steady state was reached and maintained for over 5 minutes..
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.

Instantaneous Interruption, Memory Failure, and Loss of Power:

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 5 minutes.

3. The power was removed from the database server.
4. The database server was powered back on and DB2 was allowed to recover.
5. 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.
6. 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 4-2 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	104,040
District	1,040,400
Customer	3,121,200,000
History	3,121,200,000
Orders	3,121,200,000
New Order	936,360,000
Order Line	31,211,013,719
Stock	10,404,000,000
Item	100,000

Table 4-2: 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.

The log was configured using one Logical drive attached to one of the Fiber Channel disk controllers. The log was configured as a RAID-10 disk array consisting of 12 450GB hot-swap SAS disks housed in one DS3512 drive enclosure and backed by a UPS.

There are 72 Logical Disks (LD) for the database tables. There are 72 200GB SSD drives attached via four 2-port Fiber Channel adapters. All of these drives are configured as JBOD and are partitioned identically and hold the static database tables. See Table 4-2, Table 4-3, and Table 4-4

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 9.7. 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.

FC Adapter Mappings	
Logical Adapter ID	Physical Adapter ID
FC1	21:00:00:24:FF:24:01:8C
FC2	21:00:00:24:FF:24:01:8D
FC3	21:00:00:1B:32:0A:50:7F
FC4	21:00:00:24:FF:48:93:BE
FC5	21:00:00:24:FF:48:93:BF
FC6	21:00:00:24:FF:42:86:84
FC7	21:00:00:24:FF:42:86:85

Table 4-2: IBM System x 3650 M4 with KVM Virtualization FC Adapter Mapping

FC Adapter to Block Device and UA Mapping	
Logical Adapter ID	Block Device ID and database UAs
FC1	/dev/disk/by-id/scsi-360080e500024d43e00000667506ea73a ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 1
	/dev/disk/by-id/scsi-360080e500024d43e0000066a506ea76e ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 10
	/dev/disk/by-id/scsi-360080e500024d43e0000066d506ea7a7 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 19
	/dev/disk/by-id/scsi-360080e500024d43e00000670506ea7ce ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 28
	/dev/disk/by-id/scsi-360080e500024e1320000044b506eb9e8 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 4
	/dev/disk/by-id/scsi-360080e500024e1320000044d506eba16 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 13
	/dev/disk/by-id/scsi-360080e500024e1320000044f506eba3e ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 22
	/dev/disk/by-id/scsi-360080e500024e13200000451506eba76 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 31
	/dev/disk/by-id/scsi-360080e50002d045600000411506ecd8b ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 7
	/dev/disk/by-id/scsi-360080e50002d045600000418506eccb8 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 16
	/dev/disk/by-id/scsi-360080e50002d04560000041b506ecdec ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 25
	/dev/disk/by-id/scsi-360080e50002d04560000041e506ece19 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 34
	/dev/disk/by-id/scsi-360080e5000248e9a00000376506ef474 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 2
	/dev/disk/by-id/scsi-360080e5000248e9a00000379506ef499 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 11
/dev/disk/by-id/scsi-360080e5000248e9a0000037c506ef4bc ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 20	
/dev/disk/by-id/scsi-360080e5000248e9a0000037f506ef4da ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 29	
/dev/disk/by-id/scsi-360080e500024f70200000c32506ed2da ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 5	
/dev/disk/by-id/scsi-360080e500024f70200000c3506ed2fc ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 14	
/dev/disk/by-id/scsi-360080e500024f70200000c38506ed32d ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 23	
/dev/disk/by-id/scsi-360080e500024f70200000c3b506ed355 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 32	
/dev/disk/by-id/scsi-360080e500024d455400000a17506ed5af ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 8	
/dev/disk/by-id/scsi-360080e500024d455400000a1a506ed5d1 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 17	
/dev/disk/by-id/scsi-360080e500024d455400000a1d506ed5fb ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 26	
/dev/disk/by-id/scsi-360080e500024d455400000a20506ed622 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 35	
/dev/disk/by-id/scsi-360080e5000248b9000000d80506edc97 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 3	
/dev/disk/by-id/scsi-360080e5000248b9000000d86506edcfa4 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 12	
/dev/disk/by-id/scsi-360080e5000248b9000000d83506edcc5 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 21	
/dev/disk/by-id/scsi-360080e5000248b9000000d89506edd22 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 30	
/dev/disk/by-id/scsi-360080e5000247aac00001aaf51111288 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 6	
/dev/disk/by-id/scsi-360080e5000247aac00000ba95072a72b ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 15	
/dev/disk/by-id/scsi-360080e5000247aac00000bac5072a74d ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 24	
/dev/disk/by-id/scsi-360080e5000247aac00000baf5072a778 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 33	
/dev/disk/by-id/scsi-360080e500024dc4a00000d6e506eeb4e ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 9	
/dev/disk/by-id/scsi-360080e500024dc4a00000d71506eeb78 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 18	
/dev/disk/by-id/scsi-360080e500024dc4a00000d74506eeb9f ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 27	
/dev/disk/by-id/scsi-360080e500024dc4a00000d77506eebc8 ts_W, ts_D, ts_C, ts_H, ts_O, ts_OL, ts_N, is_O, is_C for UA 36	

FC4	/dev/disk/by-id/scsi-360080e500024d72c0000135250f94a8b	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 37	
	/dev/disk/by-id/scsi-360080e500024d72c000005ed506ea9f4	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 46	
	/dev/disk/by-id/scsi-360080e500024d72c000005ef506eaa1f	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 55	
	/dev/disk/by-id/scsi-360080e500024d72c000005f1506eaa6d	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 64	
	/dev/disk/by-id/scsi-360080e500024de5800000419506eacc2	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 40	
	/dev/disk/by-id/scsi-360080e500024de5800000416506eac98	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 49	
	/dev/disk/by-id/scsi-360080e500024de580000041c506eace7	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 58	
	/dev/disk/by-id/scsi-360080e500024de580000041f506ead11	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 67	
	/dev/disk/by-id/scsi-360080e500024a77000000402506edc4a	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 43	
	/dev/disk/by-id/scsi-360080e500024a77000000400506edc21	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 52	
FC5	/dev/disk/by-id/scsi-360080e500024a77000000404506edc74	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 61	
	/dev/disk/by-id/scsi-360080e500024a77000000406506edc9d	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 70	
	/dev/disk/by-id/scsi-360080e500024a4600000030e506ee6b9	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 38	
	/dev/disk/by-id/scsi-360080e500024a46000000310506ee6da	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 47	
	/dev/disk/by-id/scsi-360080e500024a46000000312506ee6f8	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 56	
	/dev/disk/by-id/scsi-360080e500024a46000000314506ee71b	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 65	
	/dev/disk/by-id/scsi-360080e50002d011200000b8a506ed3cf	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 41	
	/dev/disk/by-id/scsi-360080e50002d011200000b8c506ed3f1	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 50	
	/dev/disk/by-id/scsi-360080e50002d011200000b8e506ed414	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 59	
	/dev/disk/by-id/scsi-360080e50002d011200000b90506ed435	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 68	
FC6	/dev/disk/by-id/scsi-360080e500024d545400000a29506ed6b7	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 44	
	/dev/disk/by-id/scsi-360080e50002d040c000009bc506ed68b	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 53	
	/dev/disk/by-id/scsi-360080e50002d040c000009be506ed6c1	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 62	
	/dev/disk/by-id/scsi-360080e50002d040c000009c0506ed6e8	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 71	
	/dev/disk/by-id/scsi-360080e500024d3f000000d2d506ee986	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 39	
	/dev/disk/by-id/scsi-360080e500024d3f000000d2f506ee9a8	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 48	
	/dev/disk/by-id/scsi-360080e500024d3f000000d31506ee9c8	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 57	
	/dev/disk/by-id/scsi-360080e500024d3f000000d33506ee9ee	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 66	
	/dev/disk/by-id/scsi-360080e500024d60a00000b635072aa08	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 42	
	/dev/disk/by-id/scsi-360080e500024d60a00000b655072aa2f	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 51	
FC7	/dev/disk/by-id/scsi-360080e500024d60a00000b675072aa4f	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 60	
	/dev/disk/by-id/scsi-360080e500024d60a00000b695072aa78	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 69	
	/dev/disk/by-id/scsi-360080e500024dd1400000cef506edf0c	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 45	
	/dev/disk/by-id/scsi-360080e500024dd1400000cef1506edf2c	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 54	
	/dev/disk/by-id/scsi-360080e500024dd1400000cef3506edf4f	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 63	
	/dev/disk/by-id/scsi-360080e500024dd1400000cef5506edf72	ts_W	ts_D	ts_C	ts_H	ts_O	ts_OL	ts_N	is_O	is_C	for UA 72	
	/dev/disk/by-id/scsi-360080e500023620800005d5506c0368	for log										

Table 4-3: IBM System x 3650 M4 with KVM Virtualization Disk Mapping

SSD Partition Number to Tablespace Mapping	
Device Partition Number	Tablespace ID
1	ts_W
2	ts_D
3	ts_I
4	<extended partition>
5	ts_S
6	ts_C
7	ts_H
8	ts_O
9	ts_OL
10	ts_N
11	is_O
12	is_C

Table 4-4: IBM System x 3650 M4 with KVM Virtualization SSD Partition Mapping

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	104,040
Measured TpmC	1,320,082

Table	Rows	Table	Index	5% Space	Total Space
Warehouse	104,040	54	0	3	57
District	1,040,400	198	0	10	208
Item	100,000	10	0	1	11
Stock	10,404,000,000	3,387,024	0	169,351	3,556,375
Customer	3,121,200,000	2,438,784	156,240	129,751	2,724,775
New-Order	936,360,000	38,664	0	1,933	40,597
Orders	3,121,200,000	120,044	87,840	0	207,884
Order-Line	31,212,000,000	2,093,082	0	0	2,093,082
History	3,121,200,000	193,248	0	0	193,248
Additional Overhead		1,927,738			1,927,738

Free Space	588,882		
Dynamic Space	2,406,374	<u>30 Minute log Computations</u>	
Static Space	8,337,601	Log Written (KB)	89,105,535
Daily Growth	488,522	New-Order Txns	39,602,460
Daily Spread	0	Log Written per New-Order (KB)	2.25

Data Storage Requirement

60 Days (MB)	37,648,893
60 Days (GB)	36,766

Log Storage Requirement

8 Hours (GB)	1,359.64
--------------	----------

Data + Log

38,126.139

Disk Sizing

Disk Type	Formatted		SUT		Priced	
	Capacity (GB)	# of Disks	Capacity (GB)	# of Disks	Capacity (GB)	# of Disks
LOG 450GB SAS HDD RAID10	418.69	12	2,512	12	2,512	12
DB 200GB SAS SSD JBOD	185.80	72	13,378	72	13,378	72
BACKUP 1TB SAS HDD JBOD	931.01	27	25,137	27	25,137	27
OS 146GB SAS HDD RAID1	135.97	4	271.94	4	271.94	4

Total Capacity

41,299

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 5-2 lists the response times and the ninetieth percentiles for each of the transaction types for the measured system.

Table 5-1: Response Times

<u>Response Times (in seconds)</u>	<u>90th %</u>	<u>Average</u>	<u>Maximum</u>
New Order	0.160	0.124	1.875
Payment	0.160	0.121	3.078
Order-Status	0.160	0.122	0.953
Delivery (interactive)	0.160	0.117	0.703
Delivery (deferred)	0.030	0.020	1.01
Stock-Level	0.160	0.123	1.110
Menu	0.141	0.117	1.046

Response time delay added for emulated components was 0.1 seconds

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 5-22 lists the TPC-C keying and think times for the measured system.

Table 5-2: Think and Keying Times

<u>Keying/Think Times (in seconds)</u>	<u>Min.</u>	<u>Average</u>	<u>Max.</u>
New Order	18.000/0.00	18.000/12.035	18.062/120.343
Payment	3.000/0.00	3.000/12.037	3.062/120.344
Order-Status	2.000/0.00	2.000/10.033	2.047/100.329
Delivery	2.000/0.00	2.000/5.036	2.047/50.328
Stock-Level	2.000/0.00	2.000/5.035	2.047/50.328

5.3. Response Time Frequency Distribution

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

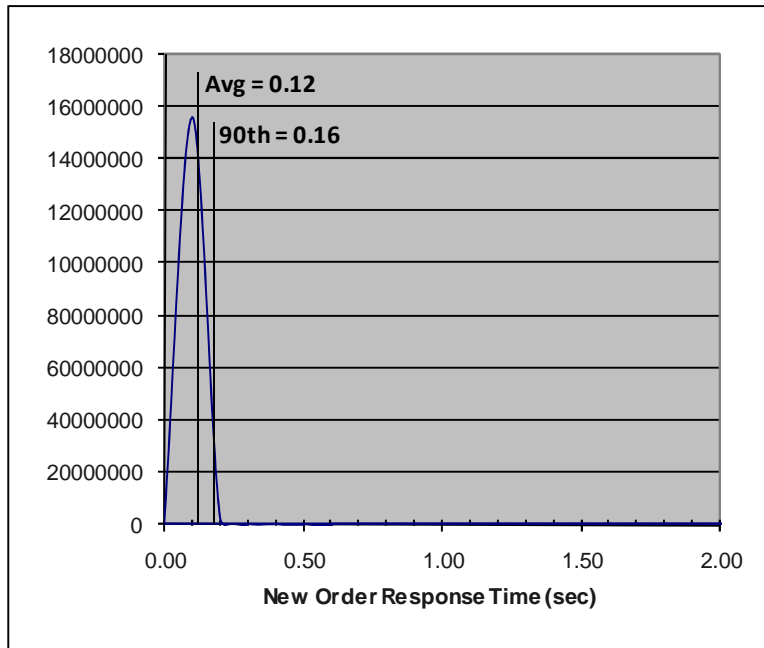


Figure 5-3: New-Order Response Time Distribution

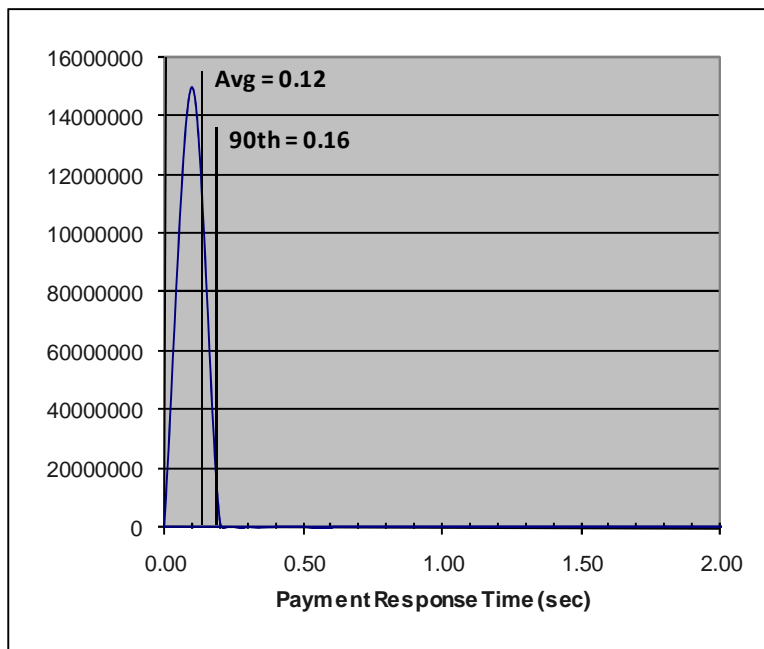


Figure 5-4: Payment Response Time Distribution

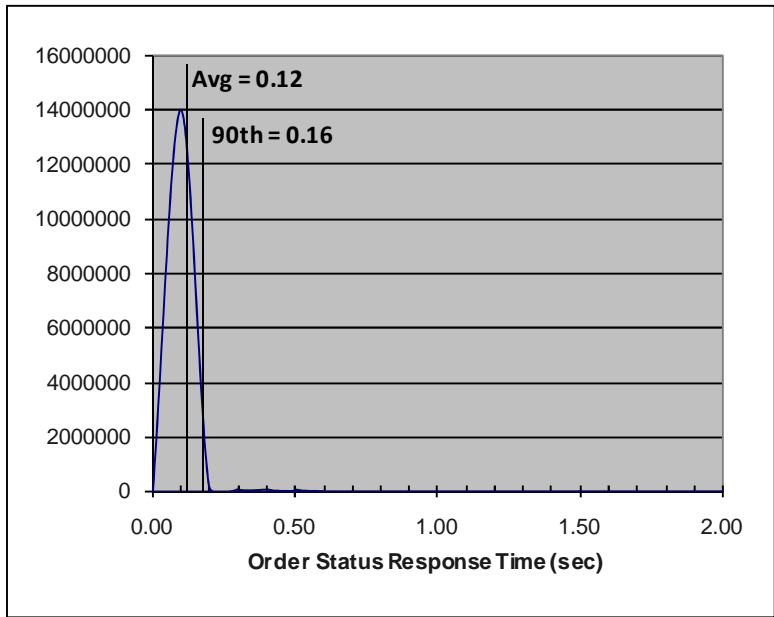


Figure 5-5: Order-Status Response Time Distribution

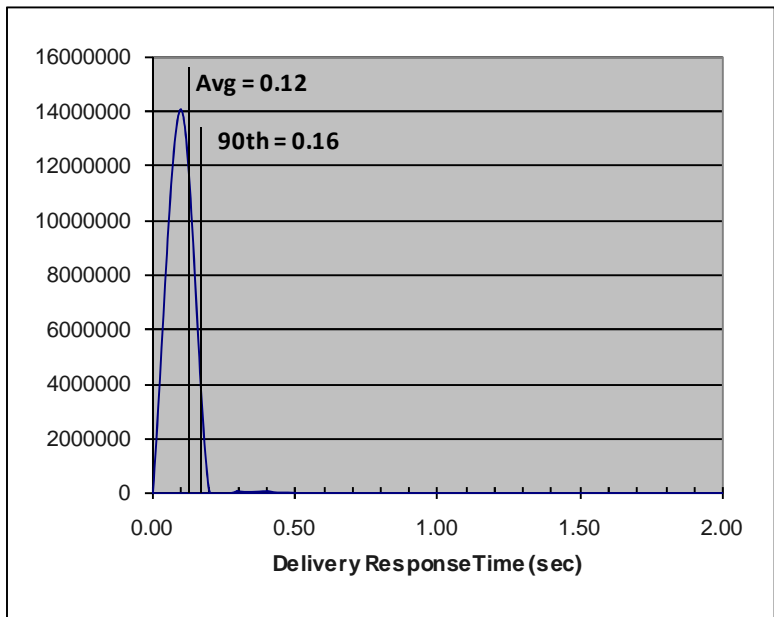


Figure 5-6: Delivery (Interactive) Response Time Distribution

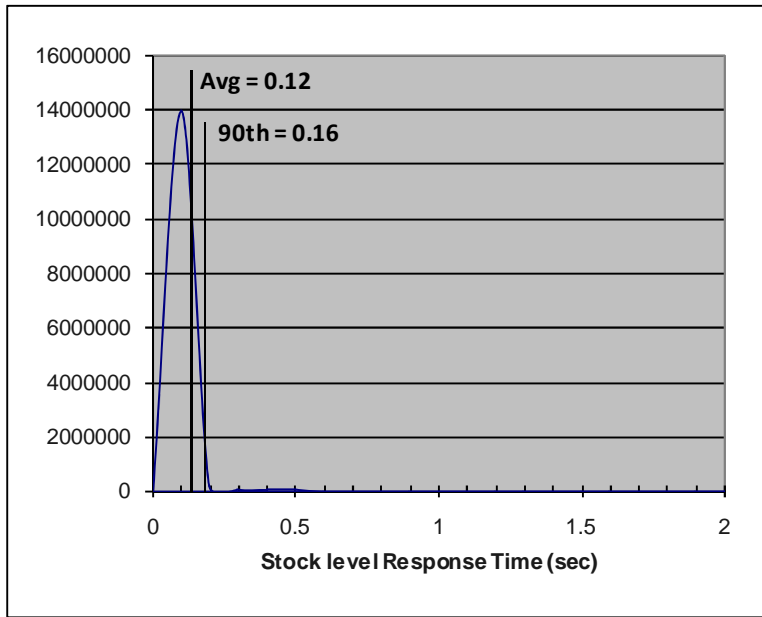


Figure 5-7: Stock Level Response Time Distribution

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.

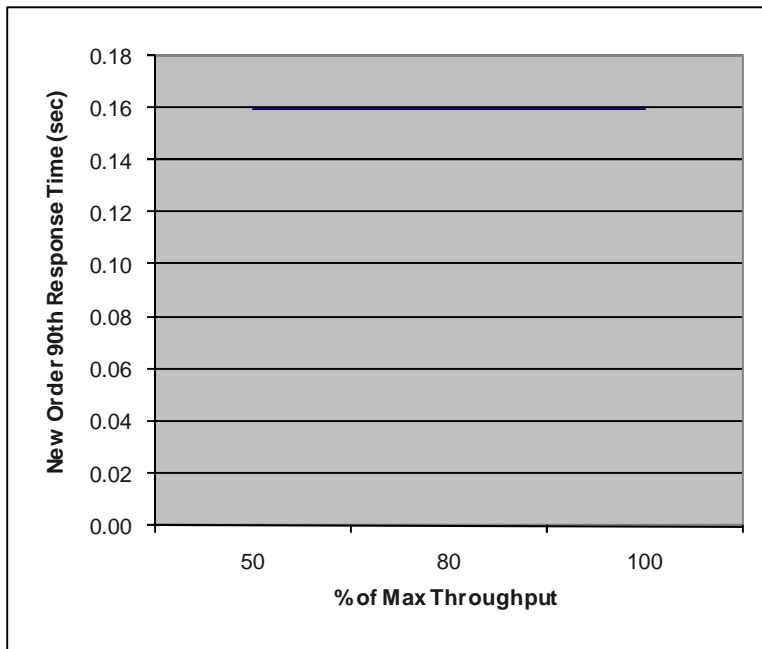


Figure 5-8: New-Order Response Time vs. Throughput

5.5. Think Time Frequency Distribution

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

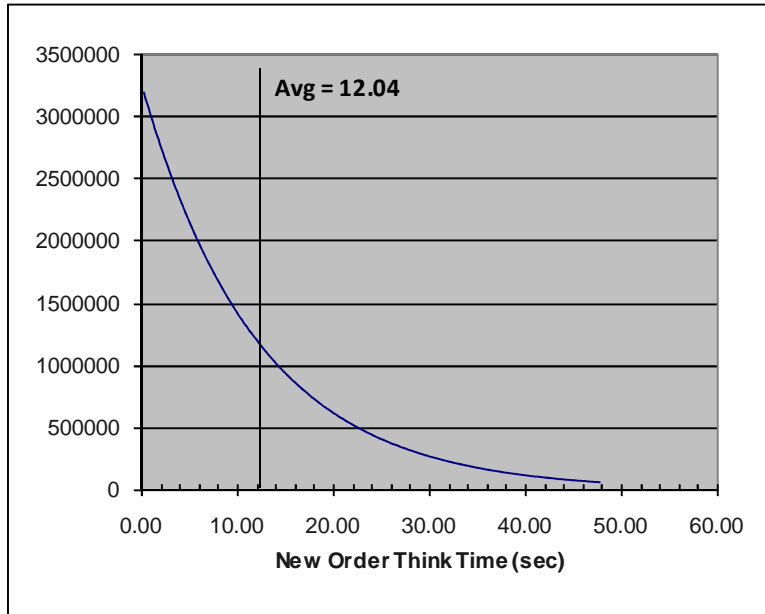


Figure 5-9: New-Order Think Time Distribution

5.6. Throughput versus Elapsed Time

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

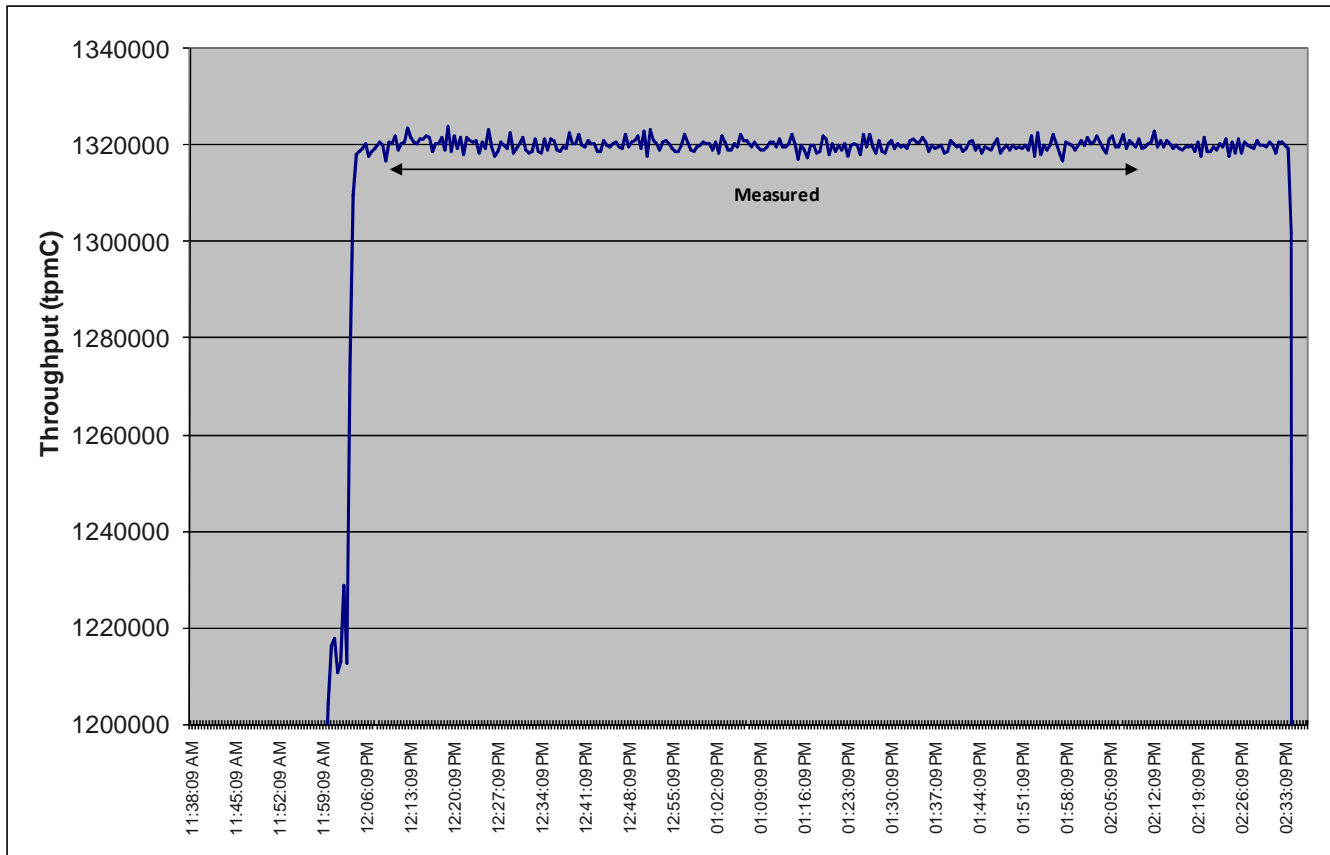


Figure 5-10: New-Order Throughput vs. Elapsed Time

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- 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 6.0 (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 6.0. 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 2003 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:

- o 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.
- o IIS accepts the filled in GET request, parses, and validates all values entered by the user.
- o It then proceeds to transmit those values to the COM+ server through an transaction type specific COM+ api interface.
- o The COM+ Pool Manager receives the request and first decides if there is a connection object in the pool available to service it.
- o If so, the connection is used to send the transaction request to the Server.
- o If no connection is available, the request will enter a COM+ internal queue and will be serviced by the next available connection.
- o 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).
- o The transaction is committed and the DB2 back end client returns control back to the COM pool thread.
- o COM pool thread returns control to the ISAPI caller.
- o (All transaction results are inside the data structure that the ISAPI caller provided to the COM+ api in the parameter list).
- o 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://](http://portal.acm.org/citation.cfm?id=128770&coll=portal&dl=ACM&CFID=10343790&CFTOKEN=42047146)

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. A total of 104,040 warehouses were configured; 104,040 were accessed during the runs. A rampup time of 41 minutes and 52 seconds was specified, along with a run time of two hours.

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 US list prices.

Discounts are based on US list prices and for similar quantities and configurations. A discount of 50.31% has been applied to specified IBM hardware, software, 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 February 25, 2013.

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 x3650 M4 with KVM	1,320,082	\$667,882 USD	\$0.51 USD	February 25, 2013

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:



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

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

February 18, 2013

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

Platform: **IBM System x3650 M4 c/s**
Operating system: **RHEL 6.4 with KVM**
Database Manager: **DB2 9.7 Enterprise Edition**
Transaction Manager: **Microsoft COM+**

The results were:

CPU's Speed	Memory	Disks	New Order 90% Response Time	tpmC
Server: IBM System x3650 M4				
2 x Intel Xeon E5-2690 (2.90 GHz)	768 GB (20 MB L3)	27 x 1 TB 7200 rpm SAS 12 x 450 GB 15K rpm SAS 72 x 200 GB SAS 2.5" SSD 4 x 146 GB 10K rpm SAS	0.160 Second	1,320,082
4 Clients: IBM System x3550 M2 (each with)				
2 x Intel Xeon E5620 (2.90 GHz)	4 GB (12 MB L3)	1 x 146 GB 10K rpm SAS	n/a	n/a

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

The following verification items were given special attention:

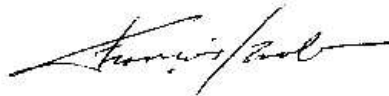
- The transactions were correctly implemented
- The database records were the proper size
- The database was properly scaled and populated

- The ACID properties were met
- Input data was generated according to the specified percentages
- The transaction cycle times included the required keying and think times
- The reported response times were correctly measured.
- At least 90% of all delivery transactions met the 80 Second completion time limit
- All 90% response times were under the specified maximums
- The measurement interval was representative of steady state conditions
- The reported measurement interval was 120 minutes
- 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:

For availability reasons the Intel Xeon E5530 processors configured in the clients during testing were substituted one for one with Intel Xeon E5620 in the priced configuration. Based on the characteristics of the two processors it is my opinion that this substitution would have no significant effect on performance.

Respectfully Yours,



François Raab, President

9. Appendix A: Client Server Code

9.1. Client/Terminal Handler Code

9.1.1. Makefile.config

```
#####
###
### Licensed Materials - Property of IBM
###
### (C) COPYRIGHT International Business Machines Corp. 1996, 2010
### 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
LD_FLAGS_EXEC=
LD_FLAGS_SHLIB=/DLL
LD_FLAGS_STORP=$(LD_FLAGS_SHLIB) /DEF:rpctpc.def
LD_FLAGS_LIB=/LIBPATH:$(TPCC_SQLLIB)\lib /LIBPATH:"C:\Program
Files\Microsoft Visual Studio\VC98\Lib" db2api.lib winmm.lib
LD_FLAGS_OUT=/OUT:

# Library Configuration
AR=lib.exe
AR_FLAGS=
AR_FLAGS_LIB=
AR_FLAGS_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=&
```

9.1.2. Src.Cli/Makefile

```
#####
###
### Licensed Materials - Property of IBM
###
### (C) COPYRIGHT International Business Machines Corp. 1996, 2010
### 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
# #####

PRP_OPTS =          PACKAGE \
                   ISOLATION RR \
                   QUERYOPT 7 \
                   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$(TPCC_SPTYPE)

OBJ = $(TPCC_ROOT)/Src.Common/tpccdbg$(OBJEXT) \
      $(TPCC_ROOT)/Src.Common/tpccctx$(OBJEXT) \
      tpcccli$(OBJEXT)

LIBS =              tpcccli$(LIBEXT)

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

all:                connect $(OBJ) $(LIBS) disconnect
                   $(AR) $(AR_FLAGS) $(AR_FLAGS_OUT)tpcccli$(LIBEXT) $(OBJ)
$(AR_FLAGS_LIB)
                   @echo "-----"
                   @echo "Please copy lval.h, db2tpcc.h, and tpcccli$(LIBEXT) to"
                   @echo "a place where they can be #included and linked with the"
                   @echo "RTE/driver code."
                   @echo "-----"

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

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

connect:            - db2 connect to $(TPCC_DBNAME)

disconnect:        - db2 connect reset
                   - db2 terminate

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

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

tpcccli.c:
                   @echo "Prepping $*.sqc"
                   db2 prep $*.sqc $(PRP_OPTS)
                   db2 grant execute on package TPCCCLI to public

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

# Client Library:
tpcccli$(LIBEXT): $(OBJ)

# Source
tpcccli$(OBJEXT): tpcccli.c

# Headers
tpcccli.c:         $(TPCC_ROOT)/include/db2tpcc.h
$(TPCC_ROOT)/include/lval.h
```

9.1.3. Src.Cli/tpcccli.sqc

```
/*
****
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1996, 2010
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
****
*/

/*
 * tpcccli.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 ;
```

```

// Primary comparison key:  I_ID
// Secondary comparison key: W_ID

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 ;

    // Determine if order is "all-local" or not
    // NOTE: This loop will exit on the iteration *after* finding the last
    // item; this effectively takes care of the 0-based/1-based conversion
    // and we don't have to add one when assigning to s_O_OL_CNT below.
    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 ;

    // Sort the item list.  Since invalid item IDs = 100001, we will remain
    // compliant with the spec (Section 2.4.2.3 Comment 1.

    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_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_all_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 ;

// Austin RTE Integration Fix
// Austin's Screen application doesn't fill in C_W_ID and C_D_ID unless
// it's a remote payment transaction. Since we expect these to be
// filled
// in for all cases, we need to fill them in if they are not already.
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
                               , :c_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,
, D_STREET_1, D_STREET_2, D_CITY,
, C_LAST, C_FIRST, C_MIDDLE, C_STREET_1,
, C_CITY, C_STATE, C_ZIP, C_PHONE,
, 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 )
{
// Propogate the field we already knew into the output structure
// 60% of the time, we already knew 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

```

```

#ifdef 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 ) ;
}

```

9.1.4. Src.Common/Makefile

```

#####
###
## Licensed Materials - Property of IBM
##
## (C) COPYRIGHT International Business Machines Corp. 1996, 2010
## 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
# #####

PRP_OPTS =          PACKAGE \
                   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$(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

dbggen:            $(UTIL_OBJ_GEN)

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

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

connect:           - db2 connect to $(TPCC_DBNAME)

disconnect:        - db2 connect reset
                   - db2 terminate

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

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

.sqc.c:
    @echo "Prepping $*.sqc"
    db2 prep $*.sqc $(PRP_OPTS)
    db2 grant execute on package TPCCCTX to public

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

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

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

```

9.1.5. Src.Common/tpccctx.sqc

```
*****
****
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1996, 2010
** 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 create_context();
int destroy_context();
int attach_context(void*);
int detach_context(void*);
int get_context(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;

    SQLCODE = create_context();
    if (SQLCODE != 0) { return SQLCODE; }

    /* 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);

        SQLCODE = destroy_context();
        if (SQLCODE != 0) { return SQLCODE; }

        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);
    }

    SQLCODE = destroy_context();
    if (SQLCODE != 0) { return SQLCODE; }

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

int create_context(void)
{
```

```
SQL_STRUCTURE sqlca sqlca;
void *ctx;

sqlcSetTypeCtx(SQL_CTX_MULTI_MANUAL);
sqlcBeginCtx(&ctx, SQL_CTX_BEGIN_ALL, NULL, &sqlca);

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

return 0;
}

int attach_context(void *ctx)
{
    SQL_STRUCTURE sqlca sqlca;

    sqlcAttachToCtx(ctx, NULL, &sqlca);

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

    return 0;
}

int detach_context(void *ctx)
{
    SQL_STRUCTURE sqlca sqlca;

    sqlcDetachFromCtx(ctx, NULL, &sqlca);

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

    return 0;
}

int destroy_context(void)
{
    SQL_STRUCTURE sqlca sqlca;
    void *ctx;

    SQLCODE = get_context(&ctx);
    if (SQLCODE) { return SQLCODE; }

    sqlcEndCtx(&ctx, SQL_CTX_END_ALL, NULL, &sqlca);

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

    return 0;
}

int get_context(void **ctx)
{
    SQL_STRUCTURE sqlca sqlca;

    sqlcGetCurrentCtx(ctx, NULL, &sqlca);

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

    return 0;
}
```

9.1.6. Src.Common/tpccdbg.c

```
*****
****
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1996, 2010
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****
****/

/*
 * tpccdbg.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 */
/*-----*/
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, "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 (!isprint(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, "\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");

```



```

        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);
}

```

9.1.7. include/db2tpcc.h

```

/*****
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1996, 2010
** 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>
typedef __int16 int16_t;
typedef __int32 int32_t;
typedef __int64 int64_t;

#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

/* *****
**/
/* Whenever changing these structures, you MUST update the char[] array
**/
/* size in the varchar hostvars, the xx->len values in Src.Cli/???c.sqc
**/
/* and the utils/cat.ddl and utils/uncat.ddl scripts.
**/
**/
/* In all cases, the proper value to use is:
**/
**/
/* sizeof(struct xxx) - SPGENERAL_ADJUST
**/
**/
/* SPGENERAL_PAD is the number of int16_ts required to pad the structures
**/
/* so that the data is aligned the same as if the len/pad elements were
**/
/* not present.
**/
**/
/* NOTE: It is almost guaranteed that this will break when using clients
**/
/* that use a different byte ordering and/or compiler padding.
**/
**/
**/
#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);

```

```

extern int create_context(void);
extern int destroy_context(void);
extern int get_context(void**);
extern int attach_context(void*);
extern int detach_context(void*);

```

```

#ifdef __cplusplus
}
#endif

```

```

#endif // __DB2TPCC_H

```

9.1.8. include/tpccdbg.h

```

/* *****
****
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1996, 2010
** 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

```

9.1.9. tpccenv.bat

```

@REM
#####
@REM Licensed Materials - Property of IBM
@REM
@REM (C) COPYRIGHT International Business Machines Corp. 1996, 2010
@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
@REM tpccenv.bat - Windows Environment Setup
@REM

@REM The Kit Version
set TPCC_VERSION=CK101019

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

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

@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 DARIVERSION or NONDARI;
@REM set TPCC_SPTYPE=NOSP
@REM set TPCC_SPTYPE=SPGENERAL2
set TPCC_SPTYPE=SPGENERAL
@REM set TPCC_SPTYPE=DARI2SQLDA

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

@REM DB2 EE/EEE Configuration
set DB2EDITION=EE
@REM set DB2EDITION=DPF

@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\tpcc_db2_12m
set TPCC_DBNAME=TPCC
set TPCC_ROOT=c:\home\tpcc\tpcc_db2_12m\tpc-c.ibm
set TPCC_SQLLIB=c:\sqllib
set TPCC_RUNDATA=%HOME%\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. Client Transaction Code

9.2.1. Makefile.config

```

#####
###
### Licensed Materials - Property of IBM

```

```

###
### (C) COPYRIGHT International Business Machines Corp. 1996, 2010
### 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;

```

9.2.2. Src.Common/Makefile

```

#####
###
### Licensed Materials - Property of IBM
###
### (C) COPYRIGHT International Business Machines Corp. 1996, 2010
### 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
#####

PRP_OPTS =          PACKAGE \
                   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$(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 tpcctx.c

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

connect:
    - db2 connect to $(TPCC_DBNAME)

disconnect:
    - db2 connect reset
    - db2 terminate

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

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

.sqc.c:
    @echo "Prepping $*.sqc"
    db2 prep $*.sqc $(PRP_OPTS)
    db2 grant execute on package TPCCCTX to public

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

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

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

```

9.2.3. Src.Common/tpccdbg.c

```

/*****
****
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1996, 2010
** 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 */
/*-----*/
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, "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 (!isprint(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, "\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, "\t\t%s_ID[%d]          = %d\n",
                j, delivery_ptr->s_O_ID[j]);
    }
    fprintf(debug_fp, "\t}\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, "\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, "\t\t%s_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, "\t\t%s_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, "\t\t%s_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");

    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");
    items = neword_ptr->s_O_OL_CNT;
    for (j=0; j<items; j++) {
        if(j != 0)
            fprintf(debug_fp, "\n");
        fprintf(debug_fp, "\t\t%s_I_NAME[%d]          = %s\n",
                j, neword_ptr->item[j].s_I_NAME);
        fprintf(debug_fp, "\t\t%s_I_PRICE[%d]         = %.2f \n",
                j, neword_ptr->item[j].s_I_PRICE);
        fprintf(debug_fp, "\t\t%s_OL_AMOUNT[%d]      = %.2f \n",
                j, neword_ptr->item[j].s_OL_AMOUNT);
        fprintf(debug_fp, "\t\t%s_S_QUANTITY[%d]     = %d (%X)\n",
                j, neword_ptr->item[j].s_S_QUANTITY, neword_ptr-
>item[j].s_S_QUANTITY);
        fprintf(debug_fp, "\t\t%s_brand_generic[%d] = %c\n",
                j, neword_ptr->item[j].s_brand_generic);
    }
}

```

```
}
fprintf(debug_fp, "\t\n\n\n");
fclose(debug_fp);
}

/*-----*/
/* ord_debug */
/*-----*/
void ord_debug (struct out_ordstat_struct *ordstat_ptr,
                struct in_ordstat_struct *in_ordstat,
                char *msg)
{
    char debug_fn[DEBUG_PATH_SIZE + DEBUG_FILENAME_SZ];

    InitializeDebug();
    strncpy(debug_fn, debugPath, DEBUG_PATH_SIZE);
    strcat(debug_fn, "ord.debug.out");
    ord_print(ordstat_ptr, in_ordstat, debug_fn, msg);
}

/*-----*/
/* ord_print */
/*-----*/
void ord_print (struct out_ordstat_struct *ordstat_ptr,
                struct in_ordstat_struct *in_ordstat,
                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, "Order status debug information follows %s (%s)\n",
            timeStamp, msg);

    fprintf(debug_fp, "\n=====\n");

    fprintf(debug_fp, "in_ordstat_struct {\n");
    fprintf(debug_fp, "\ts_W_ID           = %d (%X)\n",
            in_ordstat->s_W_ID, in_ordstat->s_W_ID);
    fprintf(debug_fp, "\ts_D_ID           = %d (%X)\n",
            in_ordstat->s_D_ID, in_ordstat->s_D_ID);
    fprintf(debug_fp, "\ts_C_ID           = %d (%X)\n",
            in_ordstat->s_C_ID, in_ordstat->s_C_ID);
    fprintf(debug_fp, "\ts_C_LAST        = %s\n",
            in_ordstat->s_C_LAST);
    fprintf(debug_fp, "\n");

    fprintf(debug_fp, "out_ordstat_struct {\n");
    fprintf(debug_fp, "\ts_C_ID           = %d (%X)\n",
            ordstat_ptr->s_C_ID, ordstat_ptr->s_C_ID);
    fprintf(debug_fp, "\ts_C_FIRST        = %s\n",
            ordstat_ptr->s_C_FIRST);
    fprintf(debug_fp, "\ts_C_MIDDLE       = %s\n",
            ordstat_ptr->s_C_MIDDLE);
    fprintf(debug_fp, "\ts_C_LAST        = %s\n",
            ordstat_ptr->s_C_LAST);
    fprintf(debug_fp, "\ts_C_BALANCE     = %.2f\n",
            ordstat_ptr->s_C_BALANCE);
    fprintf(debug_fp, "\ts_O_ID           = %d (%X)\n",
            ordstat_ptr->s_O_ID, ordstat_ptr->s_O_ID);
    fprintf(debug_fp, "\ts_O_ENTRY_D      = %s\n",
            ordstat_ptr->s_O_ENTRY_D_time);
    fprintf(debug_fp, "\ts_O_CARRIER_ID = %d (%X)\n",
            ordstat_ptr->s_O_CARRIER_ID, ordstat_ptr->s_O_CARRIER_ID);
    fprintf(debug_fp, "\ts_ol_cnt         = %d (%X)\n",
            ordstat_ptr->s_ol_cnt, ordstat_ptr->s_ol_cnt);
    fprintf(debug_fp, "\ts_transtatus     = %d (%X)\n",
            ordstat_ptr->s_transtatus, ordstat_ptr->s_transtatus);
    fprintf(debug_fp, "\tdeadlocks       = %d (%X)\n",
            ordstat_ptr->deadlocks, ordstat_ptr->deadlocks);

    fprintf(debug_fp, "\titems {\n");
    items = ordstat_ptr->s_ol_cnt;
    for (j = 0; j < items; j++) {
        if (j != 0)
            fprintf(debug_fp, "\n");
        fprintf(debug_fp, "\ts_OL_SUPPLY_W_ID[%d] = %d (%X)\n",
                j, ordstat_ptr->item[j].s_OL_SUPPLY_W_ID, ordstat_ptr->item[j].s_OL_SUPPLY_W_ID);
        fprintf(debug_fp, "\ts_OL_I_ID[%d]     = %d (%X)\n",
                j, ordstat_ptr->item[j].s_OL_I_ID, ordstat_ptr->item[j].s_OL_I_ID);
        fprintf(debug_fp, "\ts_OL_QUANTITY[%d] = %d (%X)\n",
                j, ordstat_ptr->item[j].s_OL_QUANTITY, ordstat_ptr->item[j].s_OL_QUANTITY);
        fprintf(debug_fp, "\ts_OL_AMOUNT[%d]   = %.2f\n",
                j, ordstat_ptr->item[j].s_OL_AMOUNT);
        fprintf(debug_fp, "\ts_OL_DELIVERY_D[%d] = %s\n",
                j, ordstat_ptr->item[j].s_OL_DELIVERY_D_time);
    }
    fprintf(debug_fp, "\t}\n\n\n");
}
```

```
fclose(debug_fp);
}

/*-----*/
/* pay_debug */
/*-----*/
void pay_debug (struct out_payment_struct *payment_ptr,
                struct in_payment_struct *in_payment,
                char *msg)
{
    char debug_fn[DEBUG_PATH_SIZE + DEBUG_FILENAME_SZ];

    InitializeDebug();
    strncpy(debug_fn, debugPath, DEBUG_PATH_SIZE);
    strcat(debug_fn, "pay.debug.out");
    pay_print(payment_ptr, in_payment, debug_fn, msg);
}

/*-----*/
/* pay_print */
/*-----*/
void pay_print (struct out_payment_struct *payment_ptr,
                struct in_payment_struct *in_payment,
                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, "Payment debug information follows %s (%s)\n",
            timeStamp, msg);

    fprintf(debug_fp, "\n=====\n");

    fprintf(debug_fp, "in_payment_struct {\n");
    fprintf(debug_fp, "\ts_H_AMOUNT      = %.2f\n",
            in_payment->s_H_AMOUNT);
    fprintf(debug_fp, "\ts_C_ID          = %d (%X)\n",
            in_payment->s_C_ID, in_payment->s_C_ID);
    fprintf(debug_fp, "\ts_W_ID          = %d (%X)\n",
            in_payment->s_W_ID, in_payment->s_W_ID);
    fprintf(debug_fp, "\ts_D_ID          = %d (%X)\n",
            in_payment->s_D_ID, in_payment->s_D_ID);
    fprintf(debug_fp, "\ts_C_D_ID        = %d (%X)\n",
            in_payment->s_C_D_ID, in_payment->s_C_D_ID);
    fprintf(debug_fp, "\ts_C_W_ID        = %d (%X)\n",
            in_payment->s_C_W_ID, in_payment->s_C_W_ID);
    fprintf(debug_fp, "\ts_C_LAST        = %s\n",
            in_payment->s_C_LAST);
    fprintf(debug_fp, "\n");

    fprintf(debug_fp, "out_payment_struct {\n");
    fprintf(debug_fp, "\ts_C_CREDIT_LIM = %.2f\n",
            payment_ptr->s_C_CREDIT_LIM);
    fprintf(debug_fp, "\ts_C_DISCOUNT = %04.4f\n",
            payment_ptr->s_C_DISCOUNT);
    fprintf(debug_fp, "\ts_C_BALANCE   = %.2f\n",
            payment_ptr->s_C_BALANCE);
    fprintf(debug_fp, "\ts_C_ID        = %d (%X)\n",
            payment_ptr->s_C_ID, payment_ptr->s_C_ID);
    fprintf(debug_fp, "\ts_W_STREET_1 = %s\n",
            payment_ptr->s_W_STREET_1);
    fprintf(debug_fp, "\ts_W_STREET_2 = %s\n",
            payment_ptr->s_W_STREET_2);
    fprintf(debug_fp, "\ts_W_CITY     = %s\n",
            payment_ptr->s_W_CITY);
    fprintf(debug_fp, "\ts_W_STATE    = %s\n",
            payment_ptr->s_W_STATE);
    fprintf(debug_fp, "\ts_W_ZIP      = %s\n",
            payment_ptr->s_W_ZIP);
    fprintf(debug_fp, "\ts_D_STREET_1 = %s\n",
            payment_ptr->s_D_STREET_1);
    fprintf(debug_fp, "\ts_D_STREET_2 = %s\n",
            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");
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, "\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);
}

```

9.2.4. Src.Common/tpccmisc.c

```

/*****
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1996, 2010
** 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 <windows.h>

#define RAND_A 16807
#define RAND_M 2147483647
#define RAND_M1 2147483646
#define RAND_MD 2147483647.0
#define RAND_Q 127773
#define RAND_R 2836

static int seed = 1;
static int seedflag = 0;

void random(int);
int random(void);
double current_time_ms(void);
double current_time(void);

void srandom (int initial_seed)
{
    seed = initial_seed;
    if ((seed < 1) || (seed > RAND_M1)) seed = 1;
}

int random (void)
{
    int lo;
    int hi;
    int test;

    hi = seed / RAND_Q;
    lo = seed % RAND_Q;
    test = RAND_A * lo - RAND_R * hi;
    if (test > 0) seed = test;
    else seed = test + RAND_M;

    return (seed);
}

/* Current time in SECONDS, precision SECONDS */
double current_time(void)
{
    /* truncate fractional seconds -> seconds */
    return (double)((int)(current_time_ms()));
}

/* Current time in SECONDS, precision MILLISECONDS */
double current_time_ms(void)
{
    /* GetCurrentTime() returns ms */
    /* convert to fractional seconds */
    return (GetCurrentTime() / 1000);
}

```

9.2.5. Src.Srv/Makefile

```

#####
###
### Licensed Materials - Property of IBM
###
### (C) COPYRIGHT International Business Machines Corp. 1996, 2010
### 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
# #####

PRP_OPTS =          PACKAGE \
                   ISOLATION RR \
                   QUERYOPT 7 \
                   EXPLAIN ALL \
                   MESSAGES $*.prep.msg

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

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

LDFLAGS = $(LDFLAGS_STORP) $(LDFLAGS_LIB)

# #####
# File Collections
# #####

STORED_PROCS =      new ord del

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

```

```

$(TPCC_ROOT)/Src.Common/tpccdbg$(OBJEXT)
DLL =          rpctpc$(SHLIBEXT)
# #####
# User Targets
# #####
all:          connect explain catalog $(DLL) install disconnect
clean:        connect uncatalog unexplain disconnect
              - $(ERASE) $(TPCC_SPDIR)$(SLASH)rpctpc$(SHLIBEXT)
              - $(ERASE) *.bnd *.msg *.out *$(OBJEXT) $(DLL) tpcc_all_sql.c
# #####
# 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:      - db2 "call sysproc.sysinstallobjects('EXPLAIN','C',' ',CURRENT
SCHEMA)"
unexplain:    - db2 "call sysproc.sysinstallobjects('EXPLAIN','D',' ',CURRENT
SCHEMA)"
connect:      - db2 connect to $(TPCC_DBNAME)
disconnect:   - db2 connect reset
              - db2 terminate
# #####
# Install Targets
# #####
install:      $(DLL)
              - mkdir $(TPCC_SPDIR)
              $(COPY) $(DLL) $(TPCC_SPDIR)
# #####
# Build Rules
# #####
.SUFFIXES:   $(OBJEXT) .c .sql
tpcc_all_sql.c:
              @echo "Prepping *.sql"
              db2 prep *.sql $(PRP_OPTS)
              db2 grant execute on package TPCC_ALL to public
tpcc_all_sql$(OBJEXT):
              $(CC) -c tpcc_all_sql.c $(CFLAGS) -D$(TPCC_SPTYPE)
              $(CFLAGS_OUT)$(@)
$(DLL): $(UTIL_OBJ) tpcc_all_sql$(OBJEXT)
              $(LD_STORP) $(LDFLAGS) $(UTIL_OBJ) tpcc_all_sql$(OBJEXT)
              $(LDFLAGS_OUT)$(@)
# #####
# Dependencies
# #####
# Executables (Stored Procedures)
$(DLL): $(UTIL_OBJ) tpcc_all_sql$(OBJEXT)
# Source
tpcc_all_sql$(OBJEXT): tpcc_all_sql.c
# Headers
tpcc_all_sql.c: $(TPCC_ROOT)/include/db2tpcc.h

```

9.2.6. Src.Srv/cat-func.ddl

```

-----
--
-- Licensed Materials - Property of IBM
--
-- (C) COPYRIGHT International Business Machines Corp. 1996, 2010
-- 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 REAL
, 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 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_LAST.H_AMOUNT
        WHERE D_W_ID = PAY_C_LAST.W_ID
        AND D_ID = PAY_C_LAST.D_ID
    ) AS U
) ;

/* @twb01c: Reverted query text to version from July 2011 (3M) publish.
*/
/* 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 )
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
) ;

/* Insert into history */

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 customer matching C_ID */
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
    ) ;

/* Insert into history */
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 ) 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_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
%

```

9.2.7. Src.Srv/tpcc_all_sql.sqc

```

/*****
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1996, 2010
** 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_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 )
                                , ( 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

                    )

                    ) 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

                    )

                    ) AS X ( OL_NUMBER , 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

        )

        ) AS X ( OL_NUMBER , 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

```

```

, 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

```

```

, :id0 , :ol_quantity0 , :supply_w_id0 )
, :id1 , :ol_quantity1 , :supply_w_id1 )
, :id2 , :ol_quantity2 , :supply_w_id2 )
, :id3 , :ol_quantity3 , :supply_w_id3 )
, :id4 , :ol_quantity4 , :supply_w_id4 )
, :id5 , :ol_quantity5 , :supply_w_id5 )
, :id6 , :ol_quantity6 , :supply_w_id6 )
, :id7 , :ol_quantity7 , :supply_w_id7 )
, :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 )
, :id1 , :ol_quantity1 , :supply_w_id1 )
, :id2 , :ol_quantity2 , :supply_w_id2 )
, :id3 , :ol_quantity3 , :supply_w_id3 )
, :id4 , :ol_quantity4 , :supply_w_id4 )
, :id5 , :ol_quantity5 , :supply_w_id5 )
, :id6 , :ol_quantity6 , :supply_w_id6 )
, :id7 , :ol_quantity7 , :supply_w_id7 )
, :id8 , :ol_quantity8 , :supply_w_id8 )
, :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 )
, :id1 , :ol_quantity1 , :supply_w_id1 )
, :id2 , :ol_quantity2 , :supply_w_id2 )
, :id3 , :ol_quantity3 , :supply_w_id3 )
, :id4 , :ol_quantity4 , :supply_w_id4 )
, :id5 , :ol_quantity5 , :supply_w_id5 )
, :id6 , :ol_quantity6 , :supply_w_id6 )
, :id7 , :ol_quantity7 , :supply_w_id7 )
, :id8 , :ol_quantity8 , :supply_w_id8 )
, :id9 , :ol_quantity9 , :supply_w_id9 )
, :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 )
, :id1 , :ol_quantity1 , :supply_w_id1 )
, :id2 , :ol_quantity2 , :supply_w_id2 )
, :id3 , :ol_quantity3 , :supply_w_id3 )
, :id4 , :ol_quantity4 , :supply_w_id4 )
, :id5 , :ol_quantity5 , :supply_w_id5 )
, :id6 , :ol_quantity6 , :supply_w_id6 )
, :id7 , :ol_quantity7 , :supply_w_id7 )
, :id8 , :ol_quantity8 , :supply_w_id8 )
, :id9 , :ol_quantity9 , :supply_w_id9 )
, :id10 , :ol_quantity10 , :supply_w_id10 )
, :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

FROM Table( VALUES

```

```

, 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_ol_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

, :id0 , :ol_quantity0 , :supply_w_id0 )
, ( SMALLINT( 1 )
, :id1 , :ol_quantity1 , :supply_w_id1 )
, ( SMALLINT( 2 )
, :id2 , :ol_quantity2 , :supply_w_id2 )
, ( SMALLINT( 3 )
, :id3 , :ol_quantity3 , :supply_w_id3 )
, ( SMALLINT( 4 )
, :id4 , :ol_quantity4 , :supply_w_id4 )
, ( SMALLINT( 5 )
, :id5 , :ol_quantity5 , :supply_w_id5 )
, ( SMALLINT( 6 )
, :id6 , :ol_quantity6 , :supply_w_id6 )
, ( SMALLINT( 7 )
, :id7 , :ol_quantity7 , :supply_w_id7 )
, ( SMALLINT( 8 )
, :id8 , :ol_quantity8 , :supply_w_id8 )
, ( SMALLINT( 9 )
, :id9 , :ol_quantity9 , :supply_w_id9 )
, ( SMALLINT( 10 )
, :id10 , :ol_quantity10 , :supply_w_id10 )
, ( SMALLINT( 11 )
, :id11 , :ol_quantity11 , :supply_w_id11 )
, ( SMALLINT( 12 )
, :id12 , :ol_quantity12 , :supply_w_id12 )
, ( SMALLINT( 13 )
, :id13 , :ol_quantity13 , :supply_w_id13 )
, ( SMALLINT( 14 )
, :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_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

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_7 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 )

```

```

, 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 )

) 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_8 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 )

```

```

, :id1 , :ol_quantity1 )
, :id2 , :ol_quantity2 )
, :id3 , :ol_quantity3 )
, :id4 , :ol_quantity4 )
, :id5 , :ol_quantity5 )
, :id6 , :ol_quantity6 )
, :id7 , :ol_quantity7 )

) 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_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 )

```

```

, :id6 , :ol_quantity6 )
, :id7 , :ol_quantity7 )
, :id8 , :ol_quantity8 )
, :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 )
, ( SMALLINT( 2 )
, ( SMALLINT( 3 )
, ( SMALLINT( 4 )
, ( SMALLINT( 5 )
, ( SMALLINT( 6 )
, ( SMALLINT( 7 )
, :id0 , :ol_quantity0 )
, :id1 , :ol_quantity1 )
, :id2 , :ol_quantity2 )
, :id3 , :ol_quantity3 )
, :id4 , :ol_quantity4 )
, :id5 , :ol_quantity5 )
, :id6 , :ol_quantity6 )

```

```

, :id7 , :ol_quantity7 )
, :id8 , :ol_quantity8 )
, :id9 , :ol_quantity9 )
, :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
, 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 )
, ( SMALLINT( 2 )
, ( SMALLINT( 3 )
, ( SMALLINT( 4 )
, ( SMALLINT( 5 )
, ( SMALLINT( 6 )
, ( SMALLINT( 7 )
, :id0 , :ol_quantity0 )
, :id1 , :ol_quantity1 )
, :id2 , :ol_quantity2 )
, :id3 , :ol_quantity3 )
, :id4 , :ol_quantity4 )
, :id5 , :ol_quantity5 )
, :id6 , :ol_quantity6 )

```

```

, :id7 , :ol_quantity7 )
, :id8 , :ol_quantity8 )
, :id9 , :ol_quantity9 )
, :id10 , :ol_quantity10 )
, :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
, 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 ( SELECT :next_o_id as O_ID
, :w_id AS W_ID
, :d_id as D_ID
, OL_NUMBER
, I_ID
, I_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 )
, ( SMALLINT( 2 )
, ( SMALLINT( 3 )
, ( SMALLINT( 4 )
, ( SMALLINT( 5 )
, ( SMALLINT( 6 )
, :id0 , :ol_quantity0 )
, :id1 , :ol_quantity1 )
, :id2 , :ol_quantity2 )
, :id3 , :ol_quantity3 )
, :id4 , :ol_quantity4 )
, :id5 , :ol_quantity5 )

```

```

, :id6 , :ol_quantity6 )
, :id7 , :ol_quantity7 )
, :id8 , :ol_quantity8 )
, :id9 , :ol_quantity9 )
, :id10 , :ol_quantity10 )
, :id11 , :ol_quantity11 )
, :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
, 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_14 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_ID
, I_QTY
FROM Table( VALUES
( SMALLINT( 1 )
, ( SMALLINT( 2 )
, ( SMALLINT( 3 )
, ( SMALLINT( 4 )
, :id0 , :ol_quantity0 )
, :id1 , :ol_quantity1 )
, :id2 , :ol_quantity2 )
, :id3 , :ol_quantity3 )

```

```

, :id4 , :ol_quantity4 )
, :id5 , :ol_quantity5 )
, :id6 , :ol_quantity6 )
, :id7 , :ol_quantity7 )
, :id8 , :ol_quantity8 )
, :id9 , :ol_quantity9 )
, :id10 , :ol_quantity10 )
, :id11 , :ol_quantity11 )
, :id12 , :ol_quantity12 )
, :id13 , :ol_quantity13 )
) 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
) 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_15 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 )

```

```

, :id1 , :ol_quantity1 )
, :id2 , :ol_quantity2 )
, :id3 , :ol_quantity3 )
, :id4 , :ol_quantity4 )
, :id5 , :ol_quantity5 )
, :id6 , :ol_quantity6 )
, :id7 , :ol_quantity7 )
, :id8 , :ol_quantity8 )
, :id9 , :ol_quantity9 )
, :id10 , :ol_quantity10 )
, :id11 , :ol_quantity11 )
, :id12 , :ol_quantity12 )
, :id13 , :ol_quantity13 )
, :id14 , :ol_quantity14 )
) 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
) 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
;
// 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 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++ )
{
EXEC SQL FETCH ISOL_Local_13 INTO :item_price, :item_name,
:i_data, :stockDistrictInformation , :s_data , :s_quantity ;
NEW_CURSOR_ERROR
}
break ;
case 14:
EXEC SQL OPEN ISOL_Local_14 ;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0 ; inputItemArrayIndex <
inputItemCount ; inputItemArrayIndex++ )
{
EXEC SQL FETCH ISOL_Local_14 INTO :item_price, :item_name,
:i_data, :stockDistrictInformation , :s_data , :s_quantity ;
NEW_CURSOR_ERROR
}
break ;
case 15:
EXEC SQL OPEN ISOL_Local_15 ;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0 ; inputItemArrayIndex <
inputItemCount ; inputItemArrayIndex++ )
{
EXEC SQL FETCH ISOL_Local_15 INTO :item_price, :item_name,
:i_data, :stockDistrictInformation , :s_data , :s_quantity ;
NEW_CURSOR_ERROR
}
break ;
default:
sqlerror(NEWORD_SQL, "Default switch on local
orderline/stock/index", __FILE__, __LINE__, &sqlca );
goto ferror;
}
}
else
{
switch( inputItemCount )
{
case 5:

```

```

EXEC SQL OPEN ISOL_Remote_5 ;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0 ; inputItemArrayIndex <
inputItemCount ; inputItemArrayIndex++ )
{
EXEC SQL FETCH ISOL_Remote_5 INTO :item_price, :item_name,
:i_data, :stockDistrictInformation, :s_data, :s_quantity ;
NEW_CURSOR_ERROR
}
break ;
case 6:
EXEC SQL OPEN ISOL_Remote_6 ;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0 ; inputItemArrayIndex <
inputItemCount ; inputItemArrayIndex++ )
{
EXEC SQL FETCH ISOL_Remote_6 INTO :item_price, :item_name,
:i_data, :stockDistrictInformation, :s_data, :s_quantity ;
NEW_CURSOR_ERROR
}
break ;
case 7:
EXEC SQL OPEN ISOL_Remote_7 ;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0 ; inputItemArrayIndex <
inputItemCount ; inputItemArrayIndex++ )
{
EXEC SQL FETCH ISOL_Remote_7 INTO :item_price, :item_name,
:i_data, :stockDistrictInformation, :s_data, :s_quantity ;
NEW_CURSOR_ERROR
}
break ;
case 8:
EXEC SQL OPEN ISOL_Remote_8 ;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0 ; inputItemArrayIndex <
inputItemCount ; inputItemArrayIndex++ )
{
EXEC SQL FETCH ISOL_Remote_8 INTO :item_price, :item_name,
:i_data, :stockDistrictInformation, :s_data, :s_quantity ;
NEW_CURSOR_ERROR
}
break ;
case 9:
EXEC SQL OPEN ISOL_Remote_9 ;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0 ; inputItemArrayIndex <
inputItemCount ; inputItemArrayIndex++ )
{
EXEC SQL FETCH ISOL_Remote_9 INTO :item_price, :item_name,
:i_data, :stockDistrictInformation, :s_data, :s_quantity ;
NEW_CURSOR_ERROR
}
break ;
case 10:
EXEC SQL OPEN ISOL_Remote_10 ;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0 ; inputItemArrayIndex <
inputItemCount ; inputItemArrayIndex++ )
{
EXEC SQL FETCH ISOL_Remote_10 INTO :item_price,
:item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity ;
NEW_CURSOR_ERROR
}
break ;
case 11:
EXEC SQL OPEN ISOL_Remote_11 ;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0 ; inputItemArrayIndex <
inputItemCount ; inputItemArrayIndex++ )
{
EXEC SQL FETCH ISOL_Remote_11 INTO :item_price,
:item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity ;
NEW_CURSOR_ERROR
}
break ;
case 12:
EXEC SQL OPEN ISOL_Remote_12 ;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0 ; inputItemArrayIndex <
inputItemCount ; inputItemArrayIndex++ )
{
EXEC SQL FETCH ISOL_Remote_12 INTO :item_price,
:item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity ;
NEW_CURSOR_ERROR
}
break ;
case 13:
EXEC SQL OPEN ISOL_Remote_13 ;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0 ; inputItemArrayIndex <
inputItemCount ; inputItemArrayIndex++ )
{
EXEC SQL FETCH ISOL_Remote_13 INTO :item_price,
:item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity ;
NEW_CURSOR_ERROR
}
break ;
case 14:
EXEC SQL OPEN ISOL_Remote_14 ;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0 ; inputItemArrayIndex <
inputItemCount ; inputItemArrayIndex++ )

```

```

{
EXEC SQL FETCH ISOL_Remote_14 INTO :item_price,
:item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity ;
NEW_CURSOR_ERROR
}
break ;
case 15:
EXEC SQL OPEN ISOL_Remote_15 ;
NEW_CURSOR_OPEN_ERROR
for ( inputItemArrayIndex = 0 ; inputItemArrayIndex <
inputItemCount ; inputItemArrayIndex++ )
{
EXEC SQL FETCH ISOL_Remote_15 INTO :item_price,
:item_name, :i_data, :stockDistrictInformation, :s_data, :s_quantity ;
NEW_CURSOR_ERROR
}
break ;
default:
sqlerror(NEWORD_SQL, "Default switch on remote
orderline/stock/index", __FILE__, __LINE__, &sqlca);
goto ferror;
}
}
for ( inputItemArrayIndex = 0 ;
inputItemArrayIndex < in_neword->s_O_OL_CNT // from input
&& i_priceArray[ inputItemArrayIndex ] != 0 ;
inputItemArrayIndex++ )
{
// s_I_NAME, and s_S_QUANTITY already set as output host variables
neword->item[ inputItemArrayIndex ].s_I_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 ) )
{
neword->item[ inputItemArrayIndex ].s_brand_generic = 'B';
}
else
{
neword->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 ( neword->s_O_OL_CNT == in_neword->s_O_OL_CNT )
{
neword->s_transtatus = TRAN_OK ;
EXEC SQL COMMIT;
if( sqlca.sqlcode != 0 )
{
sqlerror(NEWORD_SQL, "COMMIT", __FILE__, __LINE__, &sqlca ) ;
goto ferror;
}
}
else
{
neword->s_transtatus = INVALID_ITEM ;
EXEC SQL ROLLBACK WORK ;
if ( sqlca.sqlcode != 0 )
{
neword->s_transtatus = FATAL_SQLERROR;
sqlerror(NEWORD_SQL, "ROLLBACK FAILED (INVALID ITEM)",
__FILE__, __LINE__, &sqlca);
// no point in ferror
}
}
}
else
{
DLCHK( retry_tran );
sqlerror( NEWORD_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( NEWORD_SQL, tempstr, __FILE__, __LINE__, &sqlca ) ;
}

ferror:
newword->s_transtatus = FATAL_SQLERROR;

EXEC SQL ROLLBACK WORK;

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

goto mexit ;

/*
** A little function to search for the string "ORIGINAL" given a string
and
** it's 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*/
8,8,8,8,1,8,8,8,8, /*60-69*/
8,4,8,3,8,8,0,8,2,7, /*70-79*/
8,8,6,8,8,8,8,8,8, /*80-89*/
8,8,8,8,8,8,8,8,8, /*90-99*/
8,8,8,8,8,8,8,8,8, /*100-109*/
8,8,8,8,8,8,8,8,8, /*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 ) ;
}

```

9.2.8. Src.Srv/uncat-func.ddl

```

-----
--
-- Licensed Materials - Property of IBM
--
-- (C) COPYRIGHT International Business Machines Corp. 1996, 2010
-- 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 %

```

9.2.9. include/db2tpcc.h

```

/*****
**
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1996, 2010
** 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>
typedef __int16 int16_t;
typedef __int32 int32_t;
typedef __int64 int64_t;

#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

/* *****
*/
/* Whenever changing these structures, you MUST update the char[] array
*/
/* size in the varchar hostvars, the xx->len values in Src.Cli/???c.sgc
*/
/* and the utils/cat.ddl and utils/uncat.ddl scripts.
*/
/*
*/
/* In all cases, the proper value to use is:
*/
/*
*/
/*          sizeof(struct xxx) - SPGENERAL_ADJUST
*/
/*
*/
/* SPGENERAL_PAD is the number of int16_ts required to pad the structures
*/
/* so that the data is aligned the same as if the len/pad elements were
*/
/* not present.
*/
/*
*/
/* NOTE: It is almost guaranteed that this will break when using clients
*/
/* that use a different byte ordering and/or compiler padding.
*/
/* *****
*/

#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);

extern int create_context(void);
extern int destroy_context(void);
extern int get_context(void**);
extern int attach_context(void*);
extern int detach_context(void*);

#ifdef __cplusplus
}
#endif

#endif // __DB2TPCC_H

```

9.2.10. include/lval.h

```

/* lval.h - generated automatically at 20130205.0910 */

#ifdef __LVAL_H
#define __LVAL_H
#define WAREHOUSES 104040
#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

```

9.2.11. include/tpccapp.h

```

/*****
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1996, 2010
** 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 [ 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; }

/* *****
*/
/* 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: HHHXNNNNSSSSXNNNNSSSS...
*/
/* (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

```

9.2.12. include/tpccdbg.h

```

/*****
****
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1996, 2010
** 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

```

9.2.13. tpccenv.sh

```

#!/bin/sh

#####
###
### Licensed Materials - Property of IBM
###
### (C) COPYRIGHT International Business Machines Corp. 1996, 2010
### 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=CK101019
#
# The DB2 Instance Name (for DB2)
export DB2INSTANCE=${USER}
#
# The OS being used (i.e. "UNIX", "LINUX", "WINDOWS")
export PLATFORM=UNIX
#
# The type of make command and slash used by the OS.
# (i.e. UNIX - "/", WINDOWS - "\").
# These are referenced all over the kit.
export SLASH="/";
export MAKE=make
#
# Specifies whether or not to use dari stored proc's for the TPC-C driver.
Set to either DARIVERISION or NONDARI;
#export TPCC_SPTYPE=NOSP
#export TPCC_SPTYPE=SPGENERAL2
export TPCC_SPTYPE=SPGENERAL
#export TPCC_SPTYPE=DARI2SQLDA
#
# The schema name is typically the SQL authorization ID (or username).
# This is required for runstats and EEE.
export TPCC_SCHEMA=${USER}
export SERVER_TPCC_SCHEMA=${USER}
#
# DB2 EE/EEE Configuration
export DB2EDITION=EE
#export DB2EDITION=DPF
#
# TPCC General Configuration
export TPCC_DBNAME=TPCC
export TPCC_ROOT=${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

10.1.1. db2set

```
DB2_LARGE_PAGE_MEM=DB
DB2_RESOURCE_POLICY=/autobench/sources/db2_tpc-1/users/tpcc/tpc-
c.ibm/cfg/db2-resource-policy.cfg
DB2_SELJUDI_COMM_BUFFER=YES
DB2_USE_ALTERNATE_PAGE_CLEANING=YES
DB2_MAX_NON_TABLE_LOCKS=500
DB2_RCT_FEATURES=GROUPUPDATE=ON
DB2_TRUSTED_BINDIN=ON
DB2_KEEPTABLELOCK=CONNECTION
DB2_NO_FORK_CHECK=ON
DB2_ALLOCATION_SIZE=8388608
DB2_FMP_COMM_HEAPSZ=1000
DB2_APM_PERFORMANCE=ALL
DB2ASSUMEUPDATE=ON
bpvars.cfg
DB2CHECKCLIENTINTERVAL=0
DB2_HASH_JOIN=OFF
DB2CHKSQLDA=OFF
DB2COMM=tcPIP
```

10.1.2. db2 get db cfg for tpc

Database Configuration for Database tpc

```
Database configuration release level = 0x0d00
Database release level = 0x0d00

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) =
Number compatibility = OFF
Varchar2 compatibility = OFF
Date compatibility = OFF
Database page size = 4096

Dynamic SQL Query management (DYN_QUERY_MGMT) = DISABLE

Statement concentrator (STMT_CONC) = OFF

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 types for opt (DFT_MTTB_TYPES) = SYSTEM
Number of frequent values retained (NUM_FREQVALUES) = 10
Number of quantiles retained (NUM_QUANTILES) = 20

Decimal floating point rounding mode (DECFLT_ROUNDING) = ROUND_HALF_EVEN

Backup pending = NO

All committed transactions have been written to disk = 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) = 191676416
Database memory threshold (DB_MEM_THRESH) = 10
Max storage for lock list (4KB) (LOCKLIST) = 64000
Percent. of lock lists per application (MAXLOCKS) = 100
Package cache size (4KB) (PCKCACHE_SZ) = (MAXAPPLS*8)
Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 250
Sort list heap (4KB) (SORTHEAP) = 16

Database heap (4KB) (DBHEAP) = 524288
Catalog cache size (4KB) (CATALOGCACHE_SZ) = (MAXAPPLS*5)
Log buffer size (4KB) (LOGBUFSZ) = 18000
Utilities heap size (4KB) (UTIL_HEAP_SZ) = 5000
Buffer pool size (pages) (BUFFPAGE) = 1000
SQL statement heap (4KB) (STMTHAP) = 131072
Default application heap (4KB) (APPLHEAPSZ) = 1000
Application Memory Size (4KB) (APPL_MEMORY) =
AUTOMATIC(40000)
```

```
Statistics heap size (4KB) (STAT_HEAP_SZ) = AUTOMATIC(4384)

Interval for checking deadlock (ms) (DLCHKTIME) = 3000
Lock timeout (sec) (LOCKTIMEOUT) = -1

Changed pages threshold (CHNGPGS_THRESH) = 60
Number of asynchronous page cleaners (NUM_IOCLEANERS) = 0
Number of I/O servers (NUM_IOSERVERS) = AUTOMATIC(3)
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) = 1000
Average number of active applications (AVG_APPLS) = 1
Max DB files open per application (MAXFILOP) = 61440

Log file size (4KB) (LOGFILSIZ) = 512000
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/tpcc/log
Overflow log path (OVERFLOWLOGPATH) =
Mirror log path (MIRRORLOGPATH) =
First active log file = S0000005.LOG
Block log on disk full (BLK_LOG_DSK_FUL) = NO
Block non logged operations (BLOCKNONLOGGED) = 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) = 4177
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
HADR peer window duration (seconds) (HADR_PEER_WINDOW) = 0

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
Auto deletion of recovery objects (AUTO_DEL_REC_OBJ) = OFF

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 statement statistics (AUTO_STMT_STATS) = OFF
Automatic statistics profiling (AUTO_STATS_PROF) = OFF
Automatic profile updates (AUTO_PROF_UPD) = OFF
Automatic reorganization (AUTO_REORG) = OFF

Auto-Revalidation (AUTO_REVAL) = DISABLED
Currently Committed (CUR_COMMIT) = DISABLED
CHAR output with DECIMAL input (DEC_TO_CHAR_FMT) = NEW
Enable XML Character operations (ENABLE_XMLCHAR) = YES
WLM Collection Interval (minutes) (WLM_COLLECT_INT) = 0
Monitor Collect Settings
Request metrics (MON_REQ_METRICS) = NONE
Activity metrics (MON_ACT_METRICS) = NONE
Object metrics (MON_OBJ_METRICS) = NONE
Unit of work events (MON_UOW_DATA) = NONE
Lock timeout events (MON_LOCKTIMEOUT) = NONE
Deadlock events (MON_DEADLOCK) = WITHOUT_HIST
Lock wait events (MON_LOCKWAIT) = NONE
Lock wait event threshold (MON_LW_THRESH) = 5000000
Number of package list entries (MON_PKGLIST_SZ) = 32
Lock event notification level (MON_LCK_MSG_LVL) = 1

SMTP Server (SMTP_SERVER) =
SQL conditional compilation flags (SQL_CFLAGS) =
Section actuals setting (SECTION_ACTUALS) = NONE
Connect procedure (CONNECT_PROC) =
```

10.1.3. db2 get dbm cfg

Database Manager Configuration

Node type = Enterprise Server Edition with local and remote clients

```
Database manager configuration release level          = 0x0d00

CPU speed (millisec/instruction)                   (CPUSPEED) = 1.220223e-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) =
/autobench/sources/db2_tpc-1/users/tpcc/sqljlib/java/jdk64

Diagnostic error capture level                      (DIAGLEVEL) = 3
Notify Level                                        (NOTIFYLEVEL) = 3
Diagnostic data directory path                      (DIAGPATH) =
Alternate diagnostic data directory path (ALT_DIAGPATH) =
Size of rotating db2diag & notify logs (MB)        (DIAGSIZE) = 0

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
Cluster manager                                  (CLUSTER_MGR) =

Database manager authentication                    (AUTHENTICATION) = CLIENT
Alternate authentication                          (ALTERNATE_AUTH_ENC) = NOT_SPECIFIED
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) =
/autobench/sources/db2_tpc-1/users/tpcc

Database monitor heap size (4KB)                  (MON_HEAP_SZ) = 4096
Java Virtual Machine heap size (4KB)              (JAVA_HEAP_SZ) = 2048
Audit buffer size (4KB)                           (AUDIT_BUF_SZ) = 0
Size of instance shared memory (4KB)              (INSTANCE_MEMORY) = 192462848
Agent stack size                                  (AGENT_STACK_SZ) = 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)              (RQRIOBLK) = 4096
Workload impact by throttled utilities(UTIL_IMPACT_LIM) = 10

Priority of agents                                (AGENTPRI) = SYSTEM
Agent pool size                                   (NUM_POOLAGENTS) = 0
Initial number of agents in pool                  (NUM_INITAGENTS) = 0
Max number of coordinating agents                 (MAX_COORDAGENTS) = AUTOMATIC(200)
Max number of client connections                 (MAX_CONNECTIONS) =
AUTOMATIC(MAX_COORDAGENTS)

Keep fenced process                              (KEEPFENCED) = YES
Number of pooled fenced processes                 (FENCED_POOL) =
AUTOMATIC(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
```

```
SSL server keydb file                            (SSL_SVR_KEYDB) =
SSL server stash file                            (SSL_SVR_STASH) =
SSL server certificate label                     (SSL_SVR_LABEL) =
SSL service name                                 (SSL_SVCENAME) =
SSL cipher specs                                 (SSL_CIPHERSPECS) =
SSL versions                                     (SSL_VERSIONS) =
SSL client keydb file                            (SSL_CLNT_KEYDB) =
SSL client stash file                           (SSL_CLNT_STASH) =

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(4096)
No. of int. communication channels               (FCM_NUM_CHANNELS) = AUTOMATIC(2048)
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
```

10.1.4. db2-resource-policy.cfg

```
<RESOURCE_POLICY>
  <DATABASE_RESOURCE_POLICY>
    <DBNAME>TPCC</DBNAME>
    <METHOD>CPUMASK</METHOD>
    <RESOURCE_BINDING>
      <RESOURCE>0xFFFF</RESOURCE>
      <DBMEM_PERCENTAGE>49.90</DBMEM_PERCENTAGE>
      <SERVICE_NAME>db2ctpc</SERVICE_NAME>
      <BUFFERPOOL_BINDING>
        <NUM_CLEANERS>3</NUM_CLEANERS>
        <BUFFERPOOL_ID>5</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>23</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>41</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>59</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>77</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>95</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>113</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>131</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>150</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>168</BUFFERPOOL_ID>
      </BUFFERPOOL_BINDING>
      <BUFFERPOOL_BINDING>
        <NUM_CLEANERS>3</NUM_CLEANERS>
        <BUFFERPOOL_ID>6</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>24</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>42</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>60</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>78</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>96</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>114</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>132</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>151</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>169</BUFFERPOOL_ID>
      </BUFFERPOOL_BINDING>
      <BUFFERPOOL_BINDING>
        <NUM_CLEANERS>3</NUM_CLEANERS>
        <BUFFERPOOL_ID>7</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>25</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>43</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>61</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>79</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>97</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>115</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>133</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>152</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>170</BUFFERPOOL_ID>
      </BUFFERPOOL_BINDING>
      <BUFFERPOOL_BINDING>
        <NUM_CLEANERS>3</NUM_CLEANERS>
        <BUFFERPOOL_ID>8</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>26</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>44</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>62</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>80</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>98</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>116</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>134</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>153</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>171</BUFFERPOOL_ID>
      </BUFFERPOOL_BINDING>
      <BUFFERPOOL_BINDING>
        <NUM_CLEANERS>3</NUM_CLEANERS>
        <BUFFERPOOL_ID>9</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>27</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>45</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>63</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>81</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>99</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>117</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>135</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>154</BUFFERPOOL_ID>
        <BUFFERPOOL_ID>172</BUFFERPOOL_ID>
      </BUFFERPOOL_BINDING>
      <BUFFERPOOL_BINDING>
        <NUM_CLEANERS>3</NUM_CLEANERS>
        <BUFFERPOOL_ID>10</BUFFERPOOL_ID>
```



```

<BUFFERPOOL_ID>28</BUFFERPOOL_ID>
<BUFFERPOOL_ID>46</BUFFERPOOL_ID>
<BUFFERPOOL_ID>64</BUFFERPOOL_ID>
<BUFFERPOOL_ID>82</BUFFERPOOL_ID>
<BUFFERPOOL_ID>100</BUFFERPOOL_ID>
<BUFFERPOOL_ID>118</BUFFERPOOL_ID>
<BUFFERPOOL_ID>136</BUFFERPOOL_ID>
<BUFFERPOOL_ID>155</BUFFERPOOL_ID>
<BUFFERPOOL_ID>173</BUFFERPOOL_ID>
</BUFFERPOOL_BINDING>
<BUFFERPOOL_BINDING>
<NUM_CLEANERS>3</NUM_CLEANERS>
<BUFFERPOOL_ID>11</BUFFERPOOL_ID>
<BUFFERPOOL_ID>29</BUFFERPOOL_ID>
<BUFFERPOOL_ID>47</BUFFERPOOL_ID>
<BUFFERPOOL_ID>65</BUFFERPOOL_ID>
<BUFFERPOOL_ID>83</BUFFERPOOL_ID>
<BUFFERPOOL_ID>101</BUFFERPOOL_ID>
<BUFFERPOOL_ID>119</BUFFERPOOL_ID>
<BUFFERPOOL_ID>137</BUFFERPOOL_ID>
<BUFFERPOOL_ID>156</BUFFERPOOL_ID>
<BUFFERPOOL_ID>174</BUFFERPOOL_ID>
</BUFFERPOOL_BINDING>
<BUFFERPOOL_BINDING>
<NUM_CLEANERS>3</NUM_CLEANERS>
<BUFFERPOOL_ID>12</BUFFERPOOL_ID>
<BUFFERPOOL_ID>30</BUFFERPOOL_ID>
<BUFFERPOOL_ID>48</BUFFERPOOL_ID>
<BUFFERPOOL_ID>66</BUFFERPOOL_ID>
<BUFFERPOOL_ID>84</BUFFERPOOL_ID>
<BUFFERPOOL_ID>102</BUFFERPOOL_ID>
<BUFFERPOOL_ID>120</BUFFERPOOL_ID>
<BUFFERPOOL_ID>138</BUFFERPOOL_ID>
<BUFFERPOOL_ID>157</BUFFERPOOL_ID>
<BUFFERPOOL_ID>175</BUFFERPOOL_ID>
</BUFFERPOOL_BINDING>
<BUFFERPOOL_BINDING>
<NUM_CLEANERS>3</NUM_CLEANERS>
<BUFFERPOOL_ID>13</BUFFERPOOL_ID>
<BUFFERPOOL_ID>31</BUFFERPOOL_ID>
<BUFFERPOOL_ID>49</BUFFERPOOL_ID>
<BUFFERPOOL_ID>67</BUFFERPOOL_ID>
<BUFFERPOOL_ID>85</BUFFERPOOL_ID>
<BUFFERPOOL_ID>103</BUFFERPOOL_ID>
<BUFFERPOOL_ID>121</BUFFERPOOL_ID>
<BUFFERPOOL_ID>139</BUFFERPOOL_ID>
<BUFFERPOOL_ID>158</BUFFERPOOL_ID>
<BUFFERPOOL_ID>176</BUFFERPOOL_ID>
</BUFFERPOOL_BINDING>
</RESOURCE_BINDING>
<RESOURCE_BINDING>
<RESOURCE>0x00000000</RESOURCE>
<DBMEM_PERCENTAGE>49.90</DBMEM_PERCENTAGE>
<SERVICE_NAME>DB2_tpc2</SERVICE_NAME>
<BUFFERPOOL_BINDING>
<NUM_CLEANERS>3</NUM_CLEANERS>
<BUFFERPOOL_ID>14</BUFFERPOOL_ID>
<BUFFERPOOL_ID>32</BUFFERPOOL_ID>
<BUFFERPOOL_ID>50</BUFFERPOOL_ID>
<BUFFERPOOL_ID>68</BUFFERPOOL_ID>
<BUFFERPOOL_ID>86</BUFFERPOOL_ID>
<BUFFERPOOL_ID>104</BUFFERPOOL_ID>
<BUFFERPOOL_ID>122</BUFFERPOOL_ID>
<BUFFERPOOL_ID>140</BUFFERPOOL_ID>
<BUFFERPOOL_ID>159</BUFFERPOOL_ID>
<BUFFERPOOL_ID>177</BUFFERPOOL_ID>
</BUFFERPOOL_BINDING>
<BUFFERPOOL_BINDING>
<NUM_CLEANERS>3</NUM_CLEANERS>
<BUFFERPOOL_ID>15</BUFFERPOOL_ID>
<BUFFERPOOL_ID>33</BUFFERPOOL_ID>
<BUFFERPOOL_ID>51</BUFFERPOOL_ID>
<BUFFERPOOL_ID>69</BUFFERPOOL_ID>
<BUFFERPOOL_ID>87</BUFFERPOOL_ID>
<BUFFERPOOL_ID>105</BUFFERPOOL_ID>
<BUFFERPOOL_ID>123</BUFFERPOOL_ID>
<BUFFERPOOL_ID>141</BUFFERPOOL_ID>
<BUFFERPOOL_ID>160</BUFFERPOOL_ID>
<BUFFERPOOL_ID>178</BUFFERPOOL_ID>
</BUFFERPOOL_BINDING>
<BUFFERPOOL_BINDING>
<NUM_CLEANERS>3</NUM_CLEANERS>
<BUFFERPOOL_ID>16</BUFFERPOOL_ID>
<BUFFERPOOL_ID>34</BUFFERPOOL_ID>
<BUFFERPOOL_ID>52</BUFFERPOOL_ID>
<BUFFERPOOL_ID>70</BUFFERPOOL_ID>
<BUFFERPOOL_ID>88</BUFFERPOOL_ID>
<BUFFERPOOL_ID>106</BUFFERPOOL_ID>
<BUFFERPOOL_ID>124</BUFFERPOOL_ID>
<BUFFERPOOL_ID>142</BUFFERPOOL_ID>
<BUFFERPOOL_ID>161</BUFFERPOOL_ID>
<BUFFERPOOL_ID>179</BUFFERPOOL_ID>
</BUFFERPOOL_BINDING>
<BUFFERPOOL_BINDING>
<NUM_CLEANERS>3</NUM_CLEANERS>
<BUFFERPOOL_ID>17</BUFFERPOOL_ID>
<BUFFERPOOL_ID>35</BUFFERPOOL_ID>
<BUFFERPOOL_ID>53</BUFFERPOOL_ID>
<BUFFERPOOL_ID>71</BUFFERPOOL_ID>
<BUFFERPOOL_ID>89</BUFFERPOOL_ID>
<BUFFERPOOL_ID>107</BUFFERPOOL_ID>
<BUFFERPOOL_ID>125</BUFFERPOOL_ID>
<BUFFERPOOL_ID>143</BUFFERPOOL_ID>

```

```

<BUFFERPOOL_ID>162</BUFFERPOOL_ID>
<BUFFERPOOL_ID>180</BUFFERPOOL_ID>
</BUFFERPOOL_BINDING>
<BUFFERPOOL_BINDING>
<NUM_CLEANERS>3</NUM_CLEANERS>
<BUFFERPOOL_ID>18</BUFFERPOOL_ID>
<BUFFERPOOL_ID>36</BUFFERPOOL_ID>
<BUFFERPOOL_ID>54</BUFFERPOOL_ID>
<BUFFERPOOL_ID>72</BUFFERPOOL_ID>
<BUFFERPOOL_ID>90</BUFFERPOOL_ID>
<BUFFERPOOL_ID>108</BUFFERPOOL_ID>
<BUFFERPOOL_ID>126</BUFFERPOOL_ID>
<BUFFERPOOL_ID>144</BUFFERPOOL_ID>
<BUFFERPOOL_ID>163</BUFFERPOOL_ID>
<BUFFERPOOL_ID>181</BUFFERPOOL_ID>
</BUFFERPOOL_BINDING>
<BUFFERPOOL_BINDING>
<NUM_CLEANERS>3</NUM_CLEANERS>
<BUFFERPOOL_ID>19</BUFFERPOOL_ID>
<BUFFERPOOL_ID>37</BUFFERPOOL_ID>
<BUFFERPOOL_ID>55</BUFFERPOOL_ID>
<BUFFERPOOL_ID>73</BUFFERPOOL_ID>
<BUFFERPOOL_ID>91</BUFFERPOOL_ID>
<BUFFERPOOL_ID>109</BUFFERPOOL_ID>
<BUFFERPOOL_ID>127</BUFFERPOOL_ID>
<BUFFERPOOL_ID>145</BUFFERPOOL_ID>
<BUFFERPOOL_ID>164</BUFFERPOOL_ID>
<BUFFERPOOL_ID>182</BUFFERPOOL_ID>
</BUFFERPOOL_BINDING>
<BUFFERPOOL_BINDING>
<NUM_CLEANERS>3</NUM_CLEANERS>
<BUFFERPOOL_ID>20</BUFFERPOOL_ID>
<BUFFERPOOL_ID>38</BUFFERPOOL_ID>
<BUFFERPOOL_ID>56</BUFFERPOOL_ID>
<BUFFERPOOL_ID>74</BUFFERPOOL_ID>
<BUFFERPOOL_ID>92</BUFFERPOOL_ID>
<BUFFERPOOL_ID>110</BUFFERPOOL_ID>
<BUFFERPOOL_ID>128</BUFFERPOOL_ID>
<BUFFERPOOL_ID>146</BUFFERPOOL_ID>
<BUFFERPOOL_ID>165</BUFFERPOOL_ID>
<BUFFERPOOL_ID>183</BUFFERPOOL_ID>
</BUFFERPOOL_BINDING>
<BUFFERPOOL_BINDING>
<NUM_CLEANERS>3</NUM_CLEANERS>
<BUFFERPOOL_ID>21</BUFFERPOOL_ID>
<BUFFERPOOL_ID>39</BUFFERPOOL_ID>
<BUFFERPOOL_ID>57</BUFFERPOOL_ID>
<BUFFERPOOL_ID>75</BUFFERPOOL_ID>
<BUFFERPOOL_ID>93</BUFFERPOOL_ID>
<BUFFERPOOL_ID>111</BUFFERPOOL_ID>
<BUFFERPOOL_ID>129</BUFFERPOOL_ID>
<BUFFERPOOL_ID>147</BUFFERPOOL_ID>
<BUFFERPOOL_ID>166</BUFFERPOOL_ID>
<BUFFERPOOL_ID>184</BUFFERPOOL_ID>
</BUFFERPOOL_BINDING>
<BUFFERPOOL_BINDING>
<NUM_CLEANERS>3</NUM_CLEANERS>
<BUFFERPOOL_ID>22</BUFFERPOOL_ID>
<BUFFERPOOL_ID>40</BUFFERPOOL_ID>
<BUFFERPOOL_ID>58</BUFFERPOOL_ID>
<BUFFERPOOL_ID>76</BUFFERPOOL_ID>
<BUFFERPOOL_ID>94</BUFFERPOOL_ID>
<BUFFERPOOL_ID>112</BUFFERPOOL_ID>
<BUFFERPOOL_ID>130</BUFFERPOOL_ID>
<BUFFERPOOL_ID>148</BUFFERPOOL_ID>
<BUFFERPOOL_ID>167</BUFFERPOOL_ID>
<BUFFERPOOL_ID>185</BUFFERPOOL_ID>
</BUFFERPOOL_BINDING>
</RESOURCE_BINDING>
</DATABASE_RESOURCE_POLICY>
</RESOURCE_POLICY>

```

10.2. Transaction Monitor Parameters

10.2.1. inetinfo_registry.reg

```

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\InetInfo\Parameters
MaxConcurrency REG_DWORD 0x640
MaxPoolThreads REG_DWORD 0x190
PoolThreadLimit REG_DWORD 0x640

```

10.2.2. tcpip_parameters_registry.reg

Windows Registry Editor Version 5.00

```

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters]
"ICSDomain"="mshome.net"
"SyncDomainWithMembership"=dword:00000001
"NV Hostname"="tpcc-client1"
"DataBasePath"=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,\
64,00,72,00,69,00,76,00,65,00,72,00,73,00,5c,00,65,00,74,00,63,00,00,00
"NameServer"=""
"ForwardBroadcasts"=dword:00000000
"IPEnableRouter"=dword:00000000

```

```
"Domain"=""
"Hostname"="tpcc-client1"
"SearchList"="austin.ibm.com,linuxperf9025.net"
"UseDomainNameDevolution"=dword:0000001
"EnableICMPRedirect"=dword:00000001
"DeadGWDetectDefault"=dword:00000001
"DontAddDefaultGatewayDefault"=dword:00000000
"MaxUserPort"=dword:0000fffe
"TcpMaxDataRetransmissions"=dword:0000fffc

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Adapters]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Adapters\{02E71305-9C31-498F-9EC3-5AEAE7EFEB9}]
"LLInterface"=""
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
00,61,00,63,00,65,00,73,00,5c,00,7b,00,30,00,32,00,45,00,37,00,31,00,33,00,\
30,00,35,00,2d,00,39,00,43,00,33,00,31,00,2d,00,34,00,39,00,38,00,46,00,2d,\
00,39,00,45,00,43,00,33,00,2d,00,35,00,41,00,45,00,41,00,45,00,37,00,45,00,\
46,00,45,00,45,00,42,00,39,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Adapters\{170923B8-287F-4AA5-9B62-ACC3DF721737}]
"LLInterface"=""
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
00,61,00,63,00,65,00,73,00,5c,00,7b,00,31,00,37,00,30,00,39,00,32,00,33,00,\
42,00,38,00,2d,00,32,00,38,00,37,00,46,00,2d,00,34,00,41,00,41,00,35,00,2d,\
00,39,00,42,00,36,00,32,00,2d,00,41,00,43,00,43,00,33,00,44,00,46,00,37,00,\
32,00,31,00,37,00,33,00,37,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Adapters\{1F5668B7-A26C-483F-B719-EF9333C9C4DC}]
"LLInterface"=""
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
00,61,00,63,00,65,00,73,00,5c,00,7b,00,31,00,46,00,35,00,36,00,36,00,38,00,\
42,00,37,00,2d,00,41,00,32,00,36,00,43,00,2d,00,34,00,38,00,33,00,46,00,2d,\
00,42,00,37,00,31,00,39,00,2d,00,45,00,46,00,39,00,33,00,33,00,38,00,43,00,\
39,00,43,00,34,00,44,00,43,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Adapters\{511B57FE-B4D6-4121-9D12-C4E2BA398E58}]
"LLInterface"=""
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
00,61,00,63,00,65,00,73,00,5c,00,7b,00,35,00,31,00,31,00,42,00,35,00,37,00,\
46,00,45,00,2d,00,42,00,34,00,44,00,36,00,2d,00,34,00,31,00,32,00,31,00,2d,\
00,39,00,44,00,31,00,32,00,2d,00,43,00,34,00,45,00,32,00,42,00,41,00,33,00,\
39,00,38,00,45,00,35,00,38,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Adapters\{62C84429-9356-407B-9572-B798C6C8AC9A}]
"LLInterface"=""
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
00,61,00,63,00,65,00,73,00,5c,00,7b,00,36,00,32,00,43,00,38,00,34,00,34,00,\
32,00,39,00,2d,00,39,00,33,00,35,00,36,00,2d,00,34,00,30,00,37,00,42,00,2d,\
\
```

```
00,39,00,35,00,37,00,32,00,2d,00,42,00,37,00,39,00,38,00,43,00,36,00,43,00,\
38,00,41,00,43,00,39,00,41,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Adapters\{724248B6-966D-41C0-8E01-F7C08D0E8648}]
"LLInterface"=""
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
00,61,00,63,00,65,00,73,00,5c,00,7b,00,37,00,32,00,34,00,32,00,34,00,38,00,\
42,00,36,00,2d,00,39,00,36,00,36,00,44,00,2d,00,34,00,31,00,43,00,30,00,2d,\
00,38,00,45,00,30,00,31,00,2d,00,46,00,37,00,43,00,30,00,38,00,44,00,30,00,\
45,00,38,00,36,00,34,00,38,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Adapters\{72E75E72-4FF7-4DE0-A81E-B4721EA039EE}]
"LLInterface"=""
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
00,61,00,63,00,65,00,73,00,5c,00,7b,00,37,00,32,00,45,00,37,00,35,00,45,00,\
37,00,32,00,2d,00,34,00,46,00,46,00,37,00,2d,00,34,00,44,00,45,00,30,00,2d,\
00,41,00,38,00,31,00,45,00,2d,00,42,00,34,00,37,00,32,00,31,00,45,00,41,00,\
30,00,33,00,39,00,45,00,45,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Adapters\{7C1C96DA-E1EC-4771-A1BC-B107F1AD3B8E}]
"LLInterface"=""
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
00,61,00,63,00,65,00,73,00,5c,00,7b,00,37,00,43,00,31,00,43,00,39,00,36,00,\
44,00,41,00,2d,00,45,00,31,00,45,00,43,00,2d,00,34,00,37,00,37,00,31,00,2d,\
00,41,00,31,00,42,00,43,00,2d,00,42,00,31,00,30,00,37,00,46,00,31,00,41,00,\
44,00,33,00,42,00,38,00,45,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Adapters\{C7B5C48B-7281-4F2B-B106-6DAE0493EE4D}]
"LLInterface"=""
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
00,61,00,63,00,65,00,73,00,5c,00,7b,00,43,00,37,00,42,00,35,00,43,00,34,00,\
38,00,42,00,2d,00,37,00,32,00,38,00,31,00,2d,00,34,00,46,00,32,00,42,00,2d,\
00,42,00,31,00,30,00,36,00,2d,00,36,00,44,00,41,00,45,00,30,00,34,00,39,00,\
33,00,45,00,45,00,34,00,44,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Adapters\{D0264C87-0470-4A06-A0D7-63EF81968E35}]
"LLInterface"=""
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
00,61,00,63,00,65,00,73,00,5c,00,7b,00,44,00,30,00,32,00,36,00,34,00,43,00,\
38,00,37,00,2d,00,30,00,34,00,37,00,30,00,2d,00,34,00,41,00,30,00,36,00,2d,\
00,41,00,30,00,44,00,37,00,2d,00,36,00,33,00,45,00,46,00,38,00,31,00,39,00,\
36,00,38,00,45,00,33,00,35,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Adapters\{D7C8DBF2-6FC6-4873-9467-8A0637B1D07A}]
"LLInterface"=""
```

```
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
00,61,00,63,00,65,00,73,00,5c,00,7b,00,44,00,37,00,43,00,38,00,44,00,42,00,\
46,00,32,00,2d,00,36,00,46,00,43,00,36,00,2d,00,34,00,38,00,37,00,33,00,2d,\
00,39,00,34,00,36,00,37,00,2d,00,38,00,41,00,30,00,36,00,33,00,37,00,42,00,\
31,00,44,00,30,00,37,00,41,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Ada
pters\{DF3B01C1-9D76-427F-BA7A-17AEAF182E30}]
"LLInterface"=""
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
00,61,00,63,00,65,00,73,00,5c,00,7b,00,44,00,46,00,33,00,42,00,30,00,31,00,\
43,00,31,00,2d,00,39,00,44,00,37,00,36,00,2d,00,34,00,32,00,37,00,46,00,2d,\
00,42,00,41,00,37,00,41,00,2d,00,31,00,37,00,41,00,45,00,41,00,46,00,31,00,\
38,00,32,00,45,00,33,00,30,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Ada
pters\{F7E95255-579D-4EF7-9A8B-20B4C40B7C48}]
"LLInterface"=""
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
00,61,00,63,00,65,00,73,00,5c,00,7b,00,46,00,37,00,45,00,39,00,35,00,32,00,\
35,00,35,00,2d,00,35,00,37,00,39,00,44,00,2d,00,34,00,45,00,46,00,37,00,2d,\
00,39,00,41,00,38,00,42,00,2d,00,32,00,30,00,42,00,34,00,43,00,34,00,30,00,\
42,00,37,00,43,00,34,00,38,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\DNS
RegisteredAdapters]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Int
erfaces]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Int
erfaces\{02E71305-9C31-498F-9EC3-5AAE7FEFEB9}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000000
"NameServer"="10.2.0.253"
"Domain"=""
"RegistrationEnabled"=dword:00000001
"RegisterAdapterName"=dword:00000000
"DhcpServer"="255.255.255.255"
"Lease"=dword:00000000
"LeaseObtainedTime"=dword:00000000
"T1"=dword:00000000
"T2"=dword:00000000
"LeaseTerminatesTime"=dword:00000000
"AddressType"=dword:00000000
"IsServerNapAware"=dword:00000000
"DhcpConnForceBroadcastFlag"=dword:00000000
"IPAddress"=hex(7):31,00,30,00,2e,00,32,00,2e,00,30,00,2e,00,31,00,30,00,3
1,00,\
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):31,00,30,00,2e,00,32,00,2e,00,30,00,2e,00,32,00,35
,00,\
33,00,00,00,00,00
"DefaultGatewayMetric"=hex(7):30,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Int
erfaces\{065f0c42-703a-11de-9954-806e6f6e6963}]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Int
erfaces\{170923B8-287F-4AA5-9B62-ACC3DF721737}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000001
"NameServer"=""
"Domain"=""
"RegistrationEnabled"=dword:00000001
"RegisterAdapterName"=dword:00000000
"DhcpIPAddress"="0.0.0.0"
"DhcpSubnetMask"="255.0.0.0"
```

```
"DhcpServer"="255.255.255.255"
"Lease"=dword:00000000
"LeaseObtainedTime"=dword:00000000
"T1"=dword:00000000
"T2"=dword:00000000
"LeaseTerminatesTime"=dword:00000000
"AddressType"=dword:00000000
"IsServerNapAware"=dword:00000000
"DhcpConnForceBroadcastFlag"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Int
erfaces\{1F5668B7-A26C-483F-B719-EF9338C9C4DC}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000001
"NameServer"=""
"Domain"=""
"RegistrationEnabled"=dword:00000001
"RegisterAdapterName"=dword:00000000
"DhcpServer"="10.1.1.250"
"Lease"=dword:00015180
"LeaseObtainedTime"=dword:4c487c64
"T1"=dword:4c492524
"T2"=dword:4c49a3b4
"LeaseTerminatesTime"=dword:4c49cde4
"AddressType"=dword:00000000
"IsServerNapAware"=dword:00000000
"DhcpConnForceBroadcastFlag"=dword:00000001
"IPAddress"=hex(7):00,00
"SubnetMask"=hex(7):00,00
"DhcpIPAddress"="10.1.1.97"
"DhcpSubnetMask"="255.255.0.0"
"DhcpInterfaceOptions"=hex:06,00,00,00,00,00,00,00,08,00,00,00,00,00,00,00,
e4,\
cd,49,4c,09,00,07,01,09,00,06,0b,03,00,00,00,00,00,00,04,00,00,00,00,00,\
00,00,e4,cd,49,4c,0a,01,01,fa,01,00,00,00,00,00,00,00,04,00,00,00,00,00,\
00,e4,cd,49,4c,ff,ff,00,00,36,00,00,00,00,00,00,00,04,00,00,00,00,00,\
e4,cd,49,4c,0a,01,01,fa,35,00,00,00,00,00,00,00,01,00,00,00,00,00,00,e4,\
cd,49,4c,05,00,00,00,fc,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,67,7c,\
48,4c,33,00,00,00,00,00,00,00,04,00,00,00,00,00,00,e4,cd,49,4c,00,01,51,\
80
"DhcpGatewayHardware"=hex:0a,01,01,fa,06,00,00,00,00,15,17,bd,e4,fd
"DhcpGatewayHardwareCount"=dword:00000001
"DhcpNameServer"="9.0.7.1 9.0.6.11"
"DhcpDefaultGateway"=hex(7):31,00,30,00,2e,00,31,00,2e,00,31,00,2e,00,32,0
0,35,\
00,30,00,00,00,00,00
"DhcpSubnetMaskOpt"=hex(7):32,00,35,00,35,00,2e,00,32,00,35,00,35,00,2e,00
,30,\
00,2e,00,30,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Int
erfaces\{511b57fe-b4d6-4121-9d12-c4e2ba398e58}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000001
"NameServer"=""
"Domain"=""
"RegistrationEnabled"=dword:00000001
"RegisterAdapterName"=dword:00000000
"DhcpIPAddress"="0.0.0.0"
"DhcpSubnetMask"="255.0.0.0"
"DhcpServer"="255.255.255.255"
"Lease"=dword:00000000
"LeaseObtainedTime"=dword:00000000
"T1"=dword:00000000
"T2"=dword:00000000
"LeaseTerminatesTime"=dword:00000000
"AddressType"=dword:00000000
"IsServerNapAware"=dword:00000000
"DhcpConnForceBroadcastFlag"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Int
erfaces\{62c84429-9356-407b-9572-b798c6c8ac9a}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000000
"NameServer"=""
"Domain"=""
"RegistrationEnabled"=dword:00000001
"RegisterAdapterName"=dword:00000000
"DhcpServer"="255.255.255.255"
"Lease"=dword:00000000
"LeaseObtainedTime"=dword:00000000
"T1"=dword:00000000
"T2"=dword:00000000
"LeaseTerminatesTime"=dword:00000000
"AddressType"=dword:00000000
"IsServerNapAware"=dword:00000000
"DhcpConnForceBroadcastFlag"=dword:00000000
"IPAddress"=hex(7):31,00,30,00,2e,00,32,00,2e,00,31,00,2e,00,32,00,00,00,00,
```

```
0,00
"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

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Int
erfaces\{724248B6-966D-41C0-8E01-F7C08D0E8648}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000001
"NameServer"=""
"Domain"=""
"RegistrationEnabled"=dword:00000001
"RegisterAdapterName"=dword:00000000
"DhcpIPAddress"="0.0.0.0"
"DhcpSubnetMask"="255.0.0.0"
"DhcpServer"="255.255.255.255"
"Lease"=dword:00000000
"LeaseObtainedTime"=dword:00000000
"T1"=dword:00000000
"T2"=dword:00000000
"LeaseTerminatesTime"=dword:00000000
"AddressType"=dword:00000000
"IsServerNapAware"=dword:00000000
"DhcpConnForceBroadcastFlag"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Int
erfaces\{72E75E72-4FF7-4DE0-A81E-B4721EA039EE}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000000
"NameServer"=""
"Domain"=""
"RegistrationEnabled"=dword:00000001
"RegisterAdapterName"=dword:00000000
"DhcpServer"="255.255.255.255"
"Lease"=dword:00000000
"LeaseObtainedTime"=dword:00000000
"T1"=dword:00000000
"T2"=dword:00000000
"LeaseTerminatesTime"=dword:00000000
"AddressType"=dword:00000000
"IsServerNapAware"=dword:00000000
"DhcpConnForceBroadcastFlag"=dword:00000000
"IPAddress"=hex(7):31,00,30,00,2e,00,33,00,2e,00,31,00,2e,00,32,00,00,00,0
0,00
"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

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Int
erfaces\{7C1C96DA-E1EC-4771-A1BC-B107F1AD3B8E}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000001
"NameServer"=""
"Domain"=""
"RegistrationEnabled"=dword:00000001
"RegisterAdapterName"=dword:00000000
"DhcpIPAddress"="10.1.1.11"
"DhcpSubnetMask"="255.255.0.0"
"DhcpServer"="10.1.1.250"
"Lease"=dword:00015180
"LeaseObtainedTime"=dword:4bfab4e4
"T1"=dword:4bf5da4
"T2"=dword:4bfbc34
"LeaseTerminatesTime"=dword:4bfc0664
"AddressType"=dword:00000000
"IsServerNapAware"=dword:00000000
"DhcpConnForceBroadcastFlag"=dword:00000001
"DhcpInterfaceOptions"=hex:03,00,00,00,00,00,00,00,04,00,00,00,00,00,00,00,
64,\
06,fc,4b,0a,01,01,fa,01,00,00,00,00,00,00,00,04,00,00,00,00,00,00,00,00,64,06
,\
fc,4b,ff,ff,00,00,36,00,00,00,00,00,00,00,04,00,00,00,00,00,00,00,00,00,64,06,fc
,\
4b,0a,01,01,fa,35,00,00,00,00,00,00,00,01,00,00,00,00,00,00,00,00,00,64,06,fc,4b
,\
05,00,00,00,fc,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,e7,b4,fa,4b,33
,\
00,00,00,00,00,00,04,00,00,00,00,00,00,00,00,00,64,06,fc,4b,00,01,51,80
"DhcpGatewayHardware"=hex:0a,01,01,fa,06,00,00,00,00,00,15,17,bd,e4,fd
"DhcpGatewayHardwareCount"=dword:00000001
"DhcpDefaultGateway"=hex(7):31,00,30,00,2e,00,31,00,2e,00,31,00,2e,00,32,0
0,35,\
00,30,00,00,00,00,00
"DhcpSubnetMaskOpt"=hex(7):32,00,35,00,35,00,2e,00,32,00,35,00,35,00,2e,00,
30,\
00,2e,00,30,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Int
erfaces\{C7B5C48B-7281-4F2B-B106-6DAE0493EE4D}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000001
"NameServer"=""
"Domain"=""
"RegistrationEnabled"=dword:00000001
"RegisterAdapterName"=dword:00000000
```

```
"DhcpIPAddress"="0.0.0.0"
"DhcpSubnetMask"="255.0.0.0"
"DhcpServer"="255.255.255.255"
"Lease"=dword:00000000
"LeaseObtainedTime"=dword:00000000
"T1"=dword:00000000
"T2"=dword:00000000
"LeaseTerminatesTime"=dword:00000000
"AddressType"=dword:00000000
"IsServerNapAware"=dword:00000000
"DhcpConnForceBroadcastFlag"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Int
erfaces\{D0264C87-0470-4A06-A0D7-63EF81968E35}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000001
"NameServer"=""
"Domain"=""
"RegistrationEnabled"=dword:00000001
"RegisterAdapterName"=dword:00000000
"DhcpServer"="255.255.255.255"
"Lease"=dword:00000000
"LeaseObtainedTime"=dword:00000000
"T1"=dword:00000000
"T2"=dword:00000000
"LeaseTerminatesTime"=dword:00000000
"AddressType"=dword:00000000
"IsServerNapAware"=dword:00000000
"DhcpConnForceBroadcastFlag"=dword:00000000
"IPAddress"=hex(7):00,00
"SubnetMask"=hex(7):00,00
"DhcpIPAddress"="0.0.0.0"
"DhcpSubnetMask"="255.0.0.0"

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Int
erfaces\{D7C8DBF2-6FC6-4873-9467-8A0637B1D07A}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000001
"NameServer"=""
"Domain"=""
"RegistrationEnabled"=dword:00000001
"RegisterAdapterName"=dword:00000000
"DhcpIPAddress"="0.0.0.0"
"DhcpSubnetMask"="255.0.0.0"
"DhcpServer"="255.255.255.255"
"Lease"=dword:00000000
"LeaseObtainedTime"=dword:00000000
"T1"=dword:00000000
"T2"=dword:00000000
"LeaseTerminatesTime"=dword:00000000
"AddressType"=dword:00000000
"IsServerNapAware"=dword:00000000
"DhcpConnForceBroadcastFlag"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Int
erfaces\{DF3B01C1-9D76-427F-BA7A-17AEAF182E30}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000000
"NameServer"="9.3.192.21"
"Domain"=""
"RegistrationEnabled"=dword:00000001
"RegisterAdapterName"=dword:00000000
"DhcpServer"="255.255.255.255"
"Lease"=dword:00000000
"LeaseObtainedTime"=dword:00000000
"T1"=dword:00000000
"T2"=dword:00000000
"LeaseTerminatesTime"=dword:00000000
"AddressType"=dword:00000000
"IsServerNapAware"=dword:00000000
"DhcpConnForceBroadcastFlag"=dword:00000000
"IPAddress"=hex(7):31,00,30,00,2e,00,32,00,2e,00,31,00,2e,00,31,00,30,00,3
1,00,\
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

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Int
erfaces\{F7E95255-579D-4EF7-9A8B-20B4C40B7C48}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000000
"NameServer"=""
"Domain"=""
"RegistrationEnabled"=dword:00000001
"RegisterAdapterName"=dword:00000000
"DhcpServer"="255.255.255.255"
"Lease"=dword:00000000
"LeaseObtainedTime"=dword:00000000
"T1"=dword:00000000
"T2"=dword:00000000
"LeaseTerminatesTime"=dword:00000000
"AddressType"=dword:00000000
"IsServerNapAware"=dword:00000000
"DhcpConnForceBroadcastFlag"=dword:00000000
"IPAddress"=hex(7):31,00,39,00,32,00,2e,00,32,00,2e,00,31,00,2e,00,32,00,0
0,00,\
```

```

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
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Per
sistentRoutes]
"0.0.0.0,0.0.0.0,10.2.0.253,-1"=""

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Win
sock]
"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:08,00,00,00,03,00,00,00,02,00,00,00,01,00,00,00,06,00,00,00,
02,\
00,00,00,01,00,00,00,00,00,00,00,02,00,00,00,00,00,00,00,06,00,00,02,00
,\
00,00,02,00,00,00,11,00,00,00,02,00,00,00,02,00,00,00,00,00,00,02,00,00
,\
00,00,00,00,00,11,00,00,00,02,00,00,00,03,00,00,00,ff,00,00,00,02,00,00
,\
03,00,00,00,00,00,00
"UseDelayedAcceptance"=dword:00000000

```

10.2.3. tpcc_com_settings.reg

Windows Registry Editor Version 5.00

```

[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\TPCC]
"dbName"="tpcc"
"dbPassword"="tpcc"
"dbType"="DB2"
"dbUserName"="tpcc"
"dlyvLogPath"="c:\\inetpub\\wwwroot\\tpcc\\"
"dlyvQueueLen"=dword:00004e20
"dlyvThreads"=dword:00000016
"errorLogFile"="c:\\inetpub\\wwwroot\\tpcc\\isapi_err.log"
"htmlTrace"=dword:00000000
"htmlTraceLogFile"="c:\\inetpub\\wwwroot\\tpcc\\isapi.log"
"isapi_trace"=dword:00000000
>nullDB"=dword:00000000
"numServers"=dword:00000001
"numUsers"=dword:0004e200
"numWarehouse"=dword:00007530
"numConnections"=dword:000000b4

```

10.2.4. tpcc_com_parameters.txt

```

Transactions not supported
Object pooling enabled
Minimum Pool Size 120
Maximum Pool Size 120
Creation Timeout 1,900,000,000
Enable Just In Time Activation
Concurrency Required

```

10.3. Linux KVM Host Parameters

10.3.1. /proc/cmdline

```

ro root=/dev/mapper/vg_turnpike-lv_root rd_NO_LUKS LANG=en_US.UTF-8
rd_LVM_LV=v_g_turnpike/lv_swap rd_NO_MD SYSFONT=lataarcyrbh-sun16
crashkernel=177M@0M KEYBOARDTYPE=pc KEYSOURCE=us rd_NO_DM
rd_LVM_LV=v_g_turnpike/lv_root
cgrou_p_disable=blkio,cpu,cpuacct,cpuset,devices,freezer,memory,net_cls
iommu=pt intel_iommu=on console=tty0 console=ttyS0,115200 hugepages=382456

```

10.3.2. setup.sh

```

#!/bin/bash

# in /etc/modprobe/kvm.conf
# options kvm_intel ple_gap=0

function modify-file {
    VALUE=$1
    shift

    for FILE in $@; do
        echo

        echo "FILE=$FILE"
        echo "VALUE=$VALUE"
    done
}

```

```

echo "File to be modified: ${FILE}"
echo "Current Value: `cat ${FILE}`"
echo "Modifying..."
echo "${VALUE}" > ${FILE}
echo "New Value: `cat ${FILE}`"
done
}

function set-sched {
    scheduler=$1
    shift
    device_list=$*

    if [ -z "$scheduler" ]; then
        echo "USAGE: $(basename $0) <anticipatory|cfq|deadline|noop>
[device]+"
        return
    fi

    case $scheduler in
        anticipatory|cfq|deadline|noop
        *)
            echo "USAGE: $(basename $0) <anticipatory|cfq|deadline|noop>
[device]+"
            return
            ;;
    esac

    if [ -z "$device_list" ]; then
        device_list=$(ls -l /sys/block)
    fi

    for device in $device_list; do
        if [ ! -d /sys/block/$device ]; then
            device=$(dev_to_sysfs_name $device)
            if [ $? -ne 0 ]; then
                continue
            fi
        fi

        if [ -e /sys/block/$device/queue/scheduler ]; then
            echo $scheduler > /sys/block/$device/queue/scheduler
        fi
    done
}

function set-read-ahead {
    read_ahead=$1
    shift
    device_list=$*

    if [ -z "$read_ahead" ]; then
        echo "USAGE: $(basename $0) <read_ahead_kb> [device]+"
        return
    fi

    if [ -z "$device_list" ]; then
        device_list=$(ls -l /sys/block)
    fi

    for device in $device_list; do
        if [ ! -d /sys/block/$device ]; then
            device=$(dev_to_sysfs_name $device)
            if [ $? -ne 0 ]; then
                continue
            fi
        fi

        if [ -e /sys/block/$device/queue/read_ahead_kb ]; then
            echo $read_ahead > /sys/block/$device/queue/read_ahead_kb
        fi
    done
}

function pin_irq {
    int=$1
    CPU=$2
    last_int=$3

    bit_mask=( 1 2 4 8 10 20 40 80 100 200 400 800 1000 2000 4000 8000
10000 20000 40000 80000 100000 200000 400000 800000 1000000 2000000
4000000 8000000 10000000 20000000 40000000 80000000 )

    let segments=$CPU/32
    let mask_index=$CPU-$segments*32

    for i in `seq 1 $segments`; do
        mask_segment="00000000$mask_segment"
    done
    cpu_mask="${bit_mask[$mask_index]}$mask_segment"

    echo "pin_irq: $cpu_mask >/proc/irq/$int/smp_affinity"
    echo $cpu_mask >/proc/irq/$int/smp_affinity

    if [ ! -z $last_int ]; then
        for i in `seq $int $last_int`; do
            echo "pin_irq: $cpu_mask >/proc/irq/$i/smp_affinity"
            echo $cpu_mask >/proc/irq/$i/smp_affinity
        done
    fi
}

```

```
set-sched deadline
set-read-ahead 512
modify-file 0 /proc/sys/vm/zone_reclaim_mode
modify-file 0 /proc/sys/vm/swappiness
service irqbalance stop
```

```
fc1_irq=116
fc2_irq=117
fc3_irq=118
fc4_irq=119
fc5_irq=120
fc6_irq=121
fc7_irq=165
eth1_irq=166
```

```
pin_irq ${fc1_irq} 0
pin_irq ${fc2_irq} 1
pin_irq ${fc3_irq} 2
pin_irq ${fc4_irq} 8
pin_irq ${fc5_irq} 9
pin_irq ${fc6_irq} 10
pin_irq ${fc7_irq} 11
pin_irq ${eth1_irq} 3
```

10.3.3. /etc/sysctl.conf

```
# Kernel sysctl configuration file for Red Hat Linux
#
# For binary values, 0 is disabled, 1 is enabled. See sysctl(8) and
# sysctl.conf(5) for more details.

# Controls IP packet forwarding
net.ipv4.ip_forward = 0

# Controls source route verification
net.ipv4.conf.default.rp_filter = 1

# Do not accept source routing
net.ipv4.conf.default.accept_source_route = 0

# Controls the System Request debugging functionality of the kernel
kernel.sysrq = 0

# Controls whether core dumps will append the PID to the core filename.
# Useful for debugging multi-threaded applications.
kernel.core_uses_pid = 1

# Controls the use of TCP syncookies
net.ipv4.tcp_syncookies = 1

# Disable netfilter on bridges.
net.bridge.bridge-nf-call-ip6tables = 0
net.bridge.bridge-nf-call-iptables = 0
net.bridge.bridge-nf-call-arptables = 0

# Controls the default maximum size of a message queue
kernel.msgmnb = 65536

# Controls the maximum size of a message, in bytes
kernel.msgmax = 65536

# Controls the maximum shared segment size, in bytes
kernel.shmmax = 406410948608

# Controls the maximum number of shared memory segments, in pages
kernel.shmall = 99221423
kernel.sem = 500 512000 64 2048
kernel.msgmni = 4096
fs.file-max = 524288
#net.core.rmem_max = 131071
#net.core.wmem_max = 131071
vm.hugetlb_shm_group = 12090
net.core.rmem_max = 1048576
net.core.wmem_max = 1048576
```

10.3.4. /proc/version

```
Linux version 2.6.32-348.el6.x86_64 (mockbuild@x86-022.build.eng.bos.redhat.com) (gcc version 4.4.7 20120313 (Red Hat 4.4.7-3) (GCC) ) #1 SMP Mon Dec 10 12:48:03 EST 2012
```

10.3.5. Libvirt Guest XML configuration

```
<domain type='kvm' id='1'>
  <name>turnpike-guest</name>
  <uuid>9f4d65a0-3dca-67d8-7c1d-cac1654fa701</uuid>
  <memory unit='KiB'>783269888</memory>
  <currentMemory unit='KiB'>783269888</currentMemory>
  <memoryBacking>
    <hugepages/>
  </memoryBacking>
  <vcpu placement='static'>32</vcpu>
  <cpu>
    <vcpupin vcpu='0' cpuset='0' />
    <vcpupin vcpu='1' cpuset='16' />
    <vcpupin vcpu='2' cpuset='1' />
    <vcpupin vcpu='3' cpuset='17' />
    <vcpupin vcpu='4' cpuset='2' />
    <vcpupin vcpu='5' cpuset='18' />
    <vcpupin vcpu='6' cpuset='3' />
    <vcpupin vcpu='7' cpuset='19' />
```

```
<vcpupin vcpu='8' cpuset='4' />
<vcpupin vcpu='9' cpuset='20' />
<vcpupin vcpu='10' cpuset='5' />
<vcpupin vcpu='11' cpuset='21' />
<vcpupin vcpu='12' cpuset='6' />
<vcpupin vcpu='13' cpuset='22' />
<vcpupin vcpu='14' cpuset='7' />
<vcpupin vcpu='15' cpuset='23' />
<vcpupin vcpu='16' cpuset='8' />
<vcpupin vcpu='17' cpuset='24' />
<vcpupin vcpu='18' cpuset='9' />
<vcpupin vcpu='19' cpuset='25' />
<vcpupin vcpu='20' cpuset='10' />
<vcpupin vcpu='21' cpuset='26' />
<vcpupin vcpu='22' cpuset='11' />
<vcpupin vcpu='23' cpuset='27' />
<vcpupin vcpu='24' cpuset='12' />
<vcpupin vcpu='25' cpuset='28' />
<vcpupin vcpu='26' cpuset='13' />
<vcpupin vcpu='27' cpuset='29' />
<vcpupin vcpu='28' cpuset='14' />
<vcpupin vcpu='29' cpuset='30' />
<vcpupin vcpu='30' cpuset='15' />
<vcpupin vcpu='31' cpuset='31' />
</cpu>
<numatune>
  <memory mode='preferred' nodeset='0' />
</numatune>
<os>
  <type arch='x86_64' machine='rhel6.4.0'>hvm</type>
  <boot dev='hd' />
</os>
<features>
  <acpi />
  <apic />
  <paex />
</features>
<cpu mode='custom' match='exact'>
  <model fallback='allow'>SandyBridge</model>
  <vendor>Intel</vendor>
  <topology sockets='2' cores='8' threads='2' />
  <feature policy='require' name='vme' />
  <feature policy='require' name='tm2' />
  <feature policy='require' name='est' />
  <feature policy='require' name='vmx' />
  <feature policy='require' name='osxsave' />
  <feature policy='require' name='smx' />
  <feature policy='require' name='ss' />
  <feature policy='require' name='ds' />
  <feature policy='require' name='tsc-deadline' />
  <feature policy='require' name='dtes64' />
  <feature policy='require' name='ht' />
  <feature policy='require' name='dca' />
  <feature policy='require' name='pbe' />
  <feature policy='require' name='tm' />
  <feature policy='require' name='pdcml' />
  <feature policy='require' name='pdpelgb' />
  <feature policy='require' name='ds_cpl' />
  <feature policy='require' name='xtpr' />
  <feature policy='require' name='acpi' />
  <feature policy='require' name='monitor' />
</cpu>
<clock offset='utc' />
<on_poweroff>destroy</on_poweroff>
<on_reboot>restart</on_reboot>
<on_crash>restart</on_crash>
<devices>
  <emulator>/usr/libexec/qemu-kvm</emulator>
  <disk type='block' device='disk'>
    <driver name='qemu' type='raw' cache='none' io='native' />
    <source dev='/dev/guests/turnpike-guest' />
    <target dev='vda' bus='virtio' />
    <alias name='virtio-disk0' />
    <address type='pci' domain='0x0000' bus='0x00' slot='0x04'
function='0x0' />
  </disk>
  <controller type='usb' index='0'>
    <alias name='usb0' />
    <address type='pci' domain='0x0000' bus='0x00' slot='0x01'
function='0x2' />
  </controller>
  <interface type='bridge'>
    <mac address='52:54:00:d7:04:2f' />
    <source bridge='br0' />
    <target dev='vnet0' />
    <model type='virtio' />
    <alias name='net0' />
    <address type='pci' domain='0x0000' bus='0x00' slot='0x03'
function='0x0' />
  </interface>
  <serial type='pty'>
    <source path='/dev/pts/3' />
    <target port='0' />
    <alias name='serial0' />
  </serial>
  <console type='pty' tty='/dev/pts/3'>
    <source path='/dev/pts/3' />
    <target type='serial' port='0' />
    <alias name='serial0' />
```

```

</console>
<input type='tablet' bus='usb'>
  <alias name='input0' />
</input>
<input type='mouse' bus='ps2' />
<graphics type='vnc' port='5900' autoport='yes' listen='127.0.0.1'>
  <listen type='address' address='127.0.0.1' />
</graphics>
<video>
<model type='cirrus' vram='9216' heads='1' />
<alias name='video0' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x02'
function='0x0' />
</video>
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0x06' slot='0x00' function='0x2' />
  </source>
  <alias name='hostdev0' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x06'
function='0x0' />
</hostdev>
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0x1b' slot='0x00' function='0x0' />
  </source>
  <alias name='hostdev1' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x07'
function='0x0' />
</hostdev>
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0x1b' slot='0x00' function='0x1' />
  </source>
  <alias name='hostdev2' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x08'
function='0x0' />
</hostdev>
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0x20' slot='0x00' function='0x0' />
  </source>
  <alias name='hostdev3' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x09'
function='0x0' />
</hostdev>
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0x86' slot='0x00' function='0x0' />
  </source>
  <alias name='hostdev4' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x0a'
function='0x0' />
</hostdev>
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0x86' slot='0x00' function='0x1' />
  </source>
  <alias name='hostdev5' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x0b'
function='0x0' />
</hostdev>
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0x8b' slot='0x00' function='0x0' />
  </source>
  <alias name='hostdev6' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x0c'
function='0x0' />
</hostdev>
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0x0000' bus='0x8b' slot='0x00' function='0x1' />
  </source>
  <alias name='hostdev7' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x0d'
function='0x0' />
</hostdev>
<memballoon model='virtio'>
  <alias name='balloon0' />
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05'
function='0x0' />
</memballoon>
</devices>
<seclabel type='dynamic' model='selinux' relabel='yes'>
<label>system_u:system_r:svirt_t:s0:c247,c828</label>
<imagelabel>system_u:object_r:svirt_image_t:s0:c247,c828</imagelabel>
</seclabel>
</domain>

```

10.4. Linux KVM Guest Parameters

10.4.1. /proc/cmdline

```

ro root=/dev/mapper/vg_turnpikequest-lv_root rd_NO_LUKS LANG=en_US.UTF-8
rd_LVM_LV=vq_turnpikequest/lv_swap rd_NO_MD
rd_LVM_LV=vq_turnpikequest/lv_root SYSFONT=latarcyrheb-sun16
crashkernel=175M@0M KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM
cgroup_disable=blkio,cpu,cpuacct,cpuset,devices,freezer,memory,net_cls

```

```

hugepages=375046 nodelayacct console=tty0 console=ttyS0,115200
clocksource=tsc nohz=off highres=off

```

10.4.2. setup.sh

```

#!/bin/bash

# in /etc/modprobe/tpcc.conf:
# options igb IntMode=2 RSS=1 QueuePairs=1 InterruptThrottleRate=1000

function modify-file {
  VALUE=$1
  shift

  for FILE in $@; do
    echo

    echo "FILE=$FILE"
    echo "VALUE=$VALUE"

    echo "File to be modified: ${FILE}"
    echo "Current Value: `cat ${FILE}`"
    echo "Modifying..."
    echo "${VALUE}" > ${FILE}
    echo "New Value: `cat ${FILE}`"
  done
}

function zio {
  if pushd /sys/class/scsi_host > /dev/null; then
    for dir in `ls -d *`; do
      if [ -f "$dir/zio" ]; then
        echo $dir
        read port_name < $dir/device/fc_host/$dir/port_name
        if [ "$port_name" != "0x21000024ff428685" ]; then
          cat $dir/zio
          cat $dir/zio_timer
          echo 6 > $dir/zio
          cat $dir/zio
          cat $dir/zio_timer
          else
            echo "skipping log device"
            cat $dir/zio
            cat $dir/zio_timer
          fi
        echo
      fi
    done
  else
    echo "push failed"
    return
  fi
}

function set-sched {
  scheduler=$1
  shift
  device_list=$*

  if [ -z "$scheduler" ]; then
    echo "USAGE: $(basename $0) <anticipatory|cfq|deadline|noop>
[device]+"
    return
  fi

  case $scheduler in
    anticipatory|cfq|deadline|noop)
      ;;
    *)
      echo "USAGE: $(basename $0) <anticipatory|cfq|deadline|noop>
[device]+"
      return
      ;;
  esac

  if [ -z "$device_list" ]; then
    device_list=$(ls -l /sys/block)
  fi

  for device in $device_list; do
    if [ ! -d /sys/block/$device ]; then
      device=$(dev_to_sysfs_name $device)
      if [ $? -ne 0 ]; then
        continue
      fi
    fi

    if [ -e /sys/block/$device/queue/scheduler ]; then
      echo $scheduler > /sys/block/$device/queue/scheduler
    fi
  done
}

function set-read-ahead {
  read_ahead=$1
  shift
  device_list=$*

  if [ -z "$read_ahead" ]; then
    echo "USAGE: $(basename $0) <read_ahead_kb> [device]+"
    return
  fi
}

```

```

if [ -z "$device_list" ]; then
    device_list=$(ls -l /sys/block)
fi

for device in $device_list; do
    if [ ! -d /sys/block/$device ]; then
        device=$(dev_to_sysfs_name $device)
        if [ $? -ne 0 ]; then
            continue
        fi
    fi

    if [ -e /sys/block/$device/queue/read_ahead_kb ]; then
        echo $read_ahead > /sys/block/$device/queue/read_ahead_kb
    fi
done

function pin_irq {
    int=$1
    CPU=$2
    last_int=$3

    bit_mask=( 1 2 4 8 10 20 40 80 100 200 400 800 1000 2000 4000 8000
10000 20000 40000 80000 100000 200000 400000 800000 1000000 2000000
4000000 8000000 10000000 20000000 40000000 80000000 )

    let segments=$CPU/32
    let mask_index=$CPU-$segments*32

    for i in `seq 1 $segments`; do
        mask_segment="00000000$mask_index"
    done
    cpu_mask="${bit_mask[$mask_index]}$mask_segment"

    echo "pin_irq: $cpu_mask >/proc/irq/$int/smp_affinity"
    echo $cpu_mask >/proc/irq/$int/smp_affinity

    if [ ! -z $last_int ]; then
        for i in `seq $int $last_int`; do
            echo "pin_irq: $cpu_mask >/proc/irq/$i/smp_affinity"
            echo $cpu_mask >/proc/irq/$i/smp_affinity
        done
    fi
}

function tune_logwriter {
    logwriter_priority_level=$1
    logwriter_smp_affinity=$2

    logwriter_pid=`su - tpcc -c "db2pd -edus" | grep db2loggw | sed -e
's/^.\{,25\}\{[0-9]\+\}\s.*\1/'`

    echo -n "Executing chrt -f -p $logwriter_priority_level on db2loggw
process (PID=${logwriter_pid})..."
    if chrt -f -p $logwriter_priority_level $logwriter_pid; then
        echo "success."
    else
        echo "failed"
    fi

    echo -n "Executing taskset -p $logwriter_smp_affinity
$logwriter_pid..."
    if taskset -p $logwriter_smp_affinity $logwriter_pid; then
        echo "success."
    else
        echo "failed."
    fi
}

function tune_pagecleaners {
    echo "Executing chrt -f -p 0; renice -n -1 on db2pclnr processes (page
cleaners)."
```

```

fc2_irq=27
fc3_irq=29
fc4_irq=31
fc5_irq=33
fc6_irq=35
fc7_irq=37
eth1_irq=44
```

```

pin_irq ${fc1_irq} 0
pin_irq ${fc2_irq} 2
pin_irq ${fc3_irq} 4
pin_irq ${fc4_irq} 16
pin_irq ${fc5_irq} 18
pin_irq ${fc6_irq} 20
pin_irq ${fc7_irq} 22
pin_irq ${eth1_irq} 6
```

```

tune_logwriter 97 0x16
tune_pagecleaners
```

10.4.3. tpcc.pm

```

package tpcc;

use strict;
use warnings;

BEGIN {
    use Exporter();
    our (@ISA, @EXPORT);
    @ISA = "Exporter";
    @EXPORT = qw( &get_device_list
&get_backup_device_list );
}

sub get_device_list {
    my @list;

    if (open(INPUT, "/autobench/sources/db2_tpcc-1/users/tpcc/tpc-
c.ibm/db-devices.txt")) {
        while (<INPUT>) {
            chomp $_;

            push @list, "/dev/disk/by-id/" . $_;
        }
    } else {
        print STDERR "ERROR: Failed to open device list\n";
    }

    return @list;
}

sub get_backup_device_list {
    my @list;

    if (open(INPUT, "/autobench/sources/db2_tpcc-1/users/tpcc/tpc-
c.ibm/backup-devices.txt")) {
        while (<INPUT>) {
            chomp $_;

            push @list, "/dev/disk/by-id/" . $_;
        }
    } else {
        print STDERR "ERROR: Failed to open backup device list\n";
    }

    return @list;
}

END { }

1;
```

10.4.4. create-device-mapping.pl

```

#!/usr/bin/perl

use tpcc;
use Data::Dumper;

use strict;
use warnings;

my @devices = get_device_list();

if (-d "/dev/tpcc") {
    unlink </dev/tpcc/*>;
    rmdir "/dev/tpcc";
}

mkdir "/dev/tpcc";

my %tablespaces_partitions = ( 'W' => 1, 'D' => 2, 'I' => 3, 'S' => 5, 'C'
=> 6, 'H' => 7, 'O' => 8, 'OL' => 9, 'N' => 10, 'O2' => 11, 'C2' => 12 );

my @tablespaces = ('W', 'D', 'S', 'C', 'H', 'O', 'OL', 'N', 'O2', 'C2');

#print Dumper \%tablespaces_partitions;
#print Dumper \@tablespaces;

for (my $i=0; $i<@devices; $i++) {
```



```

for (my $x=0; $x<@tablespaces; $x++) {
    symlink($devices[$i] . "-part" .
$tablespaces_partitions{$tablespaces[$x]}, "/dev/tpcc/" . $tablespaces[$x]
. "-" . sprintf("%03d", $i+1));
    chmod(0777, $devices[$i] . "-part" .
$tablespaces_partitions{$tablespaces[$x]});
}
}

symlink($devices[0] . "-part" . $tablespaces_partitions{'I'},
"/dev/tpcc/I");
chmod(0777, $devices[0] . "-part" . $tablespaces_partitions{'I'});

symlink("/dev/disk/by-id/scsi-360080e5000236208000005d5506c0368",
"/dev/tpcc/log");
chmod(0777, "/dev/disk/by-id/scsi-360080e5000236208000005d5506c0368");

```

10.4.5. db-devices.txt

```

scsi-360080e500024d43e00000667506ea73a
scsi-360080e5000248e9a00000376506ef474
scsi-360080e5000248b9000000d80506edc97
scsi-360080e500024e1320000044b506eb9e8
scsi-360080e500024f70200000c32506ed2da
scsi-360080e5000247aac00001aaf51111288
scsi-360080e50002d045600000415506ecd8b
scsi-360080e500024d55400000a17506ed5af
scsi-360080e500024dc4a00000d6e506eb4e
scsi-360080e500024d43e0000066a506ea76e
scsi-360080e5000248e9a00000379506ef499
scsi-360080e5000248b9000000d86506edcf4
scsi-360080e500024e1320000044d506eba16
scsi-360080e500024f70200000c35506ed2fc
scsi-360080e5000247aac00000ba95072a72b
scsi-360080e50002d045600000418506ecdb8
scsi-360080e500024d55400000a1a506ed5d1
scsi-360080e500024dc4a00000d71506eb78
scsi-360080e500024d43e0000066d506ea7a7
scsi-360080e5000248e9a0000037c506ef4bc
scsi-360080e5000248b9000000d83506edcc5
scsi-360080e500024e1320000044f506eba3e
scsi-360080e500024f70200000c38506ed32d
scsi-360080e5000247aac00000bac5072a74d
scsi-360080e50002d04560000041b506ecdec
scsi-360080e500024d55400000a1d506ed5fb
scsi-360080e500024dc4a00000d74506eb9f
scsi-360080e500024d43e00000670506ea7ce
scsi-360080e5000248e9a0000037f506ef4da
scsi-360080e5000248b9000000d89506edd22
scsi-360080e500024e13200000451506eba76
scsi-360080e500024f70200000c3b506ed355
scsi-360080e5000247aac00000baf5072a778
scsi-360080e50002d04560000041e506ece19
scsi-360080e500024d55400000a20506ed622
scsi-360080e500024dc4a00000d77506ebc8
scsi-360080e500024d72c0000135250f94a8b
scsi-360080e500024a4600000030e506ee6b9
scsi-360080e500024d3f000000d2d506ee986
scsi-360080e500024de5800000419506eacc2
scsi-360080e50002d011200000b8a506ed3cf
scsi-360080e500024d60a00000b635072aa08
scsi-360080e500024a77000000402506edc4a
scsi-360080e500024d55400000a29506ed6b7
scsi-360080e500024dd1400000cef506edf0c
scsi-360080e500024d72c000005ed506ea9f4
scsi-360080e500024a46000000310506ee6da
scsi-360080e500024d3f000000d2f506ee9a8
scsi-360080e500024de5800000416506eac98
scsi-360080e50002d011200000b8c506ed3f1
scsi-360080e500024d60a00000b655072aa2f
scsi-360080e500024a77000000400506edc21
scsi-360080e50002d040c000009bc506ed68b
scsi-360080e500024dd1400000cf1506edf2c
scsi-360080e500024d72c000005ef506eaalf
scsi-360080e500024a46000000312506ee6f8
scsi-360080e500024d3f000000d31506ee9c8
scsi-360080e500024de580000041c506eace7
scsi-360080e50002d011200000b8e506ed414
scsi-360080e500024d60a00000b675072aa4f
scsi-360080e500024a77000000404506edc74
scsi-360080e50002d040c000009be506ed6c1
scsi-360080e500024dd1400000cf3506edf4f
scsi-360080e500024d72c000005f1506eaa6d
scsi-360080e500024a46000000314506ee71b
scsi-360080e500024d3f000000d33506ee9ee
scsi-360080e500024de580000041f506ead11
scsi-360080e50002d011200000b90506ed435
scsi-360080e500024d60a00000b695072aa78
scsi-360080e500024a77000000406506edc9d
scsi-360080e50002d040c000009c0506ed6e8
scsi-360080e500024dd1400000cf5506edf72

```

10.4.6. /etc/sysctl.conf

```

# Kernel sysctl configuration file for Red Hat Linux
#
# For binary values, 0 is disabled, 1 is enabled. See sysctl(8) and
# sysctl.conf(5) for more details.

# Controls IP packet forwarding
net.ipv4.ip_forward = 0

```

```

# Controls source route verification
net.ipv4.conf.default.rp_filter = 1

# Do not accept source routing
net.ipv4.conf.default.accept_source_route = 0

# Controls the System Request debugging functionality of the kernel
kernel.sysrq = 0

# Controls whether core dumps will append the PID to the core filename.
# Useful for debugging multi-threaded applications.
kernel.core_uses_pid = 1

# Controls the use of TCP syncookies
net.ipv4.tcp_syncookies = 1

# Disable netfilter on bridges.
net.bridge.bridge-nf-call-ip6tables = 0
net.bridge.bridge-nf-call-iptables = 0
net.bridge.bridge-nf-call-arptables = 0

# Controls the default maximum size of a message queue
kernel.msgmnb = 65536

# Controls the maximum size of a message, in bytes
kernel.msgmax = 65536

# Controls the maximum shared segment size, in bytes
#kernel.shmmax = 394234920960
kernel.shmmax = 790825623552

# Controls the maximum number of shared memory segments, in pages
#kernel.shmall = 96248760
kernel.shmall = 193072662
#kernel.sem = 500      512000  64      2048
kernel.sem = 500      512000  64      387584
#kernel.msgmni = 4096
kernel.msgmni = 1550336
fs.file-max = 524288
net.core.rmem_max = 131071
net.core.wmem_max = 131071
vm.hugetlb_shm_group = 12090

net.ipv4.tcp_mem = 148799136      198398848      297598272
net.ipv4.tcp_dma_copybreak = 262144

```

10.4.7. /proc/version

```

Linux version 2.6.32-348.el6.x86_64 (mockbuild@x86-
022.build.eng.bos.redhat.com) (gcc version 4.4.7 20120313 (Red Hat 4.4.7-
3) (GCC) ) #1 SMP Mon Dec 10 12:48:03 EST 2012

```

11. Appendix C: Database Setup Code

11.1. Database Creation Scripts

11.1.1. create_database.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 - 2010
-- All Rights Reserved.
--
-- US Government Users Restricted Rights - Use, duplication or
-- disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
-----
-- create_database.ddl.sample - Sample Create Database DDL
--
drop database TPCC;

create database TPCC on /autobench/sources/db2_tpcc-1/db_home using
codeset iso8859-1 territory us collate using identity catalog tablespace
managed by system using ('/autobench/sources/db2_tpcc-1/db_home/db1catalog');
```

11.1.2. create_default_bufferpool.ddl

```
-----
-- Licensed Materials - Property of IBM
--
-- (C) COPYRIGHT International Business Machines Corp. 1996, 2010
-- All Rights Reserved.
--
-- US Government Users Restricted Rights - Use, duplication or
-- disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
-----
-- create_bufferpool.ddl.sample - Sample Create Bufferpool DDL
--
connect to tpcc;

alter bufferpool IBMDEFAULTBP size 1000;
create bufferpool IBMDEFAULT8K size 1000 pagesize 8192;
create bufferpool IBMDEFAULT16K size 1000 pagesize 16384;
create bufferpool IBMDEFAULT32K size 1000 pagesize 32768;
connect reset;
terminate;
```

11.1.3. CRTS_WAREHOUSE.ddl

```
connect to TPCC in share mode;
drop tablespace W_001;
create regular tablespace W_001 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_001' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_002;
create regular tablespace W_002 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_002' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_003;
create regular tablespace W_003 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_003' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_004;
create regular tablespace W_004 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_004' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_005;
create regular tablespace W_005 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_005' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_006;
create regular tablespace W_006 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_006' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
```

```
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_007;
create regular tablespace W_007 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_007' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_008;
create regular tablespace W_008 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_008' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_009;
create regular tablespace W_009 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_009' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_010;
create regular tablespace W_010 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_010' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_011;
create regular tablespace W_011 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_011' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_012;
create regular tablespace W_012 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_012' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_013;
create regular tablespace W_013 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_013' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_014;
create regular tablespace W_014 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_014' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_015;
create regular tablespace W_015 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_015' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_016;
create regular tablespace W_016 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_016' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_017;
create regular tablespace W_017 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_017' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_018;
create regular tablespace W_018 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_018' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_019;
create regular tablespace W_019 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_019' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_020;
create regular tablespace W_020 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_020' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_021;
create regular tablespace W_021 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_021' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_022;
create regular tablespace W_022 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_022' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
connect reset;
connect to TPCC in share mode;
drop tablespace W_023;
create regular tablespace W_023 PAGESIZE 4096 managed by database
using ( device '/dev/tpcc/W_023' 1024 ) EXTENTSIZE 32 PREFETCHSIZE 0
BUFFERPOOL IBMDEFAULTBP;
```



```

alter tablespace W_062 prefetchsize 4096;
alter tablespace W_063 prefetchsize 4096;
alter tablespace W_064 prefetchsize 4096;
alter tablespace W_065 prefetchsize 4096;
alter tablespace W_066 prefetchsize 4096;
alter tablespace W_067 prefetchsize 4096;
alter tablespace W_068 prefetchsize 4096;
alter tablespace W_069 prefetchsize 4096;
alter tablespace W_070 prefetchsize 4096;
alter tablespace W_071 prefetchsize 4096;
alter tablespace W_072 prefetchsize 4096;
connect reset;

```

11.1.15. CRTB_WAREHOUSE.dll

```

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 W_001
INDEX IN W_001
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 1 ENDING AT 1445
)
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 W_002
INDEX IN W_002
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 1446 ENDING AT 2890
)
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 W_003
INDEX IN W_003
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 2891 ENDING AT 4335
)
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 W_004
INDEX IN W_004
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 4336 ENDING AT 5780
)
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 W_005
INDEX IN W_005
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 5781 ENDING AT 7225
)
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 W_006
INDEX IN W_006
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 7226 ENDING AT 8670
)
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 W_007
INDEX IN W_007
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 8671 ENDING AT 10115
)
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 W_008
INDEX IN W_008
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 10116 ENDING AT 11560
)
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 W_009
INDEX IN W_009
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 11561 ENDING AT 13005
)
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 W_010
INDEX IN W_010
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 13006 ENDING AT 14450
)
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 W_011
INDEX IN W_011
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 14451 ENDING AT 15895
)
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 W_012
INDEX IN W_012
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 15896 ENDING AT 17340
)
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 W_013
INDEX IN W_013
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 17341 ENDING AT 18785
)
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 W_014
INDEX IN W_014
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 18786 ENDING AT 20230
)
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 W_015
INDEX IN W_015
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 20231 ENDING AT 21675
)
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 W_016
INDEX IN W_016
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 21676 ENDING AT 23120
)
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 W_017
INDEX IN W_017
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 23121 ENDING AT 24565
)
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 W_018
INDEX IN W_018
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 24566 ENDING AT 26010
)
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 W_019
INDEX IN W_019
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 26011 ENDING AT 27455
)

```

```

)
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 W_020
INDEX IN W_020
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 27456 ENDING AT 28900
)
)
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 W_021
INDEX IN W_021
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 28901 ENDING AT 30345
)
)
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 W_022
INDEX IN W_022
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 30346 ENDING AT 31790
)
)
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 W_023
INDEX IN W_023
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 31791 ENDING AT 33235
)
)
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 W_024
INDEX IN W_024

```

```

ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 33236 ENDING AT 34680
)
)
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 W_025
INDEX IN W_025
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 34681 ENDING AT 36125
)
)
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 W_026
INDEX IN W_026
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 36126 ENDING AT 37570
)
)
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 W_027
INDEX IN W_027
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 37571 ENDING AT 39015
)
)
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 W_028
INDEX IN W_028
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 39016 ENDING AT 40460
)
)
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 W_029
INDEX IN W_029
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 40461 ENDING AT 41905
)
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 W_030
INDEX IN W_030
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 41906 ENDING AT 43350
)
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 W_031
INDEX IN W_031
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 43351 ENDING AT 44795
)
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 W_032
INDEX IN W_032
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 44796 ENDING AT 46240
)
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 W_033
INDEX IN W_033
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 46241 ENDING AT 47685
)
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 W_034
INDEX IN W_034
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 47686 ENDING AT 49130
)
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 W_035
INDEX IN W_035
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 49131 ENDING AT 50575
)
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 W_036
INDEX IN W_036
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 50576 ENDING AT 52020
)
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 W_037
INDEX IN W_037
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 52021 ENDING AT 53465
)
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 W_038
INDEX IN W_038
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 53466 ENDING AT 54910
)
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 W_039
INDEX IN W_039
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 54911 ENDING AT 56355
)
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 W_040
INDEX IN W_040
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 56356 ENDING AT 57800
)
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 W_041
INDEX IN W_041
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 57801 ENDING AT 59245
)
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 W_042
INDEX IN W_042
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 59246 ENDING AT 60690
)
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 W_043
INDEX IN W_043
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 60691 ENDING AT 62135
)
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 W_044
INDEX IN W_044
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 62136 ENDING AT 63580
)
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 W_045
INDEX IN W_045
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 63581 ENDING AT 65025
)
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 W_046
INDEX IN W_046
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 65026 ENDING AT 66470
)
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 W_047
INDEX IN W_047
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 66471 ENDING AT 67915
)
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 W_048
INDEX IN W_048
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 67916 ENDING AT 69360
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE49;
CREATE TABLE WAREHOUSE49
(
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 W_049
INDEX IN W_049
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 69361 ENDING AT 70805
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE50;
CREATE TABLE WAREHOUSE50
(
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 W_050
INDEX IN W_050
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 70806 ENDING AT 72250
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE51;
CREATE TABLE WAREHOUSE51
(
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 W_051
INDEX IN W_051
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 72251 ENDING AT 73695
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE52;
CREATE TABLE WAREHOUSE52
(
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 W_052
INDEX IN W_052
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 73696 ENDING AT 75140
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE53;
CREATE TABLE WAREHOUSE53
(
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 W_053
INDEX IN W_053
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 75141 ENDING AT 76585
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE54;
CREATE TABLE WAREHOUSE54
(

```

```

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 W_054
INDEX IN W_054
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 76586 ENDING AT 78030
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE55;
CREATE TABLE WAREHOUSE55
(
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 W_055
INDEX IN W_055
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 78031 ENDING AT 79475
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE56;
CREATE TABLE WAREHOUSE56
(
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 W_056
INDEX IN W_056
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 79476 ENDING AT 80920
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE57;
CREATE TABLE WAREHOUSE57
(
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 W_057
INDEX IN W_057
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 80921 ENDING AT 82365
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE58;
CREATE TABLE WAREHOUSE58
(
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 W_058
INDEX IN W_058
ORGANIZE BY KEY SEQUENCE (
W_ID STARTING FROM 82366 ENDING AT 83810
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE59;

```



```

CREATE TABLE WAREHOUSE59
(
  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 W_059
INDEX IN W_059
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 83811 ENDING AT 85255
)
ALLOW OVERFLOW;

```

connect reset;

connect to TPCC in share mode;

DROP TABLE WAREHOUSE60;

CREATE TABLE WAREHOUSE60

```

(
  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 W_060
INDEX IN W_060
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 85256 ENDING AT 86700
)
ALLOW OVERFLOW;

```

connect reset;

connect to TPCC in share mode;

DROP TABLE WAREHOUSE61;

CREATE TABLE WAREHOUSE61

```

(
  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 W_061
INDEX IN W_061
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 86701 ENDING AT 88145
)
ALLOW OVERFLOW;

```

connect reset;

connect to TPCC in share mode;

DROP TABLE WAREHOUSE62;

CREATE TABLE WAREHOUSE62

```

(
  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 W_062
INDEX IN W_062
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 88146 ENDING AT 89590
)
ALLOW OVERFLOW;

```

connect reset;

connect to TPCC in share mode;

DROP TABLE WAREHOUSE63;

CREATE TABLE WAREHOUSE63

```

(
  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 W_063
INDEX IN W_063
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 89591 ENDING AT 91035
)
ALLOW OVERFLOW;

```

connect reset;

connect to TPCC in share mode;

DROP TABLE WAREHOUSE64;

CREATE TABLE WAREHOUSE64

```

(
  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 W_064
INDEX IN W_064
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 91036 ENDING AT 92480
)
ALLOW OVERFLOW;

```

connect reset;

connect to TPCC in share mode;

DROP TABLE WAREHOUSE65;

CREATE TABLE WAREHOUSE65

```

(
  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 W_065
INDEX IN W_065
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 92481 ENDING AT 93925
)
ALLOW OVERFLOW;

```

connect reset;

connect to TPCC in share mode;

DROP TABLE WAREHOUSE66;

CREATE TABLE WAREHOUSE66

```

(
  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 W_066
INDEX IN W_066
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 93926 ENDING AT 95370
)
ALLOW OVERFLOW;

```

connect reset;

connect to TPCC in share mode;

DROP TABLE WAREHOUSE67;

CREATE TABLE WAREHOUSE67

```

(
  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 W_067
INDEX IN W_067
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 95371 ENDING AT 96815
)
ALLOW OVERFLOW;

```

connect reset;

connect to TPCC in share mode;

DROP TABLE WAREHOUSE68;

CREATE TABLE WAREHOUSE68

```

(
  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 W_068
INDEX IN W_068
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 96816 ENDING AT 98260
)

```

```

ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE69;
CREATE TABLE WAREHOUSE69
(
  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 W_069
INDEX IN W_069
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 98261 ENDING AT 99705
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE70;
CREATE TABLE WAREHOUSE70
(
  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 W_070
INDEX IN W_070
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 99706 ENDING AT 101150
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE71;
CREATE TABLE WAREHOUSE71
(
  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 W_071
INDEX IN W_071
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 101151 ENDING AT 102595
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE WAREHOUSE72;
CREATE TABLE WAREHOUSE72
(
  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 W_072
INDEX IN W_072
ORGANIZE BY KEY SEQUENCE (
  W_ID STARTING FROM 102596 ENDING AT 104040
)
ALLOW OVERFLOW;

connect reset;

```

11.1.16. 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 D_001
INDEX IN D_001
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 1 ENDING AT 1445
)
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 D_002
INDEX IN D_002
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 1446 ENDING AT 2890
)
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 D_003
INDEX IN D_003
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 2891 ENDING AT 4335
)
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 D_004
INDEX IN D_004
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 4336 ENDING AT 5780
)
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 D_005
INDEX IN D_005
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,

```

```

D_W_ID STARTING FROM 5781 ENDING AT 7225
)
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 D_006
INDEX IN D_006
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 7226 ENDING AT 8670
)
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 D_007
INDEX IN D_007
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 8671 ENDING AT 10115
)
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 D_008
INDEX IN D_008
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 10116 ENDING AT 11560
)
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 D_009
INDEX IN D_009
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 11561 ENDING AT 13005
)
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 D_010
INDEX IN D_010
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 13006 ENDING AT 14450
)
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 D_011
INDEX IN D_011
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 14451 ENDING AT 15895
)
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 D_012
INDEX IN D_012
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 15896 ENDING AT 17340
)
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 D_013
INDEX IN D_013
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 17341 ENDING AT 18785
)
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 D_014
INDEX IN D_014
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 18786 ENDING AT 20230
)
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 D_015
INDEX IN D_015
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 20231 ENDING AT 21675
)
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 D_016
INDEX IN D_016
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 21676 ENDING AT 23120
)
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 D_017
INDEX IN D_017
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 23121 ENDING AT 24565
)
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 D_018

```

```

INDEX IN D_018
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 24566 ENDING AT 26010
)
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 D_019
INDEX IN D_019
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 26011 ENDING AT 27455
)
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 D_020
INDEX IN D_020
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 27456 ENDING AT 28900
)
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 D_021
INDEX IN D_021
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 28901 ENDING AT 30345
)
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 D_022
INDEX IN D_022
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 30346 ENDING AT 31790
)
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 D_023
INDEX IN D_023
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 31791 ENDING AT 33235
)
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 D_024
INDEX IN D_024
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 33236 ENDING AT 34680
)
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 D_025
INDEX IN D_025
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 34681 ENDING AT 36125
)
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 D_026
INDEX IN D_026
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 36126 ENDING AT 37570
)
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 D_027
INDEX IN D_027
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 37571 ENDING AT 39015
)
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
)
IN D_028
INDEX IN D_028
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 39016 ENDING AT 40460
)
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 D_029
INDEX IN D_029
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 40461 ENDING AT 41905
)
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 D_030
INDEX IN D_030
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 41906 ENDING AT 43350
)
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 D_031
    INDEX IN D_031
    ORGANIZE BY KEY SEQUENCE (
        D_ID STARTING FROM 1 ENDING AT 10,
        D_W_ID STARTING FROM 43351 ENDING AT 44795
    )
    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 D_032
INDEX IN D_032
ORGANIZE BY KEY SEQUENCE (
    D_ID STARTING FROM 1 ENDING AT 10,
    D_W_ID STARTING FROM 44796 ENDING AT 46240
)
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 D_033
INDEX IN D_033
ORGANIZE BY KEY SEQUENCE (
    D_ID STARTING FROM 1 ENDING AT 10,
    D_W_ID STARTING FROM 46241 ENDING AT 47685
)
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 D_034
INDEX IN D_034
ORGANIZE BY KEY SEQUENCE (
    D_ID STARTING FROM 1 ENDING AT 10,
    D_W_ID STARTING FROM 47686 ENDING AT 49130
)
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 D_035
INDEX IN D_035
ORGANIZE BY KEY SEQUENCE (
    D_ID STARTING FROM 1 ENDING AT 10,
    D_W_ID STARTING FROM 49131 ENDING AT 50575

```

```

    )
    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 D_036
INDEX IN D_036
ORGANIZE BY KEY SEQUENCE (
    D_ID STARTING FROM 1 ENDING AT 10,
    D_W_ID STARTING FROM 50576 ENDING AT 52020
)
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 D_037
INDEX IN D_037
ORGANIZE BY KEY SEQUENCE (
    D_ID STARTING FROM 1 ENDING AT 10,
    D_W_ID STARTING FROM 52021 ENDING AT 53465
)
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 D_038
INDEX IN D_038
ORGANIZE BY KEY SEQUENCE (
    D_ID STARTING FROM 1 ENDING AT 10,
    D_W_ID STARTING FROM 53466 ENDING AT 54910
)
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 D_039
INDEX IN D_039
ORGANIZE BY KEY SEQUENCE (
    D_ID STARTING FROM 1 ENDING AT 10,
    D_W_ID STARTING FROM 54911 ENDING AT 56355
)
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 D_040
INDEX IN D_040
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 56356 ENDING AT 57800
)
)
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 D_041
INDEX IN D_041
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 57801 ENDING AT 59245
)
)
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 D_042
INDEX IN D_042
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 59246 ENDING AT 60690
)
)
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 D_043
INDEX IN D_043
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 60691 ENDING AT 62135
)
)
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 D_044
INDEX IN D_044
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 62136 ENDING AT 63580
)
)
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 D_045
INDEX IN D_045
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 63581 ENDING AT 65025
)
)
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 D_046
INDEX IN D_046
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 65026 ENDING AT 66470
)
)
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 D_047
INDEX IN D_047
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 66471 ENDING AT 67915
)
)
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 D_048
INDEX IN D_048

```

```

ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 67916 ENDING AT 69360
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT49;
CREATE TABLE DISTRICT49
(
  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 D_049
INDEX IN D_049
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 69361 ENDING AT 70805
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT50;
CREATE TABLE DISTRICT50
(
  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 D_050
INDEX IN D_050
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 70806 ENDING AT 72250
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT51;
CREATE TABLE DISTRICT51
(
  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 D_051
INDEX IN D_051
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 72251 ENDING AT 73695
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT52;
CREATE TABLE DISTRICT52
(
  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 D_052
INDEX IN D_052
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 73696 ENDING AT 75140
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;

```

```

DROP TABLE DISTRICT53;
CREATE TABLE DISTRICT53
(
  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 D_053
INDEX IN D_053
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 75141 ENDING AT 76585
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT54;
CREATE TABLE DISTRICT54
(
  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 D_054
INDEX IN D_054
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 76586 ENDING AT 78030
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT55;
CREATE TABLE DISTRICT55
(
  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 D_055
INDEX IN D_055
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 78031 ENDING AT 79475
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT56;
CREATE TABLE DISTRICT56
(
  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 D_056
INDEX IN D_056
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 79476 ENDING AT 80920
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT57;
CREATE TABLE DISTRICT57
(
  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 D_057
INDEX IN D_057
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 80921 ENDING AT 82365
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT58;
CREATE TABLE DISTRICT58
(
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 D_058
INDEX IN D_058
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 82366 ENDING AT 83810
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT59;
CREATE TABLE DISTRICT59
(
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 D_059
INDEX IN D_059
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 83811 ENDING AT 85255
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT60;
CREATE TABLE DISTRICT60
(
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 D_060
INDEX IN D_060
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 85256 ENDING AT 86700
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT61;
CREATE TABLE DISTRICT61
(
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 D_061
INDEX IN D_061
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 86701 ENDING AT 88145
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT62;
CREATE TABLE DISTRICT62
(
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 D_062
INDEX IN D_062
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 88146 ENDING AT 89590
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT63;
CREATE TABLE DISTRICT63
(
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 D_063
INDEX IN D_063
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 89591 ENDING AT 91035
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT64;
CREATE TABLE DISTRICT64
(
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 D_064
INDEX IN D_064
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 91036 ENDING AT 92480
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT65;
CREATE TABLE DISTRICT65
(
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 D_065
INDEX IN D_065
ORGANIZE BY KEY SEQUENCE (
D_ID STARTING FROM 1 ENDING AT 10,
D_W_ID STARTING FROM 92481 ENDING AT 93925
)
)

```

```

ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT66;
CREATE TABLE DISTRICT66
(
  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 D_066
INDEX IN D_066
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 93926 ENDING AT 95370
)
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT67;
CREATE TABLE DISTRICT67
(
  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 D_067
INDEX IN D_067
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 95371 ENDING AT 96815
)
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT68;
CREATE TABLE DISTRICT68
(
  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 D_068
INDEX IN D_068
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 96816 ENDING AT 98260
)
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT69;
CREATE TABLE DISTRICT69
(
  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 D_069
INDEX IN D_069
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 98261 ENDING AT 99705
)
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT70;
CREATE TABLE DISTRICT70
(
  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 D_070
INDEX IN D_070
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 99706 ENDING AT 101150
)
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT71;
CREATE TABLE DISTRICT71
(
  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 D_071
INDEX IN D_071
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 101151 ENDING AT 102595
)
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE DISTRICT72;
CREATE TABLE DISTRICT72
(
  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 D_072
INDEX IN D_072
ORGANIZE BY KEY SEQUENCE (
  D_ID STARTING FROM 1 ENDING AT 10,
  D_W_ID STARTING FROM 102596 ENDING AT 104040
)
)
ALLOW OVERFLOW;

connect reset;

```

11.1.17. 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 I
INDEX IN I
ORGANIZE BY KEY SEQUENCE (
  I_ID STARTING FROM 1 ENDING AT 100000
)
)
ALLOW OVERFLOW;
ALTER TABLE ITEM LOCKSIZE TABLE;
connect reset;

```

11.1.18. 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,

```



```

connect to TPCC in share mode;
DROP TABLE STOCK70;
CREATE TABLE STOCK70
(
  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
)
INDEX IN S_070
ORGANIZE BY KEY SEQUENCE (
  S_I_ID STARTING FROM 1 ENDING AT 100000,
  S_W_ID STARTING FROM 99706 ENDING AT 101150
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE STOCK71;
CREATE TABLE STOCK71
(
  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
)
INDEX IN S_071
ORGANIZE BY KEY SEQUENCE (
  S_I_ID STARTING FROM 1 ENDING AT 100000,
  S_W_ID STARTING FROM 101151 ENDING AT 102595
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE STOCK72;
CREATE TABLE STOCK72
(
  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
)
INDEX IN S_072
ORGANIZE BY KEY SEQUENCE (
  S_I_ID STARTING FROM 1 ENDING AT 100000,
  S_W_ID STARTING FROM 102596 ENDING AT 104040
)
ALLOW OVERFLOW;

```

```
connect reset;
```

11.1.19. 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(50) 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
)

```

```

INDEX IN C2_001
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 1 ENDING AT 1445,
  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(50) 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
)

```

```

INDEX IN C2_002
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 1446 ENDING AT 2890,
  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(50) 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
)

```

```

INDEX IN C2_003
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 2891 ENDING AT 4335,
  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 C_004
INDEX IN C2_004
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 4336 ENDING AT 5780,
  C_D_ID STARTING FROM 1 ENDING AT 10
)

```

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 C_005
INDEX IN C2_005
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 5781 ENDING AT 7225,
  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 C_006
INDEX IN C2_006
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 7226 ENDING AT 8670,
  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 C_007

INDEX IN C2_007

ORGANIZE BY KEY SEQUENCE (

```

  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 8671 ENDING AT 10115,
  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 C_008

INDEX IN C2_008

ORGANIZE BY KEY SEQUENCE (

```

  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 10116 ENDING AT 11560,
  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 C_009

INDEX IN C2_009

ORGANIZE BY KEY SEQUENCE (

```

  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 11561 ENDING AT 13005,
  C_D_ID STARTING FROM 1 ENDING AT 10
)

```

ALLOW OVERFLOW;

connect reset;

connect to TPCC in share mode;

DROP TABLE CUSTOMER10;

CREATE TABLE CUSTOMER10

```

(
  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 C_010
INDEX IN C2_010
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 13006 ENDING AT 14450,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER11;
CREATE TABLE CUSTOMER11

```

```

(
  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 C_011
INDEX IN C2_011
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 14451 ENDING AT 15895,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER12;
CREATE TABLE CUSTOMER12

```

```

(
  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 C_012
INDEX IN C2_012
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 15896 ENDING AT 17340,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;

```

```

DROP TABLE CUSTOMER13;
CREATE TABLE CUSTOMER13

```

```

(
  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 C_013
INDEX IN C2_013
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 17341 ENDING AT 18785,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER14;
CREATE TABLE CUSTOMER14

```

```

(
  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 C_014
INDEX IN C2_014
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 18786 ENDING AT 20230,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER15;
CREATE TABLE CUSTOMER15

```

```

(
  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 C_015
INDEX IN C2_015
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 20231 ENDING AT 21675,
  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 C_016
INDEX IN C2_016
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 21676 ENDING AT 23120,
  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 C_017
INDEX IN C2_017
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 23121 ENDING AT 24565,
  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 C_018
INDEX IN C2_018
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 24566 ENDING AT 26010,
  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 C_019
INDEX IN C2_019
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 26011 ENDING AT 27455,
  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 C_020
INDEX IN C2_020
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 27456 ENDING AT 28900,
  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 C_021
INDEX IN C2_021
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
)

```



```

C_W_ID STARTING FROM 28901 ENDING AT 30345,
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 C_022
INDEX IN C2_022
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 30346 ENDING AT 31790,
  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 C_023
INDEX IN C2_023
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 31791 ENDING AT 33235,
  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 C_024
INDEX IN C2_024

```

```

ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 33236 ENDING AT 34680,
  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 C_025
INDEX IN C2_025
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 34681 ENDING AT 36125,
  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 C_026
INDEX IN C2_026
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 36126 ENDING AT 37570,
  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 C_027
INDEX IN C2_027
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 37571 ENDING AT 39015,
  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 C_028
INDEX IN C2_028
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 39016 ENDING AT 40460,
  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 C_029
INDEX IN C2_029
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 40461 ENDING AT 41905,
  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
)

```

```

C_PAYMENT_CNT  INTEGER          NOT NULL
)
IN C_030
INDEX IN C2_030
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 41906 ENDING AT 43350,
  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 C_031
INDEX IN C2_031
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 43351 ENDING AT 44795,
  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 C_032
INDEX IN C2_032
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 44796 ENDING AT 46240,
  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 C_033
INDEX IN C2_033
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 46241 ENDING AT 47685,
  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 C_034
INDEX IN C2_034
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 47686 ENDING AT 49130,
  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 C_035
INDEX IN C2_035
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 49131 ENDING AT 50575,
  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 C_036
INDEX IN C2_036
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 50576 ENDING AT 52020,
  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 C_037
INDEX IN C2_037
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 52021 ENDING AT 53465,
  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 C_038
INDEX IN C2_038
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 53466 ENDING AT 54910,
  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 C_039
INDEX IN C2_039
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 54911 ENDING AT 56355,
  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 C_040
INDEX IN C2_040
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 56356 ENDING AT 57800,
  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 C_041
INDEX IN C2_041
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 57801 ENDING AT 59245,
  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 C_042
INDEX IN C2_042
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 59246 ENDING AT 60690,
  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 C_043
INDEX IN C2_043
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 60691 ENDING AT 62135,
  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 C_044
INDEX IN C2_044
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 62136 ENDING AT 63580,
  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 C_045
INDEX IN C2_045
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 63581 ENDING AT 65025,
  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 C_046
INDEX IN C2_046
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 65026 ENDING AT 66470,
  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 C_047
INDEX IN C2_047
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 66471 ENDING AT 67915,
  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 C_048
INDEX IN C2_048
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 67916 ENDING AT 69360,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER49;
CREATE TABLE CUSTOMER49
(
  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 C_049
INDEX IN C2_049
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 69361 ENDING AT 70805,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER50;
CREATE TABLE CUSTOMER50
(
  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 C_050
INDEX IN C2_050
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 70806 ENDING AT 72250,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER51;
CREATE TABLE CUSTOMER51
(
  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 C_051
INDEX IN C2_051
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 72251 ENDING AT 73695,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER52;
CREATE TABLE CUSTOMER52
(
  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 C_052
INDEX IN C2_052
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 73696 ENDING AT 75140,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER53;
CREATE TABLE CUSTOMER53
(
  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 C_053
INDEX IN C2_053
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 75141 ENDING AT 76585,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER54;
CREATE TABLE CUSTOMER54
(
  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 C_054
INDEX IN C2_054
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 76586 ENDING AT 78030,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER55;
CREATE TABLE CUSTOMER55
(
  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 C_055
INDEX IN C2_055
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 78031 ENDING AT 79475,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER56;
CREATE TABLE CUSTOMER56
(
  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 C_056
INDEX IN C2_056
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 79476 ENDING AT 80920,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER57;
CREATE TABLE CUSTOMER57
(
  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 C_057
INDEX IN C2_057
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 80921 ENDING AT 82365,
  C_D_ID STARTING FROM 1 ENDING AT 10
)

```

```

ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER58;
CREATE TABLE CUSTOMER58
(
  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 C_058
INDEX IN C2_058
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 82366 ENDING AT 83810,
  C_D_ID STARTING FROM 1 ENDING AT 10
)

```

```

ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER59;
CREATE TABLE CUSTOMER59
(
  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 C_059
INDEX IN C2_059
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 83811 ENDING AT 85255,
  C_D_ID STARTING FROM 1 ENDING AT 10
)

```

```

ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER60;
CREATE TABLE CUSTOMER60
(

```

```

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 C_060
INDEX IN C2_060
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 85256 ENDING AT 86700,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER61;
CREATE TABLE CUSTOMER61
(
  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 C_061
INDEX IN C2_061
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 86701 ENDING AT 88145,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER62;
CREATE TABLE CUSTOMER62
(
  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 C_062
INDEX IN C2_062
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 88146 ENDING AT 89590,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER63;

```

```

CREATE TABLE CUSTOMER63
(
  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 C_063
INDEX IN C2_063
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 89591 ENDING AT 91035,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER64;
CREATE TABLE CUSTOMER64

```

```

(
  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 C_064
INDEX IN C2_064
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 91036 ENDING AT 92480,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER65;
CREATE TABLE CUSTOMER65

```

```

(
  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 C_065
INDEX IN C2_065
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 92481 ENDING AT 93925,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

```

```

connect reset;

```

```

connect to TPCC in share mode;
DROP TABLE CUSTOMER66;
CREATE TABLE CUSTOMER66

```

```

(
  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 C_066
INDEX IN C2_066
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 93926 ENDING AT 95370,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER67;
CREATE TABLE CUSTOMER67

```

```

(
  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 C_067
INDEX IN C2_067
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 95371 ENDING AT 96815,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER68;
CREATE TABLE CUSTOMER68

```

```

(
  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 C_068
INDEX IN C2_068
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 96816 ENDING AT 98260,
  C_D_ID STARTING FROM 1 ENDING AT 10
)

```



```

ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER69;
CREATE TABLE CUSTOMER69
(
  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 C_069
INDEX IN C2_069
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 98261 ENDING AT 99705,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER70;
CREATE TABLE CUSTOMER70
(
  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 C_070
INDEX IN C2_070
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 99706 ENDING AT 101150,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER71;
CREATE TABLE CUSTOMER71
(
  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 C_071
INDEX IN C2_071
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 101151 ENDING AT 102595,

```

```

C_D_ID STARTING FROM 1 ENDING AT 10
)
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE CUSTOMER72;
CREATE TABLE CUSTOMER72
(
  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 C_072
INDEX IN C2_072
ORGANIZE BY KEY SEQUENCE (
  C_ID STARTING FROM 1 ENDING AT 3000,
  C_W_ID STARTING FROM 102596 ENDING AT 104040,
  C_D_ID STARTING FROM 1 ENDING AT 10
)
)
ALLOW OVERFLOW;

connect reset;

```

11.1.20. 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 O_001
INDEX IN O2_001
ORGANIZE BY KEY SEQUENCE (
  O_ID STARTING FROM 1 ENDING AT 3794,
  O_W_ID STARTING FROM 1 ENDING AT 1445,
  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 O_002
INDEX IN O2_002
ORGANIZE BY KEY SEQUENCE (
  O_ID STARTING FROM 1 ENDING AT 3794,
  O_W_ID STARTING FROM 1446 ENDING AT 2890,
  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 O_003
INDEX IN O2_003
ORGANIZE BY KEY SEQUENCE (

```

```

O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 2891 ENDING AT 4335,
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 O_004
INDEX IN O2_004
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 4336 ENDING AT 5780,
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 O_005
INDEX IN O2_005
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 5781 ENDING AT 7225,
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 O_006
INDEX IN O2_006
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 7226 ENDING AT 8670,
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 O_007
INDEX IN O2_007
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 8671 ENDING AT 10115,
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 O_008
INDEX IN O2_008
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 10116 ENDING AT 11560,
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 O_009
INDEX IN O2_009
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 11561 ENDING AT 13005,
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 O_010
INDEX IN O2_010
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 13006 ENDING AT 14450,
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 O_011
INDEX IN O2_011
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 14451 ENDING AT 15895,
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,
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 O_012
INDEX IN O2_012
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 15896 ENDING AT 17340,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS13;

```



```

O_W_ID      INTEGER    NOT NULL,
O_D_ID      SMALLINT   NOT NULL
)
IN O_022
INDEX IN O2_022
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 30346 ENDING AT 31790,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS23;
CREATE TABLE ORDERS23
(
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 O_023
INDEX IN O2_023
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 31791 ENDING AT 33235,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS24;
CREATE TABLE ORDERS24
(
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 O_024
INDEX IN O2_024
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 33236 ENDING AT 34680,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS25;
CREATE TABLE ORDERS25
(
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 O_025
INDEX IN O2_025
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 34681 ENDING AT 36125,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS26;
CREATE TABLE ORDERS26
(
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 O_026
INDEX IN O2_026
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 36126 ENDING AT 37570,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS27;
CREATE TABLE ORDERS27

```

```

(
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 O_027
INDEX IN O2_027
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 37571 ENDING AT 39015,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS28;
CREATE TABLE ORDERS28
(
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 O_028
INDEX IN O2_028
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 39016 ENDING AT 40460,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS29;
CREATE TABLE ORDERS29
(
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 O_029
INDEX IN O2_029
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 40461 ENDING AT 41905,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS30;
CREATE TABLE ORDERS30
(
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 O_030
INDEX IN O2_030
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 41906 ENDING AT 43350,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS31;
CREATE TABLE ORDERS31
(
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 O_031
INDEX IN O2_031
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 43351 ENDING AT 44795,

```

```

O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS32;
CREATE TABLE ORDERS32
(
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 O_032
INDEX IN O2_032
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 44796 ENDING AT 46240,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS33;
CREATE TABLE ORDERS33
(
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 O_033
INDEX IN O2_033
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 46241 ENDING AT 47685,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS34;
CREATE TABLE ORDERS34
(
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 O_034
INDEX IN O2_034
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 47686 ENDING AT 49130,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS35;
CREATE TABLE ORDERS35
(
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 O_035
INDEX IN O2_035
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 49131 ENDING AT 50575,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS36;
CREATE TABLE ORDERS36
(
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 O_036
INDEX IN O2_036
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 50576 ENDING AT 52020,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS37;
CREATE TABLE ORDERS37
(
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 O_037
INDEX IN O2_037
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 52021 ENDING AT 53465,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS38;
CREATE TABLE ORDERS38
(
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 O_038
INDEX IN O2_038
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 53466 ENDING AT 54910,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS39;
CREATE TABLE ORDERS39
(
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 O_039
INDEX IN O2_039
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 54911 ENDING AT 56355,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS40;
CREATE TABLE ORDERS40
(
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 O_040
INDEX IN O2_040
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 56356 ENDING AT 57800,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS41;
CREATE TABLE ORDERS41
(

```



```

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 O_069
INDEX IN O2_069
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 98261 ENDING AT 99705,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS70;
CREATE TABLE ORDERS70
(
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 O_070
INDEX IN O2_070
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 99706 ENDING AT 101150,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS71;
CREATE TABLE ORDERS71
(
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 O_071
INDEX IN O2_071
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 101151 ENDING AT 102595,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDERS72;
CREATE TABLE ORDERS72
(
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 O_072
INDEX IN O2_072
ORGANIZE BY KEY SEQUENCE (
O_ID STARTING FROM 1 ENDING AT 3794,
O_W_ID STARTING FROM 102596 ENDING AT 104040,
O_D_ID STARTING FROM 1 ENDING AT 10
)
ALLOW OVERFLOW;

connect reset;

```

11.1.21. 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 OL_001
INDEX IN OL_001

```

```

ORGANIZE BY KEY SEQUENCE (
OL_W_ID STARTING FROM 1 ENDING AT 1445,
OL_D_ID STARTING FROM 1 ENDING AT 10,
OL_O_ID STARTING FROM 1 ENDING AT 3794,
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 OL_002
INDEX IN OL_002
ORGANIZE BY KEY SEQUENCE (
OL_W_ID STARTING FROM 1446 ENDING AT 2890,
OL_D_ID STARTING FROM 1 ENDING AT 10,
OL_O_ID STARTING FROM 1 ENDING AT 3794,
OL_NUMBER STARTING FROM 1 ENDING AT 15
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDER_LINE3;
CREATE TABLE ORDER_LINE3
(
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 OL_003
INDEX IN OL_003
ORGANIZE BY KEY SEQUENCE (
OL_W_ID STARTING FROM 2891 ENDING AT 4335,
OL_D_ID STARTING FROM 1 ENDING AT 10,
OL_O_ID STARTING FROM 1 ENDING AT 3794,
OL_NUMBER STARTING FROM 1 ENDING AT 15
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDER_LINE4;
CREATE TABLE ORDER_LINE4
(
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 OL_004
INDEX IN OL_004
ORGANIZE BY KEY SEQUENCE (
OL_W_ID STARTING FROM 4336 ENDING AT 5780,
OL_D_ID STARTING FROM 1 ENDING AT 10,
OL_O_ID STARTING FROM 1 ENDING AT 3794,
OL_NUMBER STARTING FROM 1 ENDING AT 15
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE ORDER_LINE5;
CREATE TABLE ORDER_LINE5
(
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 OL_005
INDEX IN OL_005
ORGANIZE BY KEY SEQUENCE (
OL_W_ID STARTING FROM 5781 ENDING AT 7225,
OL_D_ID STARTING FROM 1 ENDING AT 10,

```

```

OL_O_ID STARTING FROM 1 ENDING AT 3794,
OL_NUMBER STARTING FROM 1 ENDING AT 15
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDER_LINE6;
CREATE TABLE ORDER_LINE6
(
  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 OL_006
INDEX IN OL_006
ORGANIZE BY KEY SEQUENCE (
  OL_W_ID STARTING FROM 7226 ENDING AT 8670,
  OL_D_ID STARTING FROM 1 ENDING AT 10,
  OL_O_ID STARTING FROM 1 ENDING AT 3794,
  OL_NUMBER STARTING FROM 1 ENDING AT 15
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDER_LINE7;
CREATE TABLE ORDER_LINE7
(
  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 OL_007
INDEX IN OL_007
ORGANIZE BY KEY SEQUENCE (
  OL_W_ID STARTING FROM 8671 ENDING AT 10115,
  OL_D_ID STARTING FROM 1 ENDING AT 10,
  OL_O_ID STARTING FROM 1 ENDING AT 3794,
  OL_NUMBER STARTING FROM 1 ENDING AT 15
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDER_LINE8;
CREATE TABLE ORDER_LINE8
(
  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 OL_008
INDEX IN OL_008
ORGANIZE BY KEY SEQUENCE (
  OL_W_ID STARTING FROM 10116 ENDING AT 11560,
  OL_D_ID STARTING FROM 1 ENDING AT 10,
  OL_O_ID STARTING FROM 1 ENDING AT 3794,
  OL_NUMBER STARTING FROM 1 ENDING AT 15
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDER_LINE9;
CREATE TABLE ORDER_LINE9
(
  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 OL_009
INDEX IN OL_009
ORGANIZE BY KEY SEQUENCE (
  OL_W_ID STARTING FROM 11561 ENDING AT 13005,
  OL_D_ID STARTING FROM 1 ENDING AT 10,
  OL_O_ID STARTING FROM 1 ENDING AT 3794,
  OL_NUMBER STARTING FROM 1 ENDING AT 15
)

```

```

ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDER_LINE10;
CREATE TABLE ORDER_LINE10
(
  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 OL_010
INDEX IN OL_010
ORGANIZE BY KEY SEQUENCE (
  OL_W_ID STARTING FROM 13006 ENDING AT 14450,
  OL_D_ID STARTING FROM 1 ENDING AT 10,
  OL_O_ID STARTING FROM 1 ENDING AT 3794,
  OL_NUMBER STARTING FROM 1 ENDING AT 15
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDER_LINE11;
CREATE TABLE ORDER_LINE11
(
  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 OL_011
INDEX IN OL_011
ORGANIZE BY KEY SEQUENCE (
  OL_W_ID STARTING FROM 14451 ENDING AT 15895,
  OL_D_ID STARTING FROM 1 ENDING AT 10,
  OL_O_ID STARTING FROM 1 ENDING AT 3794,
  OL_NUMBER STARTING FROM 1 ENDING AT 15
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDER_LINE12;
CREATE TABLE ORDER_LINE12
(
  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 OL_012
INDEX IN OL_012
ORGANIZE BY KEY SEQUENCE (
  OL_W_ID STARTING FROM 15896 ENDING AT 17340,
  OL_D_ID STARTING FROM 1 ENDING AT 10,
  OL_O_ID STARTING FROM 1 ENDING AT 3794,
  OL_NUMBER STARTING FROM 1 ENDING AT 15
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDER_LINE13;
CREATE TABLE ORDER_LINE13
(
  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 OL_013
INDEX IN OL_013
ORGANIZE BY KEY SEQUENCE (
  OL_W_ID STARTING FROM 17341 ENDING AT 18785,
  OL_D_ID STARTING FROM 1 ENDING AT 10,
  OL_O_ID STARTING FROM 1 ENDING AT 3794,
  OL_NUMBER STARTING FROM 1 ENDING AT 15
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;

```



```

OL_W_ID STARTING FROM 101151 ENDING AT 102595,
OL_D_ID STARTING FROM 1 ENDING AT 10,
OL_O_ID STARTING FROM 1 ENDING AT 3794,
OL_NUMBER STARTING FROM 1 ENDING AT 15
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE ORDER_LINE72;
CREATE TABLE ORDER_LINE72
(
  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 OL_072
INDEX IN OL_072
ORGANIZE BY KEY SEQUENCE (
  OL_W_ID STARTING FROM 102596 ENDING AT 104040,
  OL_D_ID STARTING FROM 1 ENDING AT 10,
  OL_O_ID STARTING FROM 1 ENDING AT 3794,
  OL_NUMBER STARTING FROM 1 ENDING AT 15
)
ALLOW OVERFLOW;
connect reset;

```

11.1.22. CRTB_NEW_ORDER.ddl

```

connect to TPCC in share mode;
DROP TABLE NEW_ORDER1;
CREATE TABLE NEW_ORDER1
(
  NO_O_ID          INTEGER       NOT NULL,
  NO_D_ID          SMALLINT     NOT NULL,
  NO_W_ID          INTEGER       NOT NULL
)
IN N_001
INDEX IN N_001
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 1 ENDING AT 1445,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER2;
CREATE TABLE NEW_ORDER2
(
  NO_O_ID          INTEGER       NOT NULL,
  NO_D_ID          SMALLINT     NOT NULL,
  NO_W_ID          INTEGER       NOT NULL
)
IN N_002
INDEX IN N_002
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 1446 ENDING AT 2890,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER3;
CREATE TABLE NEW_ORDER3
(
  NO_O_ID          INTEGER       NOT NULL,
  NO_D_ID          SMALLINT     NOT NULL,
  NO_W_ID          INTEGER       NOT NULL
)
IN N_003
INDEX IN N_003
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 2891 ENDING AT 4335,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER4;
CREATE TABLE NEW_ORDER4
(
  NO_O_ID          INTEGER       NOT NULL,
  NO_D_ID          SMALLINT     NOT NULL,
  NO_W_ID          INTEGER       NOT NULL
)
IN N_004
INDEX IN N_004
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 4336 ENDING AT 5780,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

```

```

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER5;
CREATE TABLE NEW_ORDER5
(
  NO_O_ID          INTEGER       NOT NULL,
  NO_D_ID          SMALLINT     NOT NULL,
  NO_W_ID          INTEGER       NOT NULL
)
IN N_005
INDEX IN N_005
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 5781 ENDING AT 7225,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER6;
CREATE TABLE NEW_ORDER6
(
  NO_O_ID          INTEGER       NOT NULL,
  NO_D_ID          SMALLINT     NOT NULL,
  NO_W_ID          INTEGER       NOT NULL
)
IN N_006
INDEX IN N_006
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 7226 ENDING AT 8670,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER7;
CREATE TABLE NEW_ORDER7
(
  NO_O_ID          INTEGER       NOT NULL,
  NO_D_ID          SMALLINT     NOT NULL,
  NO_W_ID          INTEGER       NOT NULL
)
IN N_007
INDEX IN N_007
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 8671 ENDING AT 10115,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER8;
CREATE TABLE NEW_ORDER8
(
  NO_O_ID          INTEGER       NOT NULL,
  NO_D_ID          SMALLINT     NOT NULL,
  NO_W_ID          INTEGER       NOT NULL
)
IN N_008
INDEX IN N_008
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 10116 ENDING AT 11560,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER9;
CREATE TABLE NEW_ORDER9
(
  NO_O_ID          INTEGER       NOT NULL,
  NO_D_ID          SMALLINT     NOT NULL,
  NO_W_ID          INTEGER       NOT NULL
)
IN N_009
INDEX IN N_009
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 11561 ENDING AT 13005,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;
connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER10;
CREATE TABLE NEW_ORDER10
(
  NO_O_ID          INTEGER       NOT NULL,
  NO_D_ID          SMALLINT     NOT NULL,
  NO_W_ID          INTEGER       NOT NULL
)
IN N_010
INDEX IN N_010
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 13006 ENDING AT 14450,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;
connect reset;

```

```

connect to TPCC in share mode;
DROP TABLE NEW_ORDER11;
CREATE TABLE NEW_ORDER11
(
    NO_O_ID          INTEGER    NOT NULL,
    NO_D_ID          SMALLINT  NOT NULL,
    NO_W_ID          INTEGER    NOT NULL
)
IN N_011
INDEX IN N_011
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 14451 ENDING AT 15895,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER12;
CREATE TABLE NEW_ORDER12
(
    NO_O_ID          INTEGER    NOT NULL,
    NO_D_ID          SMALLINT  NOT NULL,
    NO_W_ID          INTEGER    NOT NULL
)
IN N_012
INDEX IN N_012
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 15896 ENDING AT 17340,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER13;
CREATE TABLE NEW_ORDER13
(
    NO_O_ID          INTEGER    NOT NULL,
    NO_D_ID          SMALLINT  NOT NULL,
    NO_W_ID          INTEGER    NOT NULL
)
IN N_013
INDEX IN N_013
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 17341 ENDING AT 18785,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER14;
CREATE TABLE NEW_ORDER14
(
    NO_O_ID          INTEGER    NOT NULL,
    NO_D_ID          SMALLINT  NOT NULL,
    NO_W_ID          INTEGER    NOT NULL
)
IN N_014
INDEX IN N_014
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 18786 ENDING AT 20230,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER15;
CREATE TABLE NEW_ORDER15
(
    NO_O_ID          INTEGER    NOT NULL,
    NO_D_ID          SMALLINT  NOT NULL,
    NO_W_ID          INTEGER    NOT NULL
)
IN N_015
INDEX IN N_015
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 20231 ENDING AT 21675,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER16;
CREATE TABLE NEW_ORDER16
(
    NO_O_ID          INTEGER    NOT NULL,
    NO_D_ID          SMALLINT  NOT NULL,
    NO_W_ID          INTEGER    NOT NULL
)
IN N_016
INDEX IN N_016
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 21676 ENDING AT 23120,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;

```

```

DROP TABLE NEW_ORDER17;
CREATE TABLE NEW_ORDER17
(
    NO_O_ID          INTEGER    NOT NULL,
    NO_D_ID          SMALLINT  NOT NULL,
    NO_W_ID          INTEGER    NOT NULL
)
IN N_017
INDEX IN N_017
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 23121 ENDING AT 24565,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER18;
CREATE TABLE NEW_ORDER18
(
    NO_O_ID          INTEGER    NOT NULL,
    NO_D_ID          SMALLINT  NOT NULL,
    NO_W_ID          INTEGER    NOT NULL
)
IN N_018
INDEX IN N_018
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 24566 ENDING AT 26010,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER19;
CREATE TABLE NEW_ORDER19
(
    NO_O_ID          INTEGER    NOT NULL,
    NO_D_ID          SMALLINT  NOT NULL,
    NO_W_ID          INTEGER    NOT NULL
)
IN N_019
INDEX IN N_019
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 26011 ENDING AT 27455,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER20;
CREATE TABLE NEW_ORDER20
(
    NO_O_ID          INTEGER    NOT NULL,
    NO_D_ID          SMALLINT  NOT NULL,
    NO_W_ID          INTEGER    NOT NULL
)
IN N_020
INDEX IN N_020
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 27456 ENDING AT 28900,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER21;
CREATE TABLE NEW_ORDER21
(
    NO_O_ID          INTEGER    NOT NULL,
    NO_D_ID          SMALLINT  NOT NULL,
    NO_W_ID          INTEGER    NOT NULL
)
IN N_021
INDEX IN N_021
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 28901 ENDING AT 30345,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER22;
CREATE TABLE NEW_ORDER22
(
    NO_O_ID          INTEGER    NOT NULL,
    NO_D_ID          SMALLINT  NOT NULL,
    NO_W_ID          INTEGER    NOT NULL
)
IN N_022
INDEX IN N_022
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 30346 ENDING AT 31790,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER23;

```

```

CREATE TABLE NEW_ORDER23
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_023
INDEX IN N_023
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 31791 ENDING AT 33235,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER24;
CREATE TABLE NEW_ORDER24
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_024
INDEX IN N_024
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 33236 ENDING AT 34680,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER25;
CREATE TABLE NEW_ORDER25
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_025
INDEX IN N_025
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 34681 ENDING AT 36125,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER26;
CREATE TABLE NEW_ORDER26
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_026
INDEX IN N_026
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 36126 ENDING AT 37570,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER27;
CREATE TABLE NEW_ORDER27
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_027
INDEX IN N_027
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 37571 ENDING AT 39015,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER28;
CREATE TABLE NEW_ORDER28
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_028
INDEX IN N_028
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 39016 ENDING AT 40460,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER29;
CREATE TABLE NEW_ORDER29

```

```

(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_029
INDEX IN N_029
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 40461 ENDING AT 41905,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER30;
CREATE TABLE NEW_ORDER30
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_030
INDEX IN N_030
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 41906 ENDING AT 43350,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER31;
CREATE TABLE NEW_ORDER31
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_031
INDEX IN N_031
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 43351 ENDING AT 44795,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER32;
CREATE TABLE NEW_ORDER32
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_032
INDEX IN N_032
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 44796 ENDING AT 46240,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER33;
CREATE TABLE NEW_ORDER33
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_033
INDEX IN N_033
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 46241 ENDING AT 47685,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER34;
CREATE TABLE NEW_ORDER34
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_034
INDEX IN N_034
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 47686 ENDING AT 49130,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER35;
CREATE TABLE NEW_ORDER35
(

```

```

NO_O_ID      INTEGER      NOT NULL,
NO_D_ID      SMALLINT    NOT NULL,
NO_W_ID      INTEGER      NOT NULL
)
IN N_035
INDEX IN N_035
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 49131 ENDING AT 50575,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER36;
CREATE TABLE NEW_ORDER36
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT    NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_036
INDEX IN N_036
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 50576 ENDING AT 52020,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER37;
CREATE TABLE NEW_ORDER37
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT    NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_037
INDEX IN N_037
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 52021 ENDING AT 53465,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER38;
CREATE TABLE NEW_ORDER38
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT    NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_038
INDEX IN N_038
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 53466 ENDING AT 54910,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER39;
CREATE TABLE NEW_ORDER39
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT    NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_039
INDEX IN N_039
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 54911 ENDING AT 56355,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER40;
CREATE TABLE NEW_ORDER40
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT    NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_040
INDEX IN N_040
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 56356 ENDING AT 57800,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER41;
CREATE TABLE NEW_ORDER41
(
  NO_O_ID      INTEGER      NOT NULL,

```

```

NO_D_ID      SMALLINT    NOT NULL,
NO_W_ID      INTEGER      NOT NULL
)
IN N_041
INDEX IN N_041
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 57801 ENDING AT 59245,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER42;
CREATE TABLE NEW_ORDER42
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT    NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_042
INDEX IN N_042
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 59246 ENDING AT 60690,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER43;
CREATE TABLE NEW_ORDER43
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT    NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_043
INDEX IN N_043
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 60691 ENDING AT 62135,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER44;
CREATE TABLE NEW_ORDER44
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT    NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_044
INDEX IN N_044
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 62136 ENDING AT 63580,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER45;
CREATE TABLE NEW_ORDER45
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT    NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_045
INDEX IN N_045
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 63581 ENDING AT 65025,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER46;
CREATE TABLE NEW_ORDER46
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT    NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_046
INDEX IN N_046
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 65026 ENDING AT 66470,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER47;
CREATE TABLE NEW_ORDER47
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT    NOT NULL,

```

```

        NO_W_ID          INTEGER      NOT NULL
    )
    IN N_047
    INDEX IN N_047
    ORGANIZE BY KEY SEQUENCE (
        NO_W_ID STARTING FROM 66471 ENDING AT 67915,
        NO_D_ID STARTING FROM 1 ENDING AT 10,
        NO_O_ID STARTING FROM 1900 ENDING AT 3794
    )
    ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER48;
CREATE TABLE NEW_ORDER48
(
    NO_O_ID          INTEGER      NOT NULL,
    NO_D_ID          SMALLINT    NOT NULL,
    NO_W_ID          INTEGER      NOT NULL
)
IN N_048
INDEX IN N_048
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 67916 ENDING AT 69360,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER49;
CREATE TABLE NEW_ORDER49
(
    NO_O_ID          INTEGER      NOT NULL,
    NO_D_ID          SMALLINT    NOT NULL,
    NO_W_ID          INTEGER      NOT NULL
)
IN N_049
INDEX IN N_049
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 69361 ENDING AT 70805,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER50;
CREATE TABLE NEW_ORDER50
(
    NO_O_ID          INTEGER      NOT NULL,
    NO_D_ID          SMALLINT    NOT NULL,
    NO_W_ID          INTEGER      NOT NULL
)
IN N_050
INDEX IN N_050
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 70806 ENDING AT 72250,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER51;
CREATE TABLE NEW_ORDER51
(
    NO_O_ID          INTEGER      NOT NULL,
    NO_D_ID          SMALLINT    NOT NULL,
    NO_W_ID          INTEGER      NOT NULL
)
IN N_051
INDEX IN N_051
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 72251 ENDING AT 73695,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER52;
CREATE TABLE NEW_ORDER52
(
    NO_O_ID          INTEGER      NOT NULL,
    NO_D_ID          SMALLINT    NOT NULL,
    NO_W_ID          INTEGER      NOT NULL
)
IN N_052
INDEX IN N_052
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 73696 ENDING AT 75140,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER53;
CREATE TABLE NEW_ORDER53
(
    NO_O_ID          INTEGER      NOT NULL,
    NO_D_ID          SMALLINT    NOT NULL,
    NO_W_ID          INTEGER      NOT NULL

```

```

    )
    IN N_053
    INDEX IN N_053
    ORGANIZE BY KEY SEQUENCE (
        NO_W_ID STARTING FROM 75141 ENDING AT 76585,
        NO_D_ID STARTING FROM 1 ENDING AT 10,
        NO_O_ID STARTING FROM 1900 ENDING AT 3794
    )
    ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER54;
CREATE TABLE NEW_ORDER54
(
    NO_O_ID          INTEGER      NOT NULL,
    NO_D_ID          SMALLINT    NOT NULL,
    NO_W_ID          INTEGER      NOT NULL
)
IN N_054
INDEX IN N_054
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 76586 ENDING AT 78030,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER55;
CREATE TABLE NEW_ORDER55
(
    NO_O_ID          INTEGER      NOT NULL,
    NO_D_ID          SMALLINT    NOT NULL,
    NO_W_ID          INTEGER      NOT NULL
)
IN N_055
INDEX IN N_055
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 78031 ENDING AT 79475,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER56;
CREATE TABLE NEW_ORDER56
(
    NO_O_ID          INTEGER      NOT NULL,
    NO_D_ID          SMALLINT    NOT NULL,
    NO_W_ID          INTEGER      NOT NULL
)
IN N_056
INDEX IN N_056
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 79476 ENDING AT 80920,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER57;
CREATE TABLE NEW_ORDER57
(
    NO_O_ID          INTEGER      NOT NULL,
    NO_D_ID          SMALLINT    NOT NULL,
    NO_W_ID          INTEGER      NOT NULL
)
IN N_057
INDEX IN N_057
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 80921 ENDING AT 82365,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER58;
CREATE TABLE NEW_ORDER58
(
    NO_O_ID          INTEGER      NOT NULL,
    NO_D_ID          SMALLINT    NOT NULL,
    NO_W_ID          INTEGER      NOT NULL
)
IN N_058
INDEX IN N_058
ORGANIZE BY KEY SEQUENCE (
    NO_W_ID STARTING FROM 82366 ENDING AT 83810,
    NO_D_ID STARTING FROM 1 ENDING AT 10,
    NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER59;
CREATE TABLE NEW_ORDER59
(
    NO_O_ID          INTEGER      NOT NULL,
    NO_D_ID          SMALLINT    NOT NULL,
    NO_W_ID          INTEGER      NOT NULL

```

```

IN N_059
INDEX IN N_059
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 83811 ENDING AT 85255,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER60;
CREATE TABLE NEW_ORDER60
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_060
INDEX IN N_060
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 85256 ENDING AT 86700,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER61;
CREATE TABLE NEW_ORDER61
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_061
INDEX IN N_061
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 86701 ENDING AT 88145,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER62;
CREATE TABLE NEW_ORDER62
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_062
INDEX IN N_062
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 88146 ENDING AT 89590,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER63;
CREATE TABLE NEW_ORDER63
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_063
INDEX IN N_063
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 89591 ENDING AT 91035,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER64;
CREATE TABLE NEW_ORDER64
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_064
INDEX IN N_064
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 91036 ENDING AT 92480,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER65;
CREATE TABLE NEW_ORDER65
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_065

```

```

INDEX IN N_065
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 92481 ENDING AT 93925,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER66;
CREATE TABLE NEW_ORDER66
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_066
INDEX IN N_066
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 93926 ENDING AT 95370,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER67;
CREATE TABLE NEW_ORDER67
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_067
INDEX IN N_067
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 95371 ENDING AT 96815,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER68;
CREATE TABLE NEW_ORDER68
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_068
INDEX IN N_068
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 96816 ENDING AT 98260,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER69;
CREATE TABLE NEW_ORDER69
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_069
INDEX IN N_069
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 98261 ENDING AT 99705,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER70;
CREATE TABLE NEW_ORDER70
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_070
INDEX IN N_070
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 99706 ENDING AT 101150,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER71;
CREATE TABLE NEW_ORDER71
(
  NO_O_ID      INTEGER      NOT NULL,
  NO_D_ID      SMALLINT     NOT NULL,
  NO_W_ID      INTEGER      NOT NULL
)
IN N_071
INDEX IN N_071

```

```

ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 101151 ENDING AT 102595,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;
connect to TPCC in share mode;
DROP TABLE NEW_ORDER72;
CREATE TABLE NEW_ORDER72
(
  NO_O_ID          INTEGER    NOT NULL,
  NO_D_ID          SMALLINT  NOT NULL,
  NO_W_ID          INTEGER    NOT NULL
)
IN N_072
INDEX IN N_072
ORGANIZE BY KEY SEQUENCE (
  NO_W_ID STARTING FROM 102596 ENDING AT 104040,
  NO_D_ID STARTING FROM 1 ENDING AT 10,
  NO_O_ID STARTING FROM 1900 ENDING AT 3794
)
ALLOW OVERFLOW;

connect reset;

```

11.1.23. CRTB_HISTORY.ddl

```

connect to TPCC in share mode;
DROP TABLE HISTORY1;
CREATE TABLE HISTORY1
(
  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 H_001
INDEX IN H_001;
ALTER TABLE HISTORY1 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY2;
CREATE TABLE HISTORY2
(
  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 H_002
INDEX IN H_002;
ALTER TABLE HISTORY2 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY3;
CREATE TABLE HISTORY3
(
  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 H_003
INDEX IN H_003;
ALTER TABLE HISTORY3 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY4;
CREATE TABLE HISTORY4
(
  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 H_004
INDEX IN H_004;
ALTER TABLE HISTORY4 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY5;
CREATE TABLE HISTORY5
(
  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 H_005
INDEX IN H_005;
ALTER TABLE HISTORY5 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY6;
CREATE TABLE HISTORY6
(
  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 H_006
INDEX IN H_006;
ALTER TABLE HISTORY6 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY7;
CREATE TABLE HISTORY7
(
  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 H_007
INDEX IN H_007;
ALTER TABLE HISTORY7 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY8;
CREATE TABLE HISTORY8
(
  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 H_008
INDEX IN H_008;
ALTER TABLE HISTORY8 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY9;
CREATE TABLE HISTORY9
(
  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 H_009
INDEX IN H_009;
ALTER TABLE HISTORY9 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY10;
CREATE TABLE HISTORY10
(
  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 H_010
INDEX IN H_010;
ALTER TABLE HISTORY10 APPEND ON;
connect reset;
connect to TPCC in share mode;
DROP TABLE HISTORY11;
CREATE TABLE HISTORY11
(
  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,

```



```

LOAD FROM /autobench/sources/db2_tpcc-
1/backup/flats/flat69/history_069_1.dat OF DEL MODIFIED BY COLDEL|
KEEPBLANKS FASTPARSE INSERT INTO HISTORY69 NONRECOVERABLE DATA BUFFER
16000 CPU_PARALLELISM 4 ;
connect reset;
CONNECT TO TPCC IN SHARE MODE;
LOAD FROM /autobench/sources/db2_tpcc-
1/backup/flats/flat70/history_070_1.dat OF DEL MODIFIED BY COLDEL|
KEEPBLANKS FASTPARSE INSERT INTO HISTORY70 NONRECOVERABLE DATA BUFFER
16000 CPU_PARALLELISM 4 ;
connect reset;
CONNECT TO TPCC IN SHARE MODE;
LOAD FROM /autobench/sources/db2_tpcc-
1/backup/flats/flat71/history_071_1.dat OF DEL MODIFIED BY COLDEL|
KEEPBLANKS FASTPARSE INSERT INTO HISTORY71 NONRECOVERABLE DATA BUFFER
16000 CPU_PARALLELISM 4 ;
connect reset;
CONNECT TO TPCC IN SHARE MODE;
LOAD FROM /autobench/sources/db2_tpcc-
1/backup/flats/flat72/history_072_1.dat OF DEL MODIFIED BY COLDEL|
KEEPBLANKS FASTPARSE INSERT INTO HISTORY72 NONRECOVERABLE DATA BUFFER
16000 CPU_PARALLELISM 4 ;
connect reset;

```

11.1.40. CRCONST_WAREHOUSE.ddl

```

connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE1 OFF;
ALTER TABLE WAREHOUSE1 DROP CONSTRAINT WAREHOUSE1CKC;
ALTER TABLE WAREHOUSE1 ADD CONSTRAINT WAREHOUSE1CKC CHECK (W_ID BETWEEN 1
AND 1445);
SET INTEGRITY FOR WAREHOUSE1 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE2 OFF;
ALTER TABLE WAREHOUSE2 DROP CONSTRAINT WAREHOUSE2CKC;
ALTER TABLE WAREHOUSE2 ADD CONSTRAINT WAREHOUSE2CKC CHECK (W_ID BETWEEN
1446 AND 2890);
SET INTEGRITY FOR WAREHOUSE2 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE3 OFF;
ALTER TABLE WAREHOUSE3 DROP CONSTRAINT WAREHOUSE3CKC;
ALTER TABLE WAREHOUSE3 ADD CONSTRAINT WAREHOUSE3CKC CHECK (W_ID BETWEEN
2891 AND 4335);
SET INTEGRITY FOR WAREHOUSE3 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE4 OFF;
ALTER TABLE WAREHOUSE4 DROP CONSTRAINT WAREHOUSE4CKC;
ALTER TABLE WAREHOUSE4 ADD CONSTRAINT WAREHOUSE4CKC CHECK (W_ID BETWEEN
4336 AND 5780);
SET INTEGRITY FOR WAREHOUSE4 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE5 OFF;
ALTER TABLE WAREHOUSE5 DROP CONSTRAINT WAREHOUSE5CKC;
ALTER TABLE WAREHOUSE5 ADD CONSTRAINT WAREHOUSE5CKC CHECK (W_ID BETWEEN
5781 AND 7225);
SET INTEGRITY FOR WAREHOUSE5 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE6 OFF;
ALTER TABLE WAREHOUSE6 DROP CONSTRAINT WAREHOUSE6CKC;
ALTER TABLE WAREHOUSE6 ADD CONSTRAINT WAREHOUSE6CKC CHECK (W_ID BETWEEN
7226 AND 8670);
SET INTEGRITY FOR WAREHOUSE6 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE7 OFF;
ALTER TABLE WAREHOUSE7 DROP CONSTRAINT WAREHOUSE7CKC;
ALTER TABLE WAREHOUSE7 ADD CONSTRAINT WAREHOUSE7CKC CHECK (W_ID BETWEEN
8671 AND 10115);
SET INTEGRITY FOR WAREHOUSE7 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE8 OFF;
ALTER TABLE WAREHOUSE8 DROP CONSTRAINT WAREHOUSE8CKC;
ALTER TABLE WAREHOUSE8 ADD CONSTRAINT WAREHOUSE8CKC CHECK (W_ID BETWEEN
10116 AND 11560);
SET INTEGRITY FOR WAREHOUSE8 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE9 OFF;
ALTER TABLE WAREHOUSE9 DROP CONSTRAINT WAREHOUSE9CKC;
ALTER TABLE WAREHOUSE9 ADD CONSTRAINT WAREHOUSE9CKC CHECK (W_ID BETWEEN
11561 AND 13005);
SET INTEGRITY FOR WAREHOUSE9 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE10 OFF;
ALTER TABLE WAREHOUSE10 DROP CONSTRAINT WAREHOUSE10CKC;
ALTER TABLE WAREHOUSE10 ADD CONSTRAINT WAREHOUSE10CKC CHECK (W_ID BETWEEN
13006 AND 14450);
SET INTEGRITY FOR WAREHOUSE10 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE11 OFF;
ALTER TABLE WAREHOUSE11 DROP CONSTRAINT WAREHOUSE11CKC;
ALTER TABLE WAREHOUSE11 ADD CONSTRAINT WAREHOUSE11CKC CHECK (W_ID BETWEEN
14451 AND 15895);
SET INTEGRITY FOR WAREHOUSE11 ALL IMMEDIATE UNCHECKED;

```

```

connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE12 OFF;
ALTER TABLE WAREHOUSE12 DROP CONSTRAINT WAREHOUSE12CKC;
ALTER TABLE WAREHOUSE12 ADD CONSTRAINT WAREHOUSE12CKC CHECK (W_ID BETWEEN
15896 AND 17340);
SET INTEGRITY FOR WAREHOUSE12 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE13 OFF;
ALTER TABLE WAREHOUSE13 DROP CONSTRAINT WAREHOUSE13CKC;
ALTER TABLE WAREHOUSE13 ADD CONSTRAINT WAREHOUSE13CKC CHECK (W_ID BETWEEN
17341 AND 18785);
SET INTEGRITY FOR WAREHOUSE13 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE14 OFF;
ALTER TABLE WAREHOUSE14 DROP CONSTRAINT WAREHOUSE14CKC;
ALTER TABLE WAREHOUSE14 ADD CONSTRAINT WAREHOUSE14CKC CHECK (W_ID BETWEEN
18786 AND 20230);
SET INTEGRITY FOR WAREHOUSE14 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE15 OFF;
ALTER TABLE WAREHOUSE15 DROP CONSTRAINT WAREHOUSE15CKC;
ALTER TABLE WAREHOUSE15 ADD CONSTRAINT WAREHOUSE15CKC CHECK (W_ID BETWEEN
20231 AND 21675);
SET INTEGRITY FOR WAREHOUSE15 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE16 OFF;
ALTER TABLE WAREHOUSE16 DROP CONSTRAINT WAREHOUSE16CKC;
ALTER TABLE WAREHOUSE16 ADD CONSTRAINT WAREHOUSE16CKC CHECK (W_ID BETWEEN
21676 AND 23120);
SET INTEGRITY FOR WAREHOUSE16 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE17 OFF;
ALTER TABLE WAREHOUSE17 DROP CONSTRAINT WAREHOUSE17CKC;
ALTER TABLE WAREHOUSE17 ADD CONSTRAINT WAREHOUSE17CKC CHECK (W_ID BETWEEN
23121 AND 24565);
SET INTEGRITY FOR WAREHOUSE17 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE18 OFF;
ALTER TABLE WAREHOUSE18 DROP CONSTRAINT WAREHOUSE18CKC;
ALTER TABLE WAREHOUSE18 ADD CONSTRAINT WAREHOUSE18CKC CHECK (W_ID BETWEEN
24566 AND 26010);
SET INTEGRITY FOR WAREHOUSE18 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE19 OFF;
ALTER TABLE WAREHOUSE19 DROP CONSTRAINT WAREHOUSE19CKC;
ALTER TABLE WAREHOUSE19 ADD CONSTRAINT WAREHOUSE19CKC CHECK (W_ID BETWEEN
26011 AND 27455);
SET INTEGRITY FOR WAREHOUSE19 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE20 OFF;
ALTER TABLE WAREHOUSE20 DROP CONSTRAINT WAREHOUSE20CKC;
ALTER TABLE WAREHOUSE20 ADD CONSTRAINT WAREHOUSE20CKC CHECK (W_ID BETWEEN
27456 AND 28900);
SET INTEGRITY FOR WAREHOUSE20 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE21 OFF;
ALTER TABLE WAREHOUSE21 DROP CONSTRAINT WAREHOUSE21CKC;
ALTER TABLE WAREHOUSE21 ADD CONSTRAINT WAREHOUSE21CKC CHECK (W_ID BETWEEN
28901 AND 30345);
SET INTEGRITY FOR WAREHOUSE21 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE22 OFF;
ALTER TABLE WAREHOUSE22 DROP CONSTRAINT WAREHOUSE22CKC;
ALTER TABLE WAREHOUSE22 ADD CONSTRAINT WAREHOUSE22CKC CHECK (W_ID BETWEEN
30346 AND 31790);
SET INTEGRITY FOR WAREHOUSE22 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE23 OFF;
ALTER TABLE WAREHOUSE23 DROP CONSTRAINT WAREHOUSE23CKC;
ALTER TABLE WAREHOUSE23 ADD CONSTRAINT WAREHOUSE23CKC CHECK (W_ID BETWEEN
31791 AND 33235);
SET INTEGRITY FOR WAREHOUSE23 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE24 OFF;
ALTER TABLE WAREHOUSE24 DROP CONSTRAINT WAREHOUSE24CKC;
ALTER TABLE WAREHOUSE24 ADD CONSTRAINT WAREHOUSE24CKC CHECK (W_ID BETWEEN
33236 AND 34680);
SET INTEGRITY FOR WAREHOUSE24 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE25 OFF;
ALTER TABLE WAREHOUSE25 DROP CONSTRAINT WAREHOUSE25CKC;
ALTER TABLE WAREHOUSE25 ADD CONSTRAINT WAREHOUSE25CKC CHECK (W_ID BETWEEN
34681 AND 36125);
SET INTEGRITY FOR WAREHOUSE25 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE26 OFF;
ALTER TABLE WAREHOUSE26 DROP CONSTRAINT WAREHOUSE26CKC;
ALTER TABLE WAREHOUSE26 ADD CONSTRAINT WAREHOUSE26CKC CHECK (W_ID BETWEEN

```



```

connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE56 OFF;
ALTER TABLE WAREHOUSE56 DROP CONSTRAINT WAREHOUSE56CKC;
ALTER TABLE WAREHOUSE56 ADD CONSTRAINT WAREHOUSE56CKC CHECK (W_ID BETWEEN
79476 AND 80920);
SET INTEGRITY FOR WAREHOUSE56 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE57 OFF;
ALTER TABLE WAREHOUSE57 DROP CONSTRAINT WAREHOUSE57CKC;
ALTER TABLE WAREHOUSE57 ADD CONSTRAINT WAREHOUSE57CKC CHECK (W_ID BETWEEN
80921 AND 82365);
SET INTEGRITY FOR WAREHOUSE57 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE58 OFF;
ALTER TABLE WAREHOUSE58 DROP CONSTRAINT WAREHOUSE58CKC;
ALTER TABLE WAREHOUSE58 ADD CONSTRAINT WAREHOUSE58CKC CHECK (W_ID BETWEEN
82366 AND 83810);
SET INTEGRITY FOR WAREHOUSE58 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE59 OFF;
ALTER TABLE WAREHOUSE59 DROP CONSTRAINT WAREHOUSE59CKC;
ALTER TABLE WAREHOUSE59 ADD CONSTRAINT WAREHOUSE59CKC CHECK (W_ID BETWEEN
83811 AND 85255);
SET INTEGRITY FOR WAREHOUSE59 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE60 OFF;
ALTER TABLE WAREHOUSE60 DROP CONSTRAINT WAREHOUSE60CKC;
ALTER TABLE WAREHOUSE60 ADD CONSTRAINT WAREHOUSE60CKC CHECK (W_ID BETWEEN
85256 AND 86700);
SET INTEGRITY FOR WAREHOUSE60 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE61 OFF;
ALTER TABLE WAREHOUSE61 DROP CONSTRAINT WAREHOUSE61CKC;
ALTER TABLE WAREHOUSE61 ADD CONSTRAINT WAREHOUSE61CKC CHECK (W_ID BETWEEN
86701 AND 88145);
SET INTEGRITY FOR WAREHOUSE61 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE62 OFF;
ALTER TABLE WAREHOUSE62 DROP CONSTRAINT WAREHOUSE62CKC;
ALTER TABLE WAREHOUSE62 ADD CONSTRAINT WAREHOUSE62CKC CHECK (W_ID BETWEEN
88146 AND 89590);
SET INTEGRITY FOR WAREHOUSE62 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE63 OFF;
ALTER TABLE WAREHOUSE63 DROP CONSTRAINT WAREHOUSE63CKC;
ALTER TABLE WAREHOUSE63 ADD CONSTRAINT WAREHOUSE63CKC CHECK (W_ID BETWEEN
89591 AND 91035);
SET INTEGRITY FOR WAREHOUSE63 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE64 OFF;
ALTER TABLE WAREHOUSE64 DROP CONSTRAINT WAREHOUSE64CKC;
ALTER TABLE WAREHOUSE64 ADD CONSTRAINT WAREHOUSE64CKC CHECK (W_ID BETWEEN
91036 AND 92480);
SET INTEGRITY FOR WAREHOUSE64 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE65 OFF;
ALTER TABLE WAREHOUSE65 DROP CONSTRAINT WAREHOUSE65CKC;
ALTER TABLE WAREHOUSE65 ADD CONSTRAINT WAREHOUSE65CKC CHECK (W_ID BETWEEN
92481 AND 93925);
SET INTEGRITY FOR WAREHOUSE65 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE66 OFF;
ALTER TABLE WAREHOUSE66 DROP CONSTRAINT WAREHOUSE66CKC;
ALTER TABLE WAREHOUSE66 ADD CONSTRAINT WAREHOUSE66CKC CHECK (W_ID BETWEEN
93926 AND 95370);
SET INTEGRITY FOR WAREHOUSE66 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE67 OFF;
ALTER TABLE WAREHOUSE67 DROP CONSTRAINT WAREHOUSE67CKC;
ALTER TABLE WAREHOUSE67 ADD CONSTRAINT WAREHOUSE67CKC CHECK (W_ID BETWEEN
95371 AND 96815);
SET INTEGRITY FOR WAREHOUSE67 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE68 OFF;
ALTER TABLE WAREHOUSE68 DROP CONSTRAINT WAREHOUSE68CKC;
ALTER TABLE WAREHOUSE68 ADD CONSTRAINT WAREHOUSE68CKC CHECK (W_ID BETWEEN
96816 AND 98260);
SET INTEGRITY FOR WAREHOUSE68 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE69 OFF;
ALTER TABLE WAREHOUSE69 DROP CONSTRAINT WAREHOUSE69CKC;
ALTER TABLE WAREHOUSE69 ADD CONSTRAINT WAREHOUSE69CKC CHECK (W_ID BETWEEN
98261 AND 99705);
SET INTEGRITY FOR WAREHOUSE69 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE70 OFF;
ALTER TABLE WAREHOUSE70 DROP CONSTRAINT WAREHOUSE70CKC;
ALTER TABLE WAREHOUSE70 ADD CONSTRAINT WAREHOUSE70CKC CHECK (W_ID BETWEEN
99706 AND 101150);

```

```

SET INTEGRITY FOR WAREHOUSE70 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE71 OFF;
ALTER TABLE WAREHOUSE71 DROP CONSTRAINT WAREHOUSE71CKC;
ALTER TABLE WAREHOUSE71 ADD CONSTRAINT WAREHOUSE71CKC CHECK (W_ID BETWEEN
101151 AND 102595);
SET INTEGRITY FOR WAREHOUSE71 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR WAREHOUSE72 OFF;
ALTER TABLE WAREHOUSE72 DROP CONSTRAINT WAREHOUSE72CKC;
ALTER TABLE WAREHOUSE72 ADD CONSTRAINT WAREHOUSE72CKC CHECK (W_ID >=
102596);
SET INTEGRITY FOR WAREHOUSE72 ALL IMMEDIATE UNCHECKED;
connect reset;

```

11.1.41. CRCONST_DISTRICT.ddl

```

connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT1 OFF;
ALTER TABLE DISTRICT1 DROP CONSTRAINT DISTRICT1CKC;
ALTER TABLE DISTRICT1 ADD CONSTRAINT DISTRICT1CKC CHECK (D_W_ID BETWEEN 1
AND 1445);
SET INTEGRITY FOR DISTRICT1 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT2 OFF;
ALTER TABLE DISTRICT2 DROP CONSTRAINT DISTRICT2CKC;
ALTER TABLE DISTRICT2 ADD CONSTRAINT DISTRICT2CKC CHECK (D_W_ID BETWEEN
1446 AND 2890);
SET INTEGRITY FOR DISTRICT2 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT3 OFF;
ALTER TABLE DISTRICT3 DROP CONSTRAINT DISTRICT3CKC;
ALTER TABLE DISTRICT3 ADD CONSTRAINT DISTRICT3CKC CHECK (D_W_ID BETWEEN
2891 AND 4335);
SET INTEGRITY FOR DISTRICT3 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT4 OFF;
ALTER TABLE DISTRICT4 DROP CONSTRAINT DISTRICT4CKC;
ALTER TABLE DISTRICT4 ADD CONSTRAINT DISTRICT4CKC CHECK (D_W_ID BETWEEN
4336 AND 5780);
SET INTEGRITY FOR DISTRICT4 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT5 OFF;
ALTER TABLE DISTRICT5 DROP CONSTRAINT DISTRICT5CKC;
ALTER TABLE DISTRICT5 ADD CONSTRAINT DISTRICT5CKC CHECK (D_W_ID BETWEEN
5781 AND 7225);
SET INTEGRITY FOR DISTRICT5 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT6 OFF;
ALTER TABLE DISTRICT6 DROP CONSTRAINT DISTRICT6CKC;
ALTER TABLE DISTRICT6 ADD CONSTRAINT DISTRICT6CKC CHECK (D_W_ID BETWEEN
7226 AND 8670);
SET INTEGRITY FOR DISTRICT6 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT7 OFF;
ALTER TABLE DISTRICT7 DROP CONSTRAINT DISTRICT7CKC;
ALTER TABLE DISTRICT7 ADD CONSTRAINT DISTRICT7CKC CHECK (D_W_ID BETWEEN
8671 AND 10115);
SET INTEGRITY FOR DISTRICT7 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT8 OFF;
ALTER TABLE DISTRICT8 DROP CONSTRAINT DISTRICT8CKC;
ALTER TABLE DISTRICT8 ADD CONSTRAINT DISTRICT8CKC CHECK (D_W_ID BETWEEN
10116 AND 11560);
SET INTEGRITY FOR DISTRICT8 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT9 OFF;
ALTER TABLE DISTRICT9 DROP CONSTRAINT DISTRICT9CKC;
ALTER TABLE DISTRICT9 ADD CONSTRAINT DISTRICT9CKC CHECK (D_W_ID BETWEEN
11561 AND 13005);
SET INTEGRITY FOR DISTRICT9 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT10 OFF;
ALTER TABLE DISTRICT10 DROP CONSTRAINT DISTRICT10CKC;
ALTER TABLE DISTRICT10 ADD CONSTRAINT DISTRICT10CKC CHECK (D_W_ID BETWEEN
13006 AND 14450);
SET INTEGRITY FOR DISTRICT10 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT11 OFF;
ALTER TABLE DISTRICT11 DROP CONSTRAINT DISTRICT11CKC;
ALTER TABLE DISTRICT11 ADD CONSTRAINT DISTRICT11CKC CHECK (D_W_ID BETWEEN
14451 AND 15895);
SET INTEGRITY FOR DISTRICT11 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR DISTRICT12 OFF;
ALTER TABLE DISTRICT12 DROP CONSTRAINT DISTRICT12CKC;
ALTER TABLE DISTRICT12 ADD CONSTRAINT DISTRICT12CKC CHECK (D_W_ID BETWEEN
15896 AND 17340);
SET INTEGRITY FOR DISTRICT12 ALL IMMEDIATE UNCHECKED;

```



```

connect to TPCC in share mode;
SET INTEGRITY FOR STOCK58 OFF;
ALTER TABLE STOCK58 DROP CONSTRAINT STOCK58CKC;
ALTER TABLE STOCK58 ADD CONSTRAINT STOCK58CKC CHECK (S_W_ID BETWEEN 82366
AND 83810);
SET INTEGRITY FOR STOCK58 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR STOCK59 OFF;
ALTER TABLE STOCK59 DROP CONSTRAINT STOCK59CKC;
ALTER TABLE STOCK59 ADD CONSTRAINT STOCK59CKC CHECK (S_W_ID BETWEEN 83811
AND 85255);
SET INTEGRITY FOR STOCK59 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR STOCK60 OFF;
ALTER TABLE STOCK60 DROP CONSTRAINT STOCK60CKC;
ALTER TABLE STOCK60 ADD CONSTRAINT STOCK60CKC CHECK (S_W_ID BETWEEN 85256
AND 86700);
SET INTEGRITY FOR STOCK60 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR STOCK61 OFF;
ALTER TABLE STOCK61 DROP CONSTRAINT STOCK61CKC;
ALTER TABLE STOCK61 ADD CONSTRAINT STOCK61CKC CHECK (S_W_ID BETWEEN 86701
AND 88145);
SET INTEGRITY FOR STOCK61 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR STOCK62 OFF;
ALTER TABLE STOCK62 DROP CONSTRAINT STOCK62CKC;
ALTER TABLE STOCK62 ADD CONSTRAINT STOCK62CKC CHECK (S_W_ID BETWEEN 88146
AND 89590);
SET INTEGRITY FOR STOCK62 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR STOCK63 OFF;
ALTER TABLE STOCK63 DROP CONSTRAINT STOCK63CKC;
ALTER TABLE STOCK63 ADD CONSTRAINT STOCK63CKC CHECK (S_W_ID BETWEEN 89591
AND 91035);
SET INTEGRITY FOR STOCK63 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR STOCK64 OFF;
ALTER TABLE STOCK64 DROP CONSTRAINT STOCK64CKC;
ALTER TABLE STOCK64 ADD CONSTRAINT STOCK64CKC CHECK (S_W_ID BETWEEN 91036
AND 92480);
SET INTEGRITY FOR STOCK64 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR STOCK65 OFF;
ALTER TABLE STOCK65 DROP CONSTRAINT STOCK65CKC;
ALTER TABLE STOCK65 ADD CONSTRAINT STOCK65CKC CHECK (S_W_ID BETWEEN 92481
AND 93925);
SET INTEGRITY FOR STOCK65 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR STOCK66 OFF;
ALTER TABLE STOCK66 DROP CONSTRAINT STOCK66CKC;
ALTER TABLE STOCK66 ADD CONSTRAINT STOCK66CKC CHECK (S_W_ID BETWEEN 93926
AND 95370);
SET INTEGRITY FOR STOCK66 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR STOCK67 OFF;
ALTER TABLE STOCK67 DROP CONSTRAINT STOCK67CKC;
ALTER TABLE STOCK67 ADD CONSTRAINT STOCK67CKC CHECK (S_W_ID BETWEEN 95371
AND 96815);
SET INTEGRITY FOR STOCK67 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR STOCK68 OFF;
ALTER TABLE STOCK68 DROP CONSTRAINT STOCK68CKC;
ALTER TABLE STOCK68 ADD CONSTRAINT STOCK68CKC CHECK (S_W_ID BETWEEN 96816
AND 98260);
SET INTEGRITY FOR STOCK68 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR STOCK69 OFF;
ALTER TABLE STOCK69 DROP CONSTRAINT STOCK69CKC;
ALTER TABLE STOCK69 ADD CONSTRAINT STOCK69CKC CHECK (S_W_ID BETWEEN 98261
AND 99705);
SET INTEGRITY FOR STOCK69 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR STOCK70 OFF;
ALTER TABLE STOCK70 DROP CONSTRAINT STOCK70CKC;
ALTER TABLE STOCK70 ADD CONSTRAINT STOCK70CKC CHECK (S_W_ID BETWEEN 99706
AND 101150);
SET INTEGRITY FOR STOCK70 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR STOCK71 OFF;
ALTER TABLE STOCK71 DROP CONSTRAINT STOCK71CKC;
ALTER TABLE STOCK71 ADD CONSTRAINT STOCK71CKC CHECK (S_W_ID BETWEEN 101151
AND 102595);
SET INTEGRITY FOR STOCK71 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR STOCK72 OFF;
ALTER TABLE STOCK72 DROP CONSTRAINT STOCK72CKC;
ALTER TABLE STOCK72 ADD CONSTRAINT STOCK72CKC CHECK (S_W_ID >= 102596);
SET INTEGRITY FOR STOCK72 ALL IMMEDIATE UNCHECKED;

```

```
connect reset;
```

11.1.43. CRCONST_CUSTOMER.ddl

```

connect to TPCC in share mode;
SET INTEGRITY FOR CUSTOMER1 OFF;
ALTER TABLE CUSTOMER1 DROP CONSTRAINT CUSTOMER1CKC;
ALTER TABLE CUSTOMER1 ADD CONSTRAINT CUSTOMER1CKC CHECK (C_W_ID BETWEEN 1
AND 1445);
SET INTEGRITY FOR CUSTOMER1 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR CUSTOMER2 OFF;
ALTER TABLE CUSTOMER2 DROP CONSTRAINT CUSTOMER2CKC;
ALTER TABLE CUSTOMER2 ADD CONSTRAINT CUSTOMER2CKC CHECK (C_W_ID BETWEEN
1446 AND 2890);
SET INTEGRITY FOR CUSTOMER2 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR CUSTOMER3 OFF;
ALTER TABLE CUSTOMER3 DROP CONSTRAINT CUSTOMER3CKC;
ALTER TABLE CUSTOMER3 ADD CONSTRAINT CUSTOMER3CKC CHECK (C_W_ID BETWEEN
2891 AND 4335);
SET INTEGRITY FOR CUSTOMER3 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR CUSTOMER4 OFF;
ALTER TABLE CUSTOMER4 DROP CONSTRAINT CUSTOMER4CKC;
ALTER TABLE CUSTOMER4 ADD CONSTRAINT CUSTOMER4CKC CHECK (C_W_ID BETWEEN
4336 AND 5780);
SET INTEGRITY FOR CUSTOMER4 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR CUSTOMER5 OFF;
ALTER TABLE CUSTOMER5 DROP CONSTRAINT CUSTOMER5CKC;
ALTER TABLE CUSTOMER5 ADD CONSTRAINT CUSTOMER5CKC CHECK (C_W_ID BETWEEN
5781 AND 7225);
SET INTEGRITY FOR CUSTOMER5 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR CUSTOMER6 OFF;
ALTER TABLE CUSTOMER6 DROP CONSTRAINT CUSTOMER6CKC;
ALTER TABLE CUSTOMER6 ADD CONSTRAINT CUSTOMER6CKC CHECK (C_W_ID BETWEEN
7226 AND 8670);
SET INTEGRITY FOR CUSTOMER6 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR CUSTOMER7 OFF;
ALTER TABLE CUSTOMER7 DROP CONSTRAINT CUSTOMER7CKC;
ALTER TABLE CUSTOMER7 ADD CONSTRAINT CUSTOMER7CKC CHECK (C_W_ID BETWEEN
8671 AND 10115);
SET INTEGRITY FOR CUSTOMER7 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR CUSTOMER8 OFF;
ALTER TABLE CUSTOMER8 DROP CONSTRAINT CUSTOMER8CKC;
ALTER TABLE CUSTOMER8 ADD CONSTRAINT CUSTOMER8CKC CHECK (C_W_ID BETWEEN
10116 AND 11560);
SET INTEGRITY FOR CUSTOMER8 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR CUSTOMER9 OFF;
ALTER TABLE CUSTOMER9 DROP CONSTRAINT CUSTOMER9CKC;
ALTER TABLE CUSTOMER9 ADD CONSTRAINT CUSTOMER9CKC CHECK (C_W_ID BETWEEN
11561 AND 13005);
SET INTEGRITY FOR CUSTOMER9 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR CUSTOMER10 OFF;
ALTER TABLE CUSTOMER10 DROP CONSTRAINT CUSTOMER10CKC;
ALTER TABLE CUSTOMER10 ADD CONSTRAINT CUSTOMER10CKC CHECK (C_W_ID BETWEEN
13006 AND 14450);
SET INTEGRITY FOR CUSTOMER10 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR CUSTOMER11 OFF;
ALTER TABLE CUSTOMER11 DROP CONSTRAINT CUSTOMER11CKC;
ALTER TABLE CUSTOMER11 ADD CONSTRAINT CUSTOMER11CKC CHECK (C_W_ID BETWEEN
14451 AND 15895);
SET INTEGRITY FOR CUSTOMER11 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR CUSTOMER12 OFF;
ALTER TABLE CUSTOMER12 DROP CONSTRAINT CUSTOMER12CKC;
ALTER TABLE CUSTOMER12 ADD CONSTRAINT CUSTOMER12CKC CHECK (C_W_ID BETWEEN
15896 AND 17340);
SET INTEGRITY FOR CUSTOMER12 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR CUSTOMER13 OFF;
ALTER TABLE CUSTOMER13 DROP CONSTRAINT CUSTOMER13CKC;
ALTER TABLE CUSTOMER13 ADD CONSTRAINT CUSTOMER13CKC CHECK (C_W_ID BETWEEN
17341 AND 18785);
SET INTEGRITY FOR CUSTOMER13 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR CUSTOMER14 OFF;
ALTER TABLE CUSTOMER14 DROP CONSTRAINT CUSTOMER14CKC;
ALTER TABLE CUSTOMER14 ADD CONSTRAINT CUSTOMER14CKC CHECK (C_W_ID BETWEEN
18786 AND 20230);
SET INTEGRITY FOR CUSTOMER14 ALL IMMEDIATE UNCHECKED;
connect reset;

```



```

ALTER TABLE NEW_ORDER62 DROP CONSTRAINT NEW_ORDER62CKC;
ALTER TABLE NEW_ORDER62 ADD CONSTRAINT NEW_ORDER62CKC CHECK (NO_W_ID
BETWEEN 88146 AND 89590);
SET INTEGRITY FOR NEW_ORDER62 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR NEW_ORDER63 OFF;
ALTER TABLE NEW_ORDER63 DROP CONSTRAINT NEW_ORDER63CKC;
ALTER TABLE NEW_ORDER63 ADD CONSTRAINT NEW_ORDER63CKC CHECK (NO_W_ID
BETWEEN 89591 AND 91035);
SET INTEGRITY FOR NEW_ORDER63 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR NEW_ORDER64 OFF;
ALTER TABLE NEW_ORDER64 DROP CONSTRAINT NEW_ORDER64CKC;
ALTER TABLE NEW_ORDER64 ADD CONSTRAINT NEW_ORDER64CKC CHECK (NO_W_ID
BETWEEN 91036 AND 92480);
SET INTEGRITY FOR NEW_ORDER64 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR NEW_ORDER65 OFF;
ALTER TABLE NEW_ORDER65 DROP CONSTRAINT NEW_ORDER65CKC;
ALTER TABLE NEW_ORDER65 ADD CONSTRAINT NEW_ORDER65CKC CHECK (NO_W_ID
BETWEEN 92481 AND 93925);
SET INTEGRITY FOR NEW_ORDER65 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR NEW_ORDER66 OFF;
ALTER TABLE NEW_ORDER66 DROP CONSTRAINT NEW_ORDER66CKC;
ALTER TABLE NEW_ORDER66 ADD CONSTRAINT NEW_ORDER66CKC CHECK (NO_W_ID
BETWEEN 93926 AND 95370);
SET INTEGRITY FOR NEW_ORDER66 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR NEW_ORDER67 OFF;
ALTER TABLE NEW_ORDER67 DROP CONSTRAINT NEW_ORDER67CKC;
ALTER TABLE NEW_ORDER67 ADD CONSTRAINT NEW_ORDER67CKC CHECK (NO_W_ID
BETWEEN 95371 AND 96815);
SET INTEGRITY FOR NEW_ORDER67 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR NEW_ORDER68 OFF;
ALTER TABLE NEW_ORDER68 DROP CONSTRAINT NEW_ORDER68CKC;
ALTER TABLE NEW_ORDER68 ADD CONSTRAINT NEW_ORDER68CKC CHECK (NO_W_ID
BETWEEN 96816 AND 98260);
SET INTEGRITY FOR NEW_ORDER68 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR NEW_ORDER69 OFF;
ALTER TABLE NEW_ORDER69 DROP CONSTRAINT NEW_ORDER69CKC;
ALTER TABLE NEW_ORDER69 ADD CONSTRAINT NEW_ORDER69CKC CHECK (NO_W_ID
BETWEEN 98261 AND 99705);
SET INTEGRITY FOR NEW_ORDER69 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR NEW_ORDER70 OFF;
ALTER TABLE NEW_ORDER70 DROP CONSTRAINT NEW_ORDER70CKC;
ALTER TABLE NEW_ORDER70 ADD CONSTRAINT NEW_ORDER70CKC CHECK (NO_W_ID
BETWEEN 99706 AND 101150);
SET INTEGRITY FOR NEW_ORDER70 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR NEW_ORDER71 OFF;
ALTER TABLE NEW_ORDER71 DROP CONSTRAINT NEW_ORDER71CKC;
ALTER TABLE NEW_ORDER71 ADD CONSTRAINT NEW_ORDER71CKC CHECK (NO_W_ID
BETWEEN 101151 AND 102595);
SET INTEGRITY FOR NEW_ORDER71 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR NEW_ORDER72 OFF;
ALTER TABLE NEW_ORDER72 DROP CONSTRAINT NEW_ORDER72CKC;
ALTER TABLE NEW_ORDER72 ADD CONSTRAINT NEW_ORDER72CKC CHECK (NO_W_ID >=
102596);
SET INTEGRITY FOR NEW_ORDER72 ALL IMMEDIATE UNCHECKED;
connect reset;

```

11.1.47. 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 1445);
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 1446
AND 2890);
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 2891
AND 4335);
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 4336
AND 5780);
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 5781
AND 7225);
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 7226
AND 8670);
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 8671
AND 10115);
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
10116 AND 11560);
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
11561 AND 13005);
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
13006 AND 14450);
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
14451 AND 15895);
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
15896 AND 17340);
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
17341 AND 18785);
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
18786 AND 20230);
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
20231 AND 21675);
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
21676 AND 23120);
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
23121 AND 24565);
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
24566 AND 26010);
SET INTEGRITY FOR HISTORY18 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;

```



```

67916 AND 69360);
SET INTEGRITY FOR HISTORY48 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY49 OFF;
ALTER TABLE HISTORY49 DROP CONSTRAINT HISTORY49CKC;
ALTER TABLE HISTORY49 ADD CONSTRAINT HISTORY49CKC CHECK (H_W_ID BETWEEN
69361 AND 70805);
SET INTEGRITY FOR HISTORY49 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY50 OFF;
ALTER TABLE HISTORY50 DROP CONSTRAINT HISTORY50CKC;
ALTER TABLE HISTORY50 ADD CONSTRAINT HISTORY50CKC CHECK (H_W_ID BETWEEN
70806 AND 72250);
SET INTEGRITY FOR HISTORY50 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY51 OFF;
ALTER TABLE HISTORY51 DROP CONSTRAINT HISTORY51CKC;
ALTER TABLE HISTORY51 ADD CONSTRAINT HISTORY51CKC CHECK (H_W_ID BETWEEN
72251 AND 73695);
SET INTEGRITY FOR HISTORY51 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY52 OFF;
ALTER TABLE HISTORY52 DROP CONSTRAINT HISTORY52CKC;
ALTER TABLE HISTORY52 ADD CONSTRAINT HISTORY52CKC CHECK (H_W_ID BETWEEN
73696 AND 75140);
SET INTEGRITY FOR HISTORY52 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY53 OFF;
ALTER TABLE HISTORY53 DROP CONSTRAINT HISTORY53CKC;
ALTER TABLE HISTORY53 ADD CONSTRAINT HISTORY53CKC CHECK (H_W_ID BETWEEN
75141 AND 76585);
SET INTEGRITY FOR HISTORY53 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY54 OFF;
ALTER TABLE HISTORY54 DROP CONSTRAINT HISTORY54CKC;
ALTER TABLE HISTORY54 ADD CONSTRAINT HISTORY54CKC CHECK (H_W_ID BETWEEN
76586 AND 78030);
SET INTEGRITY FOR HISTORY54 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY55 OFF;
ALTER TABLE HISTORY55 DROP CONSTRAINT HISTORY55CKC;
ALTER TABLE HISTORY55 ADD CONSTRAINT HISTORY55CKC CHECK (H_W_ID BETWEEN
78031 AND 79475);
SET INTEGRITY FOR HISTORY55 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY56 OFF;
ALTER TABLE HISTORY56 DROP CONSTRAINT HISTORY56CKC;
ALTER TABLE HISTORY56 ADD CONSTRAINT HISTORY56CKC CHECK (H_W_ID BETWEEN
79476 AND 80920);
SET INTEGRITY FOR HISTORY56 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY57 OFF;
ALTER TABLE HISTORY57 DROP CONSTRAINT HISTORY57CKC;
ALTER TABLE HISTORY57 ADD CONSTRAINT HISTORY57CKC CHECK (H_W_ID BETWEEN
80921 AND 82365);
SET INTEGRITY FOR HISTORY57 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY58 OFF;
ALTER TABLE HISTORY58 DROP CONSTRAINT HISTORY58CKC;
ALTER TABLE HISTORY58 ADD CONSTRAINT HISTORY58CKC CHECK (H_W_ID BETWEEN
82366 AND 83810);
SET INTEGRITY FOR HISTORY58 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY59 OFF;
ALTER TABLE HISTORY59 DROP CONSTRAINT HISTORY59CKC;
ALTER TABLE HISTORY59 ADD CONSTRAINT HISTORY59CKC CHECK (H_W_ID BETWEEN
83811 AND 85255);
SET INTEGRITY FOR HISTORY59 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY60 OFF;
ALTER TABLE HISTORY60 DROP CONSTRAINT HISTORY60CKC;
ALTER TABLE HISTORY60 ADD CONSTRAINT HISTORY60CKC CHECK (H_W_ID BETWEEN
85256 AND 86700);
SET INTEGRITY FOR HISTORY60 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY61 OFF;
ALTER TABLE HISTORY61 DROP CONSTRAINT HISTORY61CKC;
ALTER TABLE HISTORY61 ADD CONSTRAINT HISTORY61CKC CHECK (H_W_ID BETWEEN
86701 AND 88145);
SET INTEGRITY FOR HISTORY61 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY62 OFF;
ALTER TABLE HISTORY62 DROP CONSTRAINT HISTORY62CKC;
ALTER TABLE HISTORY62 ADD CONSTRAINT HISTORY62CKC CHECK (H_W_ID BETWEEN
88146 AND 89590);
SET INTEGRITY FOR HISTORY62 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY63 OFF;

```

```

ALTER TABLE HISTORY63 DROP CONSTRAINT HISTORY63CKC;
ALTER TABLE HISTORY63 ADD CONSTRAINT HISTORY63CKC CHECK (H_W_ID BETWEEN
89591 AND 91035);
SET INTEGRITY FOR HISTORY63 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY64 OFF;
ALTER TABLE HISTORY64 DROP CONSTRAINT HISTORY64CKC;
ALTER TABLE HISTORY64 ADD CONSTRAINT HISTORY64CKC CHECK (H_W_ID BETWEEN
91036 AND 92480);
SET INTEGRITY FOR HISTORY64 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY65 OFF;
ALTER TABLE HISTORY65 DROP CONSTRAINT HISTORY65CKC;
ALTER TABLE HISTORY65 ADD CONSTRAINT HISTORY65CKC CHECK (H_W_ID BETWEEN
92481 AND 93925);
SET INTEGRITY FOR HISTORY65 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY66 OFF;
ALTER TABLE HISTORY66 DROP CONSTRAINT HISTORY66CKC;
ALTER TABLE HISTORY66 ADD CONSTRAINT HISTORY66CKC CHECK (H_W_ID BETWEEN
93926 AND 95370);
SET INTEGRITY FOR HISTORY66 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY67 OFF;
ALTER TABLE HISTORY67 DROP CONSTRAINT HISTORY67CKC;
ALTER TABLE HISTORY67 ADD CONSTRAINT HISTORY67CKC CHECK (H_W_ID BETWEEN
95371 AND 96815);
SET INTEGRITY FOR HISTORY67 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY68 OFF;
ALTER TABLE HISTORY68 DROP CONSTRAINT HISTORY68CKC;
ALTER TABLE HISTORY68 ADD CONSTRAINT HISTORY68CKC CHECK (H_W_ID BETWEEN
96816 AND 98260);
SET INTEGRITY FOR HISTORY68 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY69 OFF;
ALTER TABLE HISTORY69 DROP CONSTRAINT HISTORY69CKC;
ALTER TABLE HISTORY69 ADD CONSTRAINT HISTORY69CKC CHECK (H_W_ID BETWEEN
98261 AND 99705);
SET INTEGRITY FOR HISTORY69 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY70 OFF;
ALTER TABLE HISTORY70 DROP CONSTRAINT HISTORY70CKC;
ALTER TABLE HISTORY70 ADD CONSTRAINT HISTORY70CKC CHECK (H_W_ID BETWEEN
99706 AND 101150);
SET INTEGRITY FOR HISTORY70 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY71 OFF;
ALTER TABLE HISTORY71 DROP CONSTRAINT HISTORY71CKC;
ALTER TABLE HISTORY71 ADD CONSTRAINT HISTORY71CKC CHECK (H_W_ID BETWEEN
101151 AND 102595);
SET INTEGRITY FOR HISTORY71 ALL IMMEDIATE UNCHECKED;
connect reset;
connect to TPCC in share mode;
SET INTEGRITY FOR HISTORY72 OFF;
ALTER TABLE HISTORY72 DROP CONSTRAINT HISTORY72CKC;
ALTER TABLE HISTORY72 ADD CONSTRAINT HISTORY72CKC CHECK (H_W_ID >=
102596);
SET INTEGRITY FOR HISTORY72 ALL IMMEDIATE UNCHECKED;
connect reset;

```

11.1.48. CRIDX_CUST_INDXB.ddl

```

connect to TPCC in share mode;
DROP INDEX CUST_IDXB1;
CREATE INDEX CUST_IDXB1
ON CUSTOMER1(C_LAST, C_W_ID, C_D_ID, C_FIRST,
C_ID) PCTFREE 0;
connect reset;
connect to TPCC in share mode;
DROP INDEX CUST_IDXB2;
CREATE INDEX CUST_IDXB2
ON CUSTOMER2(C_LAST, C_W_ID, C_D_ID, C_FIRST,
C_ID) PCTFREE 0;
connect reset;
connect to TPCC in share mode;
DROP INDEX CUST_IDXB3;
CREATE INDEX CUST_IDXB3
ON CUSTOMER3(C_LAST, C_W_ID, C_D_ID, C_FIRST,
C_ID) PCTFREE 0;
connect reset;
connect to TPCC in share mode;
DROP INDEX CUST_IDXB4;
CREATE INDEX CUST_IDXB4
ON CUSTOMER4(C_LAST, C_W_ID, C_D_ID, C_FIRST,
C_ID) PCTFREE 0;
connect reset;
connect to TPCC in share mode;
DROP INDEX CUST_IDXB5;
CREATE INDEX CUST_IDXB5
ON CUSTOMER5(C_LAST, C_W_ID, C_D_ID, C_FIRST,
C_ID) PCTFREE 0;
connect reset;
connect to TPCC in share mode;

```



```

ALTER BUFFERPOOL D13 DEFERRED SIZE 1652;
ALTER BUFFERPOOL D14 DEFERRED SIZE 1652;
ALTER BUFFERPOOL D15 DEFERRED SIZE 1652;
ALTER BUFFERPOOL D16 DEFERRED SIZE 1652;
ALTER BUFFERPOOL D17 DEFERRED SIZE 1652;
ALTER BUFFERPOOL D18 DEFERRED SIZE 1652;
ALTER BUFFERPOOL C1 DEFERRED SIZE 520200;
ALTER BUFFERPOOL C2 DEFERRED SIZE 520200;
ALTER BUFFERPOOL C3 DEFERRED SIZE 520200;
ALTER BUFFERPOOL C4 DEFERRED SIZE 520200;
ALTER BUFFERPOOL C5 DEFERRED SIZE 520200;
ALTER BUFFERPOOL C6 DEFERRED SIZE 520200;
ALTER BUFFERPOOL C7 DEFERRED SIZE 520200;
ALTER BUFFERPOOL C8 DEFERRED SIZE 520200;
ALTER BUFFERPOOL C9 DEFERRED SIZE 520200;
ALTER BUFFERPOOL C10 DEFERRED SIZE 520200;
ALTER BUFFERPOOL C11 DEFERRED SIZE 520200;
ALTER BUFFERPOOL C12 DEFERRED SIZE 520200;
ALTER BUFFERPOOL C13 DEFERRED SIZE 520200;
ALTER BUFFERPOOL C14 DEFERRED SIZE 520200;
ALTER BUFFERPOOL C15 DEFERRED SIZE 520200;
ALTER BUFFERPOOL C16 DEFERRED SIZE 520200;
ALTER BUFFERPOOL C17 DEFERRED SIZE 520200;
ALTER BUFFERPOOL C18 DEFERRED SIZE 520200;
ALTER BUFFERPOOL I DEFERRED SIZE 1242;
ALTER BUFFERPOOL OL1 DEFERRED SIZE 231200;
ALTER BUFFERPOOL OL2 DEFERRED SIZE 231200;
ALTER BUFFERPOOL OL3 DEFERRED SIZE 231200;
ALTER BUFFERPOOL OL4 DEFERRED SIZE 231200;
ALTER BUFFERPOOL OL5 DEFERRED SIZE 231200;
ALTER BUFFERPOOL OL6 DEFERRED SIZE 231200;
ALTER BUFFERPOOL OL7 DEFERRED SIZE 231200;
ALTER BUFFERPOOL OL8 DEFERRED SIZE 231200;
ALTER BUFFERPOOL OL9 DEFERRED SIZE 231200;
ALTER BUFFERPOOL OL10 DEFERRED SIZE 231200;
ALTER BUFFERPOOL OL11 DEFERRED SIZE 231200;
ALTER BUFFERPOOL OL12 DEFERRED SIZE 231200;
ALTER BUFFERPOOL OL13 DEFERRED SIZE 231200;
ALTER BUFFERPOOL OL14 DEFERRED SIZE 231200;
ALTER BUFFERPOOL OL15 DEFERRED SIZE 231200;
ALTER BUFFERPOOL OL16 DEFERRED SIZE 231200;
ALTER BUFFERPOOL OL17 DEFERRED SIZE 231200;
ALTER BUFFERPOOL OL18 DEFERRED SIZE 231200;
ALTER BUFFERPOOL C21 DEFERRED SIZE 208080;
ALTER BUFFERPOOL C22 DEFERRED SIZE 208080;
ALTER BUFFERPOOL C23 DEFERRED SIZE 208080;
ALTER BUFFERPOOL C24 DEFERRED SIZE 208080;
ALTER BUFFERPOOL C25 DEFERRED SIZE 208080;
ALTER BUFFERPOOL C26 DEFERRED SIZE 208080;
ALTER BUFFERPOOL C27 DEFERRED SIZE 208080;
ALTER BUFFERPOOL C28 DEFERRED SIZE 208080;
ALTER BUFFERPOOL C29 DEFERRED SIZE 208080;
ALTER BUFFERPOOL C210 DEFERRED SIZE 208080;
ALTER BUFFERPOOL C211 DEFERRED SIZE 208080;
ALTER BUFFERPOOL C212 DEFERRED SIZE 208080;
ALTER BUFFERPOOL C213 DEFERRED SIZE 208080;
ALTER BUFFERPOOL C214 DEFERRED SIZE 208080;
ALTER BUFFERPOOL C215 DEFERRED SIZE 208080;
ALTER BUFFERPOOL C216 DEFERRED SIZE 208080;
ALTER BUFFERPOOL C217 DEFERRED SIZE 208080;
ALTER BUFFERPOOL C218 DEFERRED SIZE 208080;
connect reset;

```

11.2. Data Generation

11.2.1. Makefile.config

```

#####
###
## Licensed Materials - Property of IBM
##
## (C) COPYRIGHT International Business Machines Corp. 1996, 2010
## 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=;

```

11.2.2. Src.Common/Makefile

```

#####
###
## Licensed Materials - Property of IBM
##
## (C) COPYRIGHT International Business Machines Corp. 1996, 2010
## 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
# #####
PRP_OPTS = PACKAGE \
           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$(TPCC_SPTYPE)
UTIL_OBJ_DBG = tpccdbg$(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
#
# #####
# Build Rules
# #####
.SUFFIXES:
.SUFFIXES: $(OBJEXT) .c .sqc
.sqc.c:
@echo "Prepping $*.sqc"
db2 prep $*.sqc $(PRP_OPTS) bindfile
db2 grant execute on package TPCCCTX to public
#
# #####
# Dependencies
# #####

```



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

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

11.2.3. Src.Common/tpccmisc.c

```
/*
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1996, 2010
** 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));
}

```

11.2.4. dbgen/Makefile

```
#####
###
## Licensed Materials - Property of IBM
##
## (C) COPYRIGHT International Business Machines Corp. 1996, 2010
## 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) $(CFLAGS_DEBUG)

LDFLAGS = $(LDFLAGS_EXEC) $(LDFLAGS_LIB)

# #####
# File Collections
# #####

OBJS = tpccrnd$(OBJEXT) \
       $(TPCC_ROOT)/Src.Common/tpccmisc$(OBJEXT)
OBJ_EEE = $(TPCC_ROOT)/Src.Common/tpccihw$(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) $(LDFLAGS_OUT)$(OBJEXT)

# #####
# 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
```

11.2.5. dbgen/gendata.c

```
/*
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1996, 2010
** 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 <ctype.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 quiet_mode = 0;
sqlint32 ware_start = 1;
sqlint32 ware_end = WAREHOUSES;

char fmtWare[] = "%s|%s|%s|%s|%s|%s|04.4f|%.2f|\n";
char fmtDist[] = "%d|04.4f|%.2f|s|s|s|s|s|s|s|s|s|s|s|\n";
char fmtItem[] = "%s|.2f|s|s|s|s|\n";
char fmtStock[] = "%d|d|d|d|d|s|s|s|s|s|s|s|s|s|s|s|s|s|\n";
char fmtCust[] = "%d|s|s|s|s|.2f|s|s|04.4f|s|s|s|s|s|s|s|s|s|d|d|d|.2f|.2f|\n";
char fmtHist[] = "%d|d|d|d|d|d|s|.2f|s|\n";
char fmtOrder[] = "%d|s|s|d|d|d|d|d|d|d|d|\n";
char fmtOLine[] = "%s|.2f|d|d|d|d|s|d|d|d|d|d|\n";
char fmtNewOrd[] = "%d|d|d|\n";
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;

    /* ***** */
    /* Process Command Line Arguments          */
    /* ***** */

    /* Valid Command Line Options
    *-----*/
    * Table Option:          -t <table>          (-t 3 for warehouse)
    * 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, -r, -s and -q options are optional.
    *
    * 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], "-f") == 0) ||
                 (strcmp(argv[i], "-f1") == 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 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();
elapsed = timestamp2 - timestamp1;
if (rc == 0) {
    if (!quiet_mode) {
        fprintf(stdout, "\nITEM table generated in %8.2f
seconds.\n\n", elapsed);
        fflush(stdout);
    }
} else {
    fprintf(stderr, "\nITEM table FAILED at (I %d) after %8.2f
seconds.\n\n", item_num, elapsed);
    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, outtypel, outnamel);
    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();
    elapsed = timestamp2 - timestamp1;
    if (rc == 0) {
        if (!quiet_mode) {
            fprintf(stdout, "\nSTOCK table generated in %8.2f
seconds.\n\n", elapsed);
            fflush(stdout);
        }
    } else {
        fprintf(stderr, "\nSTOCK table FAILED at (S %d W %d) after %8.2f
seconds.\n\n", stock_num, ware_num, elapsed);
        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, outtypel, outnamel);
    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, elapse);
    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, outtypel, 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 = 30000.00;

            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();
    elapse = timestamp2 - timestamp1;
    if (rc == 0) {
        if (!quiet_mode) {
            fprintf(stdout, "\nDISTRICT table generated in %8.2f
seconds.\n\n", elapse);
            fflush(stdout);
        }
    } else {
        fprintf(stderr, "\nDISTRICT table FAILED at (W %d D %d) after %8.2f
seconds.\n\n", ware_num, dist_num, elapse);
        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, outtypel, 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 */
            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;

            /* -----
            // As per Francois' request, we dump the vaue of C_C_LAST_LOAD
            // in the C_FIRST field of the first customer.
            // This makes is simple to validate what value of
            C_C_LAST_LOAD
            // was used to generate the database.
            // -----

            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, outtypel, outnamel);
    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, outtypel, outnamel);
    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 ord_r_carrier_id;
    sqlint32 ord_r_ol_cnt;
    sqlint32 oline_ol_num;
    sqlint32 oline_item_num;

    double oline_amount;
    char oline_dist_info[25];

    IOH_NUM numBytes;
    ioHandle hndl, hnd2;
    char Buffer[1024];

    char currtmstp[27];
    char nulltmstp[27] = "0001-01-01 00:00:00";

    oline_dist_info[24] = '\0';

    timestamp1 = current_time();

    rc1 = GenericOpen(&hndl, outtype1, outname1);
    if (rc1 != 0) { goto ool_done; }
    rc2 = GenericOpen(&hnd2, outtype2, outname2);
    if (rc2 != 0) { goto ool_done; }

    createTimestampString(currtmstp);

    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\t", dist_num);
                fflush(stdout);
            }

            seed_1_3000();

            for (ord_num = 1; ord_num <= CUSTOMERS_PER_DISTRICT; ord_num++)
            {
                if (ord_num < 2101)
                    ord_r_carrier_id = rand_integer( 1, 10 );
                else
                    ord_r_carrier_id = 0;

                cust_num = random_1_3000();
                ord_r_ol_cnt = rand_integer(MIN_OL_PER_ORDER,MAX_OL_PER_ORDER);

                numBytes = sprintf(Buffer, fmtOrdr,
                    cust_num,
                    currtmstp,
                    ord_r_carrier_id,
                    ord_r_ol_cnt,
                    1,
                    ord_num,
                    ware_num,
                    dist_num);

                rc1 = GenericWrite(&hndl, Buffer, numBytes);
                if (rc1 != 0) { goto ool_done; }

                for ( oline_ol_num = 1; oline_ol_num <= ord_r_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) ? currtmstp :
nulltmstp),
                        ((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(&hndl);

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);
    }
}

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");
}

```

11.2.6. dbgen/tpccrnd.c

```

/*-----*/
****
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1996, 2010
** 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"

#include <stdlib.h>

static char tbl_cust[CUSTOMERS_PER_DISTRICT];

static char alnum[] =
    "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNQPQRSTUVWXYZ";

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);
}

/*
*****
*
*   NUrnd_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 NUrval ( 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 = NUrval( 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));
}

```

11.2.7. include/lval.h

```

/* lval.h - generated automatically at 20130211.1629 */

```

```

#ifndef __LVAL_H
#define __LVAL_H
#define WAREHOUSES 104040
#define DISTRICTS_PER_WAREHOUSE 10
#define CUSTOMERS_PER_DISTRICT 3000
#define ITEMS 100000
#define STOCK_PER_WAREHOUSE 100000
#define MIN_ORDER 5
#define MAX_ORDER 15
#define NU_ORDERS_PER_DISTRICT 900
#endif // __LVAL_H

```

11.2.8. include/platform.h

```

/*****
****
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1996, 2010
** 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 <fontl.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>
#include <unistd.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

11.2.9. include/tpccmisc.h

/*
*****
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1996, 2010
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****
**/

/*
** tpccmisc.h - Miscellanouse Routines
**
*/

#ifdef __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

```

11.2.10. include/tpccrnd.h

```

/*****
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1996, 2010
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****/

/*
 * tpccrnd.h - Random generation functions for TPC-C
 */

#ifndef __TPCCRND_H
#define __TPCCRND_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 NUrnd_val( int A, int val_lo, int val_hi, int C );

void seed_l_3000( void );
int random_l_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 // __TPCCRND_H

```

11.2.11. tpccenv.sh

```

#!/bin/sh

#####
###
### Licensed Materials - Property of IBM
###
### (C) COPYRIGHT International Business Machines Corp. 1996, 2010
### 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=CK101019

# The DB2 Instance Name (for DB2)
export DB2INSTANCE=${USER}

# The OS being used (i.e. "UNIX", "LINUX", "WINDOWS")
export PLATFORM=UNIX
export SERVER_PLATFORM=UNIX

# 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 DARIVERISION or NONDARI;
#export TPCC_SPTYPE=NOSP
#export TPCC_SPTYPE=SPGENERAL2
#export TPCC_SPTYPE=SPGENERAL
#export TPCC_SPTYPE=DARI2SQLDA

# The schema name is typically the SQL authorization ID (or username).
# This is required for runstats and EEE.
export TPCC_SCHEMA=${USER}
export SERVER_TPCC_SCHEMA=${USER}

# DB2 EE/EEE Configuration
export DB2EDITION=EE
#export DB2EDITION=DPF

# 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

February 1, 2013

IBM Corporation
Ray Venditti
11501 Burnet Road
Austin, TX 78758

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
LWA-00984	Windows Server 2008 R2 Web Server Edition <i>Full License</i> <i>No Discounts Applied</i>	\$469.00	4	\$1,876.00
127-00166	Microsoft Visual Studio 2008 Professional <i>Full License</i> <i>No Discounts Applied</i>	\$250.00	1	\$250.00
N/A	Microsoft Problem Resolution Services <i>Professional Support</i> <i>(1 Incident).</i>	\$259.00	1	\$259.00

Windows Server 2008 R2 Web Edition is currently orderable and available through Microsoft's normal distribution channels. A list of Microsoft's resellers can be found in the Microsoft Product Information Center at

<http://www.microsoft.com/products/info/render.aspx?view=22&type=how>

Defect support is included in the purchase price. Additional support is available from Microsoft PSS on an incident by incident basis at \$259.00 call.

This quote is valid for the next 90 days.

Reference ID: TPCC_qhttplylGYLKTBTDFhkiNlloJikjLj3372.



RED HAT STORE

PRODUCTS



SOLUTIONS

SUPPORT

TRAINING

CONSULTING

Shopping Cart

Item	Quantity	Price	Line Total	
New Subscription Contract February 11, 2013 - February 10, 2014				
Red Hat Enterprise Linux Server, Premium (1-2 sockets) (Up to 1 guest) February 11, 2013 - February 10, 2014	3  	Remove	\$1,299.00	\$3,897.00
Promotion Code: <input type="text"/>	Update Cart	Subtotal:	\$3,897.00	

If you are purchasing a subscription to support Red Hat Enterprise Linux 4, please contact [customer service](#) (requires [Extended Lifecycle Support](#) -- not available online).

[Continue shopping](#)[Continue to Checkout](#)

Qty.	Product Description	Savings	Total Price
3	 <p>Acer S181HLGb Black 18.5" 5ms Widescreen LED Backlight LCD Monitor Item #: N82E16824009391 Return Policy: Monitor Standard Return Policy</p>		<p>\$287.97 (\$95.99 each)</p>
3	 <p>NETGEAR 48 Port Gigabit Smart Switch w/ 4 Combo ports - Lifetime Warranty (GS748T) Item #: N82E16833122147 Return Policy: Networking Extended Return Policy</p>	-\$100.00 Instant	<p>\$1,599.97 \$1,499.97 (\$499.99 each)</p>
3	 <p>Kensington K72436AM Black USB Wired Standard Keyboard for Life Desktop Set Item #: N82E16823155065 Return Policy: Standard Return Policy</p>		<p>\$59.97 (\$19.99 each)</p>
3	 <p>powercom KIN-1500AP 1500 VA 900 Watts UPS Item #: N82E16842106115 Return Policy: Standard Return Policy</p>		<p>\$449.97 (\$149.99 each)</p>
Subtotal:			\$2,297.88