TPC Benchmark™ C

Full Disclosure Report

Second Edition

31–May–2017

Using

Goldilocks v3.1 Standard Edition

on

Jet-speed™ HHA2212

Second Edition: 31-May-2017

TTA, Telecommunications Technology Association, believes that all the information in this document is accurate as of the publication date. The information in this document is subject to change without notice. TTA, the sponsor of this benchmart test, assumes no responsibility for any errors that may appear in this document. The pricing information in this document is believed to accurately reflect the current prices as of the publication date. However, the sponsor provide no warranty of the pricing information in this document.

The performance are highgly dependant upon many of SW and HW factors; relative result may vary due to those of changes. The sponsor does not warrant or represent that a user can or will achieve similar performance result. No warranty of system performance or price/performance is expressed or implied in this report

Trademarks

The following termws 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*
- *TTA is a registered trademarks of Telecommunications Technology Association*
- *Goldilocks is a registered trademarks of SUNJESOFT, Inc.*
- *Jet-Speed is a registered trademarks of Taejin Infotech Co., LTD.*
- *JBoss is a registered trademarks of RedHat, Inc.*
- *Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation.*
- *All other trademarks and copyrights are property of their respective owners.*

Table of Contents

Abstract

This report documents the methodology and results of the TPC Benchmark™ C (TPC-C) test conducted on the Goldilocks v3.1 Standard Edition on Jet-speed™ HHA2212

Goldilocks v3.1 Standard Edition on TJS212

Company Name	System Name	Database Software	Opeating System
Telecommunications Technology Association	Jet-speed™ HHA2212	Goldilocks v3.1 Standard Edition	CentOS 6.6

TPC Benchmark™ C Metrics

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date
₩ 241,936,000 (KRW)	139,909 tpmC	1,730 KRW/tpmC	09-May-2017

Preface

The Transaction Processing Performance Council (TPC™) is a non-profit corporation founded to define transaction processing and database benchmarks and to disseminate objective, verifiable TPC performance data to the industry. The TPC Benchmark© C is an on-line transaction processing benchmark (OLTP) developed by the TPC.

TPC Benchmark™ C Overview

TPC Benchmark™ C (TPC-C) simulates a complete computing environment where a population of users executes transactions against a database. The benchmark is centered around the principal activities (transactions) of an order-entry environment. These transactions include entering and delivering orders, recording payments, checking the status of orders, and monitoring the level of stock at the warehouses. While the benchmark portrays the activity of a wholesale supplier, TPC-C is not limited to the activity of any particular business segment, but, rather represents any industry that must manage, sell, or distribute a product or service.

TPC-C consists of a mixture of read-only and update intensive transactions that simulate the activities found in complex OLTP application environments. It does so by exercising a breadth of system components associated with such environments, which are characterized by:

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

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

TPC-C uses terminology and metrics that are similar to other benchmarks, originated by the TPC or others. Such similarity in terminology does not in any way imply that TPC-C results are comparable to other benchmarks. The only benchmark results comparable to TPC-C are other TPC-C results conformant with the same revision.

Despite the fact that this benchmark offers a rich environment that emulates many OLTP applications, this benchmark does not reflect the entire range of OLTP requirements. In addition, the extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-C approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments. Extrapolations to other environments are not recommended.

Benchmark results are highly dependent upon workload, specific application requirements, and systems design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC-C should not be used as a substitute for a specific customer application benchmark when critical capacity planning and/or product evaluation decisions are contemplated.

Further information is available at www.tpc.org

	Goldilocks v3.1 Standard Edition on Jet-speed™ HHA2212			**TPC-C Version 5.11.0** **TPC Pricing 2.1.0** **Report Date** 09-May-2017 **Revised Date** 31-May-2017
Total System Cost	TPC-C Throughput	Price/Performance		Availability Date
₩ 241,936,000 (KRW)	**139,909 tpmC**	**1,730 KRW/tpmC**		**09-May-2017**
Server Processors/Cores/Threads	Database Manager Operating System	Operating System	Other Software	Number of Users
2/16/16	**Goldilocks v3.1** **Standard Edition**	**CentOS 6.6**	**JBOSS**	**110,000**

DB server

HP 1420-24G-2SFP
(Ethernet 1G switch)

WAS server

1 x DB Server Based Jet-speed™ HHA2212
- 2 x Intel Xeon E5-2630v3 2.4GHz
- 24 x 64GB DIMM (1.5TB RAM)
- 2 x 300GB SAS 15K rpm
- 2 x FC HBA 8Gb
- 1 x Intel NIC X520-RS2

1 x Storage JS2800
- 8 x DRAM-SSD 64GB
- 1 x PCI-Express External Switch Card

1 x Storage NGS500
- 10 x 1.2TB SAS 10K
- 2 x FC 8Gb Target Port

3 x WAS Server TJS104
- 2 x Intel Xeon E5-2630v3 2.4GHz
- 2 x 300GB SAS 15K
- 2 x 16GB DDR4 RAM
- 1 x Intel NIC x520-SR2

System Components	DB Server		WAS Server	
	Quantity	**Description**	**Quantity**	**Description**
Processors/Cores/Threads	2/16/16	2.4GHz Xeon E5-2630V3	2/16/32	2.4GHz Xeon E5-2630V3
Memory	24	64GB	2	16GB
Storage Controller	1 2	SAS 12G Raid 8 Gb FC HBA	1	SAS 12G Raid
Storage Device	2 10 8	300GB SAS HDD (int.) 1.2TB SAS HDD 64GB DRAM-SSD	2	300GB
Total Storage Capacity		13.1 TB		

	Goldilocks v3.1 Standard Edition on Jet-speed™ HHA2212	TPC-C Version 5.11.0 TPC Pricing 2.1.0 Report Date: 09-May-2017 Revised Date: 31-May-2017 Availability Date: 09-May-2017

Description	Part Number	Source	Unit Price	Qty	Price	3-Yr. Maint. Price
Server Hardware						
DB Server Based Jet-speed™ HHA2212	HHA2212	1	36,500,000	1	36,500,000	
HHA2212 Barebone Kit with 800W Redundant PSU	92002-0008-00	1	(include)	1		
E5-2630V3 Intel Xeon 2.4GHz 8_Core L3_20M	20104-0004-00	1	(include)	2		
Memory 64GB DDR4 PC4-2133 ECC Reg.	20231-0004-00	1	(include)	24		
HDD SAS 300GB 15k	20320-0013-00	1	(include)	2		
HBA QLE-2562 8Gb	22201-0006-00	1	(include)	1		
SAS/SATA 8ch 12G LSI MegaRAID 9361-8i	22204-0017-00	1	(include)	1		
NIC Intel X520-SR2	22202-0104-00	1	(include)	1		
UTP CAT5e Ethernet Cable 1M	42050-0001-00	1	(include)	1		
Power Cord, NICETECH, 2.5M	42119-0005-00	1	(include)	2		
DELL KB216 Eng/Kor Keyboard	91010-0001-00	1	(include)	1		
Optical Mouse, Two Buttons, USB	91010-0002-00	1	(include)	1		
Monitor 27 inch	91009-0001-01	1	(include)	1		
3-yrs 24x7x4hrs NBD Onsite Support Service		1	5,700,000	1		5,700,000
3 x WAS Servers (per server)	TJS104	1	5,500,000	3	16,500,000	
TJS104 Barebone Kit with 800W Redundant PSU	92002-0010-00	1	(include)	1		
E5-2630V3 Intel Xeon 2.4GHz 8_Core L3_20M	20104-0004-00	1	(include)	2		
Memory Memory 16GB DDR4 PC4-2133 ECC Reg.	20231-0002-00	1	(include)	2		
HDD SAS 300GB 15k	20320-0014-00	1	(include)	2		
NIC Intel X520-SR2	22202-0104-00	1	(include)	1		
UTP CAT5e Ethernet Cable 1M	42050-0001-00	1	(include)	1		
Power Cord, NICETECH, 2.5M	42119-0005-00	1	(include)	2		
3-yrs 24x7x4hrs Onsite Support Service		1	2,640,000	1		2,640,000
Sub Total					**53,000,000**	**8,340,000**
Storage Hardware						
NGS500	NGS500	1	41,580,000	1	41,580,000	
NGS500 Controller with 800W Redundant PSU	92004-0001-00	1	(include)	1		
SSD SATA 480 6G	20320-0015-00	1	(include)	2		
SAS 1.2TB 10K	22201-0006-02	1	(include)	10		
FC 8Gb Target Port	61002-0001-01	1	(include)	2		
Network Management Port	22201-0006-02	1	(include)	1		
OM3 LC-LC 5M Cable	42040-0003-00	1	(include)	2		
UTP CAT5e Ethernet Cable 1M	61001-0001-00	1	(include)	1		
Storage Management SW		1	(include)	1		
Power Cord, NICETECH, 2.5M	42119-0005-00	1	(include)	2		
3-yrs 24x7x4hrs Onsite Support Service		1	9,900,000	1		9,900,000

JS2800	JS2800	1	23,160,000	1	23,160,000	
JS2800 Barebone Kit with 550W Redundant PSU	92002-0018-00	1	(include)	1		
DRAM-SSD 64GB	30102-0002-00	1	(include)	8		
PCI-Express External Switch Card	30120-0009-00	1	(include)	1		
PCI-Express External Cable	42122-0001-00	1	(include)	1		
Power Cord, NICETECH, 2.5M	42119-0005-00	1	(include)	2		
3-yrs 24x7x4hrs Onsite Support Service		1	5,550,000	1		5,550,000
Sub Total					**64,740,000**	**15,450,000**
Client/Server Software						
CentOS Linux OS Platform 3yr 24x7x4hrs		3	7,200,000	4		28,800,000
JBoss Web Server 3yr 24x7x4hrs		3	12,600,000	3		37,800,000
Goldilocks v3.1 Standard Edition 16Core		2	96,000,000	1	96,000,000	
Goldilocks v3.1 Standard Edition 16Core Technical Supports – 3-yrs 24x7x4hrs		2	10,000,000	3		30,000,000
Sub Total					**96,000,000**	**96,600,000**
Other Hardware						
HP 1420-24G-2SFP (1/100/1000, 24-port))(w/spares)	JH017A	4	242,000	3	726,000	
Sub Total					**726,000**	
Discounts*						
CentOS Discount		3				-5,760,000
JBoss Discount		3				-7,560,000
SUNJESOFT Discount		2			-64,000,000	-15,600,000
Sub Total					**-64,000,000**	**-28,920,000**
Total					**150,466,000**	**91,470,000**

Pricing Notes

1) Taejin Infotech Co., LTD 3) http://www.linux.co.kr

2) SUNJESOFT Inc. 4) http://mjnetwork.co.kr

All of the prices are based on South Korea's currency, KRW (₩, Korean Won) and excluded VAT.

* All discounts are based on Korea list prices and for similar quantities and configurations. Discounts for similarly sized configurations will be similar to those quoted here, but may vary based on the components in the configuration.

Three year cost of ownership KRW(₩): 241,936,000

Benchmark rating: 139,909 tpmC

Price/Performance: 1,730 ₩ / tpmC

Benchmark implementation and results independantly audited by Doug Johnson of InfoSizing (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 pricing specifications. If you find that the stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org. Thank you.

Goldilocks v3.1 Standard Edition on Jet-speed™ HHA2212

TPC-C Version 5.11.0 TPC Pricing 2.1.0	
Report Date 09-May-2017	**Revised Date** 31-May-2017
Availability Date: 09-May-2017	

MQTh, computed Maximum Qualified Throughput	**139,909 tpmC**

Response Times (seconds)	**Min**	**Average**	**90th**	**Max**
New-Order	0.102	0.105	0.106	0.954
Payment	0.102	0.105	0.105	0.952
Order-Status	0.102	0.104	0.104	0.950
Delivery (interactive portion)	0.101	0.101	0.101	0.948
Delivery (deferred portion)	0.003	0.016	0.014	4.625
Stock-Level	0.102	0.104	0.105	0.950
Menu	0.101	0.102	0.101	0.948

Emulated Display Delay: 0.1 sec.

Transaction Mix	**Percent**	**Number**
New-Order	44.982%	67,158,862
Payment	43.010%	64,214,272
Order-Status	4.002%	5,975,686
Delivery	4.003%	5,976,495
Stock-Level	4.003%	5,976,176

Keying Times (seconds)	**Min**	**Average**	**Max**
New-Order	18.000	18.000	18.847
Payment	3.000	3.000	3.847
Order-Status	2.000	2.000	2.847
Delivery	2.000	2.000	2.847
Stock-Level	2.000	2.000	2.847

Think Times (seconds)	**Min**	**Average**	**Max**
New-Order	0.000	12.002	120.947
Payment	0.000	12.006	120.947
Order-Status	0.000	10.008	100.100
Delivery	0.000	5.005	50.947
Stock-Level	0.000	5.011	50.100

Test Duration	
Ramp-up time	30 min
Measurement Interval (MI)	480 min
Checkpoints in MI	17
Checkpoint interval	Avg 27.5 min
Number of transactions (all types) in MI	149,301,491

General Items

4.4 Application Code and Definition Statements

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

Appendix A contains the application source code for the transactions.

0.2 Benchmark Sponsor

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

This benchmark was sponsored by TTA, Telecommunications Technology Association. The implementation was developed and engineered in partnership with SUNJESOFT Inc. and Taejin InfoTech Co., LTD.

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 by not limited to:

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

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

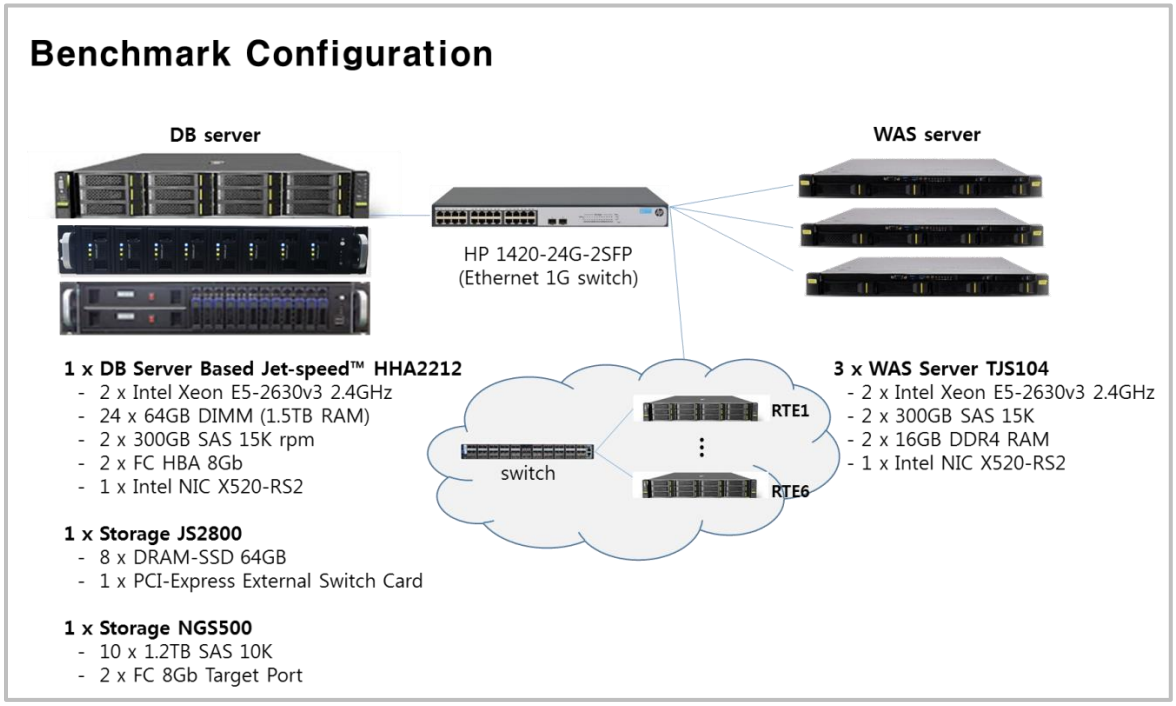
Appendix C contains the tunable parameters for the database, the operating system, and the transaction monitor.

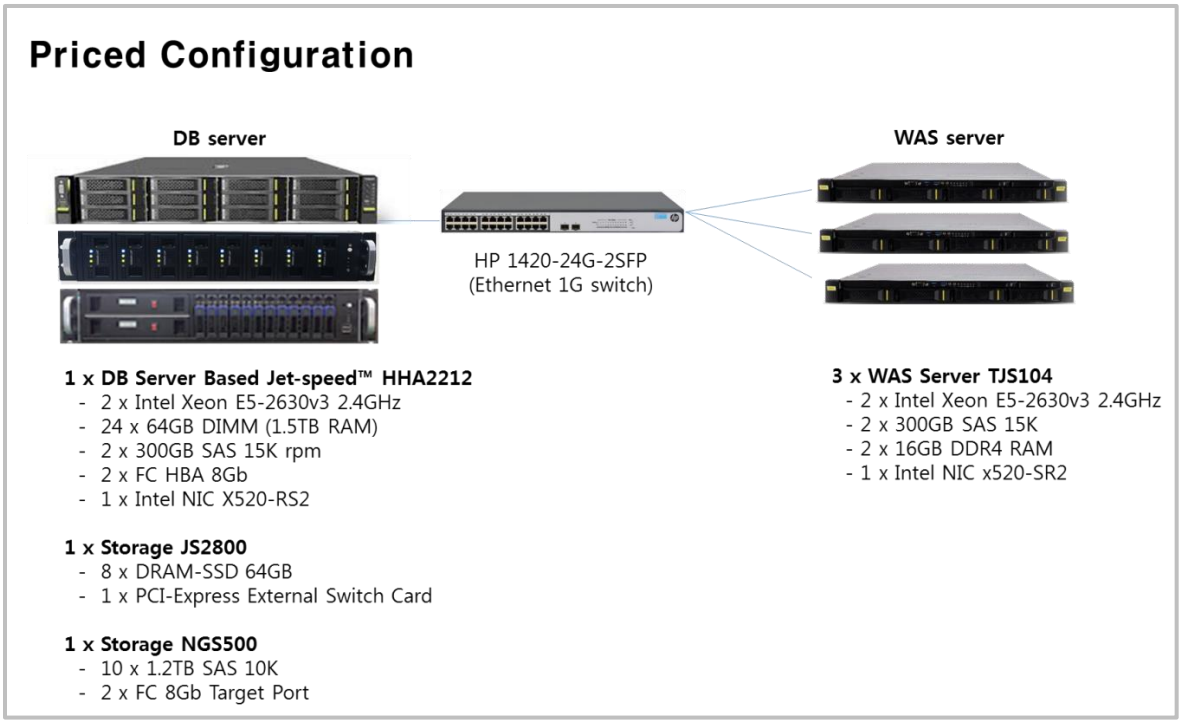
0.4 Configuration Diagrams

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

The configuration diagram for both the tested and priced system is depicted in Figure 0.1.

Figure 0.1: Benchmarked and Priced Configuration





Clause 1: Logical Database Design

4.4 Table Definitions

Listing must be provided for all table definition statements and all other statements used to set up the database.

Appendix B contains the code used to define and load the database tables.

1.2 Physical Organization of Database

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

The physical organization of the database is shown in Table 1.2.

Table 1.2: Physical Organization of the Database

Controller	Array	RAID Array	Drives	Content
LSI MegaRAID SAS 9361-8i	Internal	RAID 1	2 x SAS 300GB 10K	OS
LSI MegaRAID SAS 9361-8i	NG500 Array	RAID 1	2 x SAS 1.2GB 10K	Database files
			2 x SAS 1.2GB 10K	Database files
			2 x SAS 1.2GB 10K	Database files
			2 x SAS 1.2GB 10K	Database files
			2 x SAS 1.2GB 10K	Database files
PCIe Host Interface Card	JS2800 Array	RAID 10	8 x Jetspeed 64GB	Redo Logs

1.3 Insert and Delete Operations

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

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

1.4 Horizontal or Vertical Partitioning

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

No horizontal or vertical partitioning was used.

1.5 Replication or Duplication

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

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

Clause 2: Transaction and Terminal Profiles

2.1 Random Number Generation

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

Random numbers were generated using ‘SysVr4 rand_r()’ call. The seed value for ‘rand_r()’ was collected and sent to the auditor for review.

2.2 Input/Output Screens

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

All screen layouts followed the requirements of the specifications.

2.3 Priced Terminal Feature

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

The terminal attributes were manually verified by the auditor by validating that each required feature was implemented.

2.4 Presentation Managers

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

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

2.5 Transaction Statistics

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

Table 2.1: Transaction Statistics

Statistic		Value
New Order	Home warehouse order lines Remote warehouse order lines Rolled back transactions Average items per order	99.001% 0.999% 1.002% 10.000
Payment	Home warehouse Remote warehouse Accessed by last name	84.994% 15.006% 59.998%
Order Status	Accessed by last name	60.025%
Delivery	Skipped transactions	0
Transaction Mix	New Order Payment Order status Delivery Stock level	44.982% 43.010% 4.002% 4.003% 4.003%

2.6 Queuing Mechanism

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

The queuing mechanism was implemented using ‘BlockingQueue’ provided by Java.

Clause 3: Transaction and System Properties

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

All ACID property tests conducted according to the specification.

3.1 Atomicity

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

3.1.1 Atomicity of Completed Transactions

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.

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

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.

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

3.2 Consistency

Consistency is the property of the application that requires any execution of a data base transaction to take the database 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 Consistency conditions 1 through 4 were demonstrated as follows:

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:

d_next_o_id – 1 = max(o_id) = max(no_o_id)
where (d_w_id = o_w_id = no_w_id) and (d_id = o_d_id = no_d_id)

3. For each District within a Warehouse, the value of the most recent Order ID [max(no_o_id)] minus the first.
4. 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:
   max(no_o_id) – min(no_o_id) + 1 = number of rows in New Order for the Warehouse/District
5. 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:
   sum(o_ol_cnt) = number of rows in the Order Line table for the Warehouse/District

To test consistency, a short (5 to 10 minutes) RTE run was executed against a freshly loaded and consistent database. After the run, consistency conditions 1 through 4 were tested using a script issuing queries against the database. All queries showed that the database was in a consistent state according to the four specified conditions.

3.3 Isolation

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

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. Each included timestamps to demonstrate the concurrency of operations. The results of the queries were logged. The captured logs were verified to demonstrate the required isolation had been met.

Isolation Test 1

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

The test proceeds as follows:

1. An Order-Status transaction T0 was executed and committed for a randomly selected customer, and the order returned was noted.
2. A New-Order transaction T1 was started for the same customer used in T0. T1 was stopped prior to COMMIT.
3. An Order-Status transaction T2 was started for the same customer used in T1. T2 completed and was committed without being blocked by T1. T2 returned the same order that T0 had returned.
4. T1 was allowed to complete and was committed.
5. An Order-Status transaction T3 was started for the same customer used in T1. T3 returned the order inserted by T1.

Isolation Test 2

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

The test proceeds as follows:

1. An Order-Status transaction T0 was executed and committed for a randomly selected customer and the order returned was noted.
2. A New-Order transaction T1 with an invalid item number was started for the same customer used in T0. T1 was stopped immediately prior to ROLLBACK.
3. An Order-Status transaction T2 was started for the same customer used in T1. T2 completed and was committed without being blocked by T1. T2 returned the same order that T0 had returned.
4. T1 was allowed to ROLLBACK.
5. An Order-Status transaction T3 was started for the same customer used in T1. T3 returned the same order that T0 had returned.

Isolation Test 3

This test demonstrates isolation for write-write conflicts of two New-Order transactions when both transactions are committed.

The test proceeds as follows:

1. The D_NEXT_O_ID of a randomly selected district was retrieved.
2. A New-Order transaction T1 was started for a randomly selected customer within the district used in step 1. T1 was stopped immediately prior to COMMIT.
3. Another New-Order transaction T2 was started for the same customer used in T1. T2 waited.
4. T1 was allowed to complete. T2 completed and was committed.
5. The order number returned by T1 was the same as the D_NEXT_O_ID retrieved in step 1. The order number returned by T2 was one greater than the order number returned by T1.
6. The D_NEXT_O_ID of the same district was retrieved again. It had been incremented by two (i.e. it was one greater than the order number returned by T2).

Isolation Test 4

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

The test proceeds as follows:

1. The D_NEXT_O_ID of a randomly selected district was retrieved.
2. A New-Order transaction T1, with an invalid item number, was started for a randomly selected customer within the district used in step 1. T1 was stopped immediately prior to ROLLBACK.
3. Another New-Order transaction T2 was started for the same customer used in T1. T2 waited.
4. T1 was allowed to roll back, and T2 completed and was committed.
5. The order number returned by T2 was the same as the D_NEXT_O_ID retrieved in step 1.
6. The D_NEXT_O_ID of the same district was retrieved again. It had been incremented by one (i.e. one greater than the order number returned by T2).

Isolation Test 5

This test demonstrates isolation for write-write conflicts of Payment and Delivery transactions when Delivery transaction is committed.

The test proceeds as follows:

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

Isolation Test 6

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

The test proceeds as follows:

1. A query was executed to find out the customer who is to be updated by the next delivery transaction for a randomly selected warehouse and district.
2. The C_BALANCE of the customer found in step 1 was retrieved.
3. A Delivery transaction T1 was started for the same warehouse used in step 1. T1 was stopped immediately prior to ROLLBACK.
4. A Payment transaction T2 was started for the same customer found in step 1. T2 waited.
5. T1 was allowed to ROLLBACK. T2 completed and was committed. The C_BALANCE of the customer found in step 1 was retrieved again. The C_BALANCE reflected the results of only T2.

Isolation Test 7

This test demonstrates repeatable reads for the New-Order transaction while an interactive transaction updates the prices of some items.

The test proceeds as follows:

1. The I_PRICE of two randomly selected items X and Y were retrieved.
2. A New-Order transaction T1 with a group of items including items X and Y was started. T1 was stopped immediately after retrieving the prices of all items. The prices of items X and Y retrieved matched those retrieved in step 1.
3. A transaction T2 was started to increase the price of items X and Y by 10%.
4. T2 did not stall and was committed.
5. T1 was resumed, and the prices of all items were retrieved again within T1. The prices of items X and Y matched those retrieved in step 1.

6. T1 was committed.
7. The prices of items X and Y were retrieved again. The values matched the values set by T2.

The Execution followed Case D, where T3 does not stall and no transaction is rolled back. Query T4 verifies the price change made by T3.

Isolation Test 8

This test demonstrates isolation for phantom protection between New-Order and Order-Status transactions.

The test proceeds as follows:

1. An Order-Status transaction T1 was started for a randomly selected customer.
2. T1 was stopped immediately after reading the order table for the selected customer to find the most recent order for that customer.
3. A New-Order transaction T2 was started for the same customer. T2 completed and was committed without being blocked by T1.
4. T1 was resumed and the ORDER table was read again to determine the most recent order for the same customer. The order found was the same as the one found in step 2.
5. T1 completed and was committed.

Isolation Test 9.

This test demonstrates isolation for phantom protection between New-Order and Delivery transactions.

The test proceeds as follows:

1. The NO_D_ID of all NEW_ORDER rows for a randomly selected warehouse and district was changed to 11. The changes were committed.
2. A Delivery transaction T1 was started for the selected warehouse.
3. T1 was stopped immediately after reading the NEW_ORDER table for the selected warehouse and district. No qualifying row was found.
4. A New-Order transaction T2 was started for the same warehouse and district. T2 completed and was committed without being blocked by T1.
5. T1 was resumed and the NEW_ORDER table was read again. No qualifying row was found.
6. T1 completed and was committed.
7. The NO_D_ID of all NEW_ORDER rows for the selected warehouse and district was restored to the original value. The changes were committed.

3.4 Durability

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)*

3.4.1 Durable Media Failure

3.4.1.1 Loss of Log Media and Data Media

These tests were conducted on a fully scaled database. To demonstrate recovery from a permanent failure of durable medium containing TPC-C Log Media and Data Media, the following steps were executed:

1. The total number of orders is determined by the sum of D_NEXT_O_ID of all rows in the DISTRICT table; giving the beginning count.
2. The consistency is verified.
3. The RTE is started with full user load.
4. The test is allowed to run for a minimum of 1 minutes after ramp-up.
5. A first checkpoint is initiated and completed.
6. The test is allowed to run for a minimum of 1 more minute.
7. A second checkpoint is initiated.
8. Before the second checkpoint completes, one data disk is disabled by removing it physically. Since the data disks are configured with redundancy, the transactions continued to run without interruption.
9. The test is allowed to run for a minimum of 5 more minutes.
10. Before the second checkpoint completes, one log device is disabled by removing it physically. Since the log devices are configured with redundancy, the transactions continued to run without interruption.
11. The test is allowed to run until the second checkpoint has completed, but no less than 5 more minutes.
12. The RTE run is completed.
13. The consistency is verified.
14. Step 1 is repeated, giving the ending count.
15. The RTE result file is used to determine the number of New-Order transactions successfully completed during the full run.
16. The difference between the counts in step 1 and 14 is compared with the count from step 15. The result should verify that all committed transactions have been successfully recovered.
17. Data from the success file is used to query the database to demonstrate that the last 500 successful New-Orders have corresponding rows in the ORDER table.

3.4.2 Instantaneous Interruption, Loss of Memory

As the loss of power erases the contents of memory, the instantaneous interruption and the loss of memory tests were combined into a single test. This test was executed on a fully scaled database. The following steps were executed:

1. The total number of orders is determined by the sum of D_NEXT_O_ID of all rows in the DISTRICT table; giving the beginning count.
2. The consistency is verified.
3. The RTE is started with full user load.
4. The test is allowed to run for a minimum of 1 minutes at full load (after ramp-up).
5. A first checkpoint is initiated and completed.
6. The test is allowed to run for a minimum of 1 more minute.
7. A second checkpoint is initiated.
8. Before the second checkpoint completes, the primary power to the back-end server is shut off (removing power cord).
9. The RTE is shutdown.
10. Power is restored to the back-end server and the system performs an automatic recovery.
11. GOLDILOCKS is restarted and performs an automatic recovery.
12. Step 1 is repeated, giving the ending count.
13. The consistency is verified.
14. RTE report is used to determine the number of New-Order transactions successfully completed during the full run.
15. The difference between the counts in step 1 and 12 is compared with the count from step 14. The result should verify that all committed transactions have been successfully recovered.
16. Data from the success file is used to query the database to demonstrate that the last 500 successful New-Orders have corresponding rows in the ORDER table.

Clause 4: Scaling and Database Population

4. 1 Cardinality of Tables

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

Table 4.1 shows that number of rows for each table as they were initially built.

Table 4.1: Number of Rows for Server

Table	Cardinality
Warehouse	11,000
District	110,000
Customer	330,000,000
History	330,000,000
Order	330,000,000
New Order	99,000,000
Order Line	3,299,229,458
Stock	1,100,000,000
Item	100,000
Unused Warehouses	0

4.2 Database Implementation

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

Goldilocks v3.1 is a main memory DBMS, implemented relational model.

Procedure, placed in PSM (Persistent Stored Module), is used for oprating transactions. This procedure is called by JDBC's 'Call Statement'. All application code and procedures are listed in Appendix A.

4.3 Distribution of Database Files

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

The database files are stored on a set of ten 1.2TB disks configured as five RAID1 pairs. The database log files are stored on a single RAID10 set of eight 64GB DRAM-SSD

Table 4.3: Database file locations

Name	Location	Description
system_XXX.dbf	/data/db/db1	system_data.dbf, system_dict.dbf, system_undo.dbf
tpcc_data_XX.dbf	/data/db/db1 /data/db/db2 /data/db/db3 /data/db/db4 /data/db/db5	tpcc_data01.dbf ~tpcc_data69.dbf
redo_X_X.log	/jsm/wal	redo_0_0.log, redo_1_0.log, redo_2_0.log, redo_3_0.log, redo_4_0.log

The distribution of tables and logs across storage media is shown in Table 1.2.

4.4 60 Day Space

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

Table 4.4: 60 Day Space Calculations

Base Unit (Kbytes)	1
tpmC	139,909.976

Table	Rows	Data	Index	Initial Population	5% Growth	8-Hour Growth	Required Runtime Space
WAREHOUSE	11,000	88,680	256	88,936	4,447	0	93,383
DISTRICT	110,000	13,960	2,912	16,872	844	0	17,716
CUSTOMER	330,000,000	212,024,072	22,846,608	234,870,680	11,743,534	0	246,614,214
NEW_ORDER	99,000,000	6,245,344	2,953,016	9,198,360	459,918	0	9,658,278
ITEM	100,000	10,808	2,368	13,176	659	0	13,835
STOCK	1,100,000,000	404,482,040	30,328,320	434,810,360	21,740,518	0	456,550,878
HISTORY	330,000,000	27,046,016	0	27,046,016	0	5,504,011	32,550,027
ORDERS	330,000,000	20,886,104	21,037,000	41,923,104	0	4,250,435	46,173,539
ORDER_LINE	3,299,229,458	309,993,736	108,528,664	418,522,400	0	63,085,405	481,607,805
Total		**980,790,760**	**185,699,144**	**1,166,489,904**	**33,949,919**	**72,839,852**	**1,273,279,675**

Storage Requirements		**Memory Requirements**	
Dynamic-Space	357,925,856	Final Allocation	1,310,971,128
Free-Space	1,471,936	Non-Growing 5%	33,949,919
Static-Space	808,564,048		
Daily-Growth	72,839,852		
Daily-Spread	0		
60-Day Space	**5,178,955,139**	**1-Day Memory**	**1,344,921,047**

Clause 5: Performance Metrics

5.1 TPC Benchmark C Metrics

The TPC-C Metrics are reported in the front of this report as part of the executive summary.

5.2 Response Times

Ninetieth percentile, maximum and average response times must be reported for all transaction types as well as for the menu response time.

During the performance run transactions are submitted by the RTE in accordance to the required mix, keying times and think times of the benchmark Specification. Transactions are submitted by emulated users via HTTP. After each transaction the emulated user waits for a randomly generated think time before selecting the next transaction. All timings are recorded by the RTE. The response time is measured from the submission of the transaction until the last byte of response is received by the RTE.

Table 5.2 shows the average, maximum and 90th percentile response time for each transaction types and for the menu.

Table 5.2: Transaction Response Times

Type	Average	Maximum	90th %
New-Order	0.105	0.954	0.106
Payment	0.105	0.952	0.105
Order-Status	0.104	0.950	0.104
Interactive Delivery	0.101	0.948	0.101
Deferred Delivery	0.016	4.625	0.014
Stock-Level	0.104	0.950	0.105
Menu	0.102	0.948	0.101

5.3 Keying and Think Times

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

Table 5.1 shows the minimum, average and maximum keying time and think time for each transaction types.

Table 5.3: Keying Times/Think Times

Type	Minimum	Average	Maximum
New-Order	18.000/0.000	18.000/12.002	18.847/120.947
Payment	3.000/0.000	3.000/12.006	3.847/120.947
Order-Status	2.000/0.000	2.000/10.008	2.847/100.100
Delivery (interactive)	2.000/0.000	2.000/5.005	2.847/50.947
Stock-Level	2.000/0.000	2.000/5.011	2.847/50.100

5.4 Distribution and Performance Curves

5.4.1 Response Time frequency distribution curves

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

Figure 5.4.1.1 shows the Response Time frequency distribution curves for the New-Order transaction.

Figure 5.4.1.2 shows the Response Time frequency distribution curves for the Payment transaction.

Figure 5.4.1.3 shows the Response Time frequency distribution curves for the Order-Status transaction.

Figure 5.4.1.4 shows the Response Time frequency distribution curves for the interactive portion of the Delivery transaction.

Figure 5.4.1.5 shows the Response Time frequency distribution curves for the Stock-Level transaction.

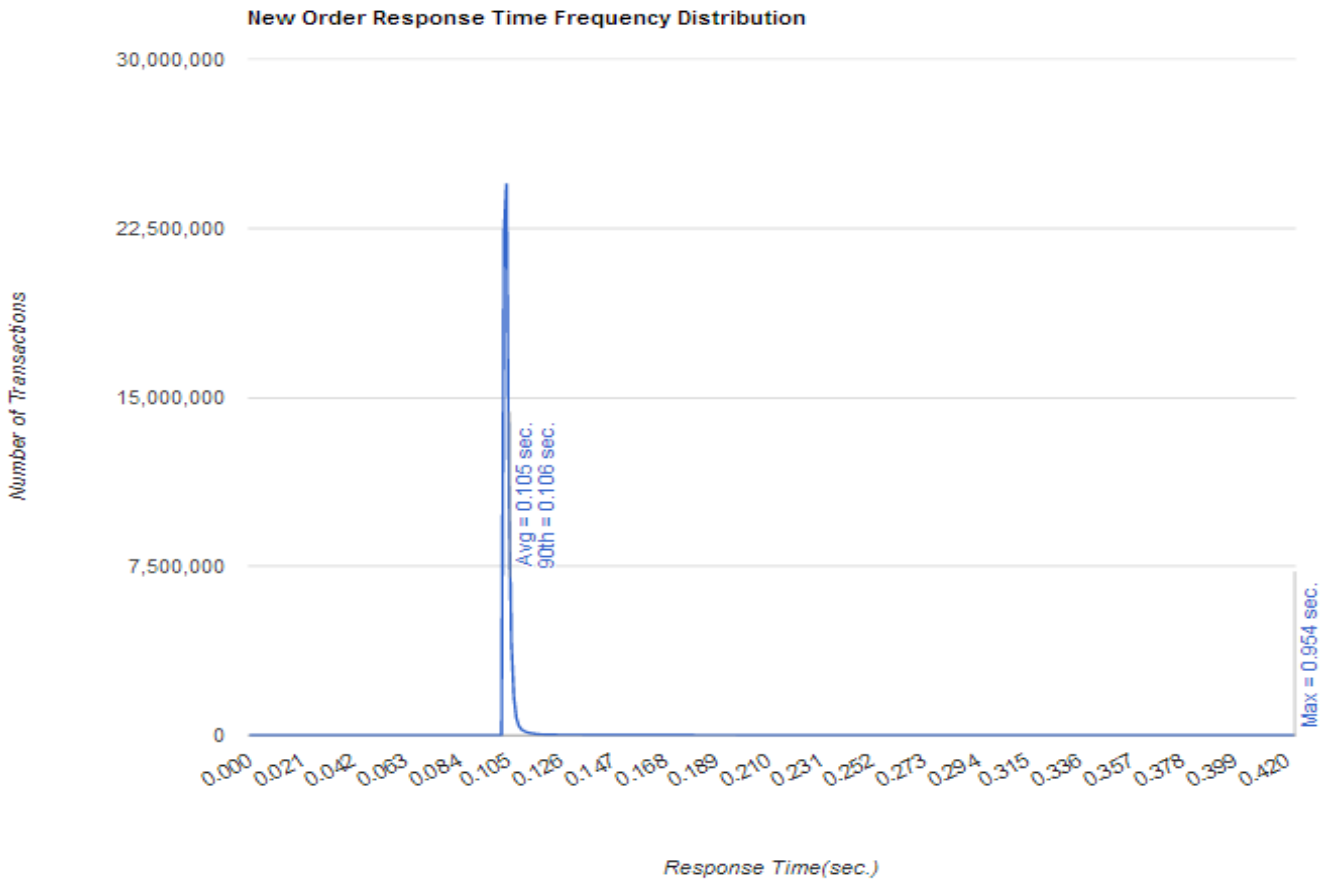


Figure 5.4.1.1: New-Order RT Frequency Distribution

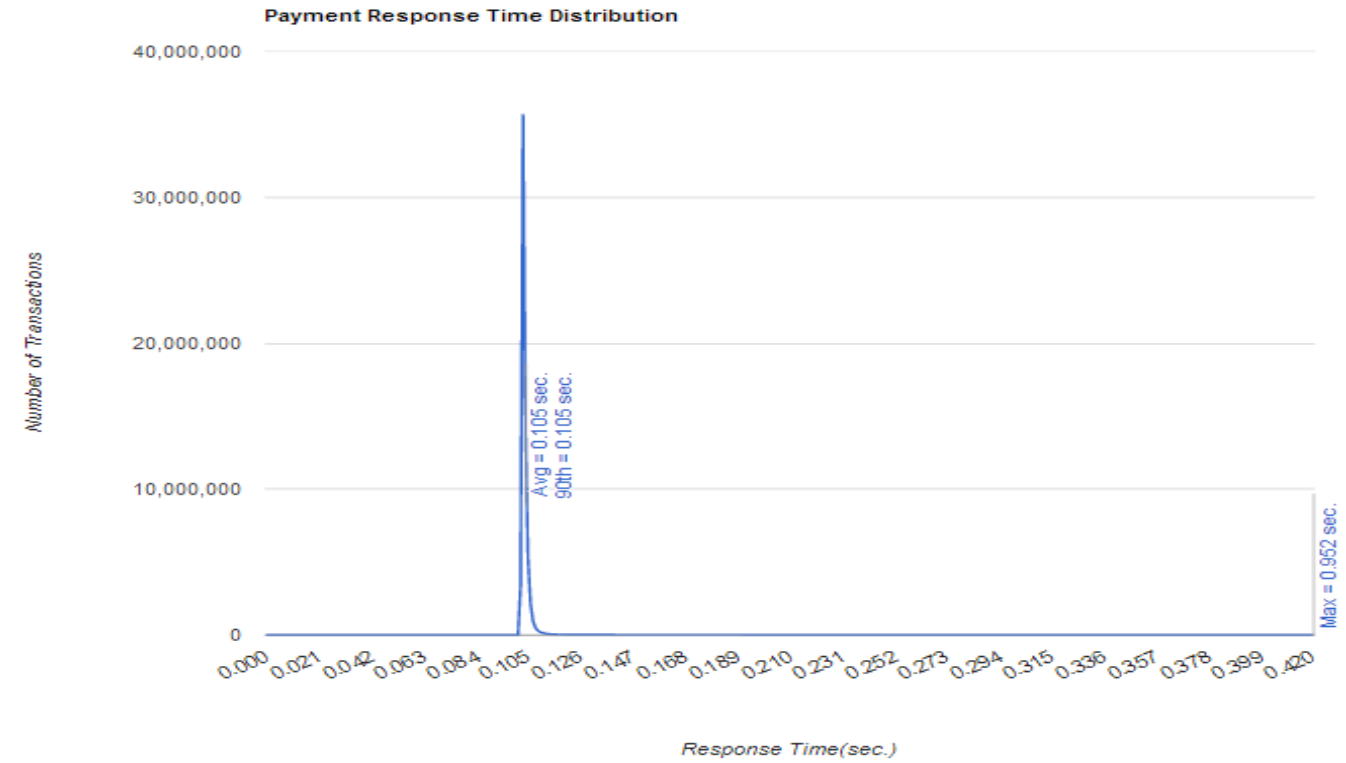


Figure 5.4.1.2: Payment RT Frequency Distribution

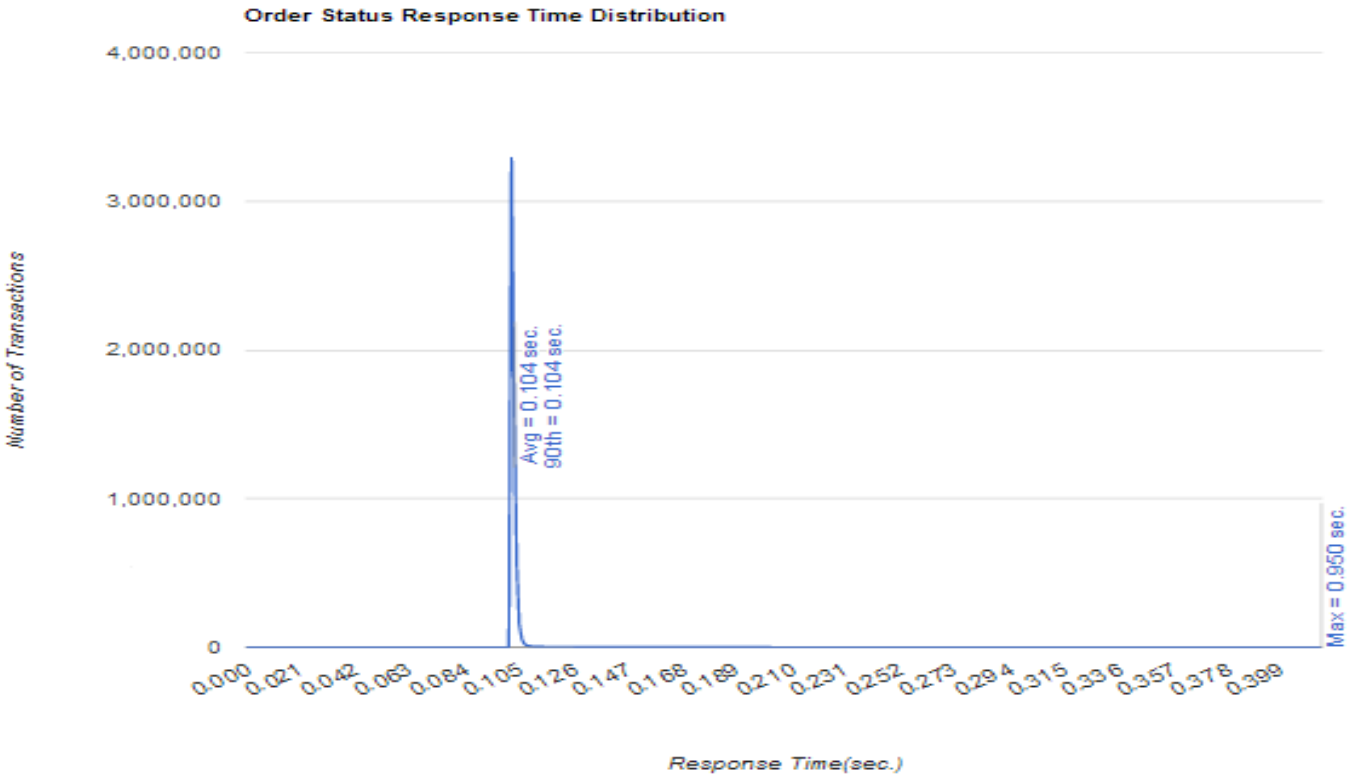


Figure 5.4.1.3: Order-Status RT Frequency Distribution

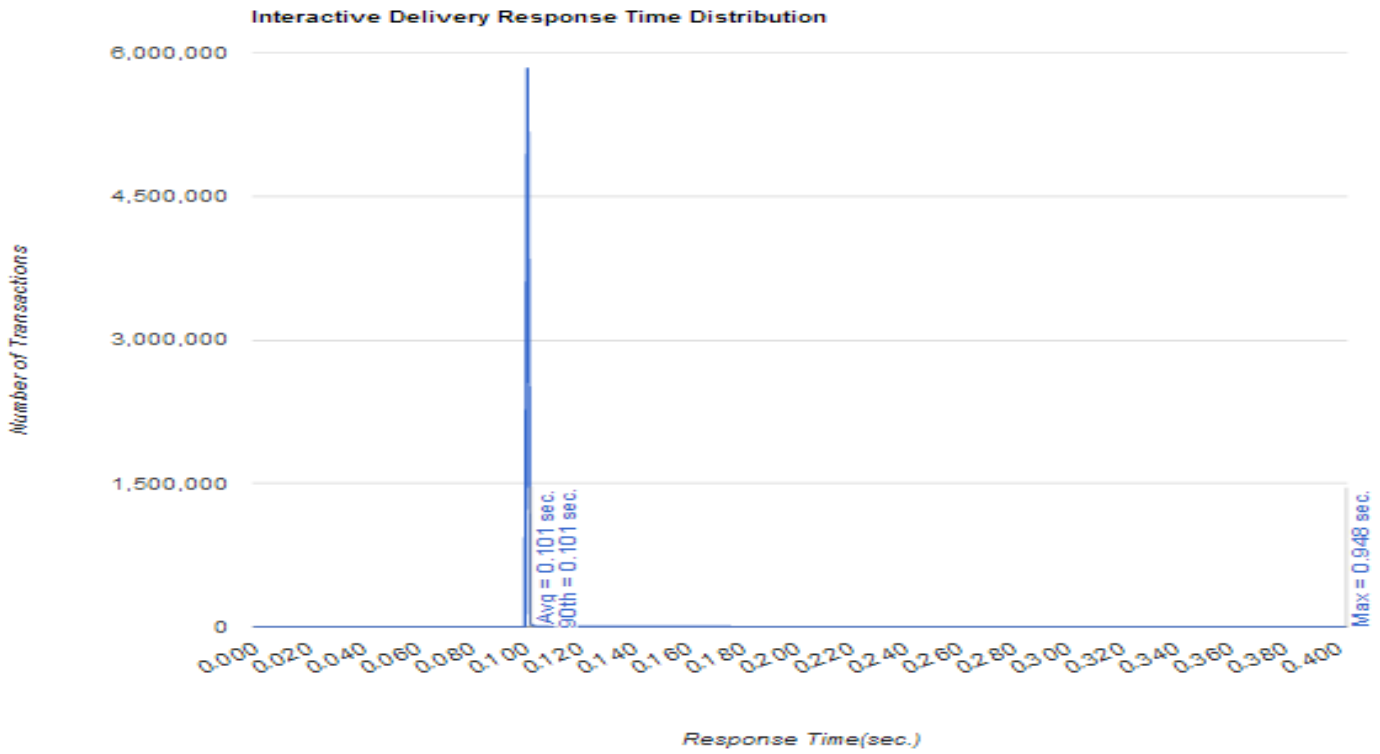


Figure 5.4.1.4: Delivery (Interactive) RT Frequency Distribution

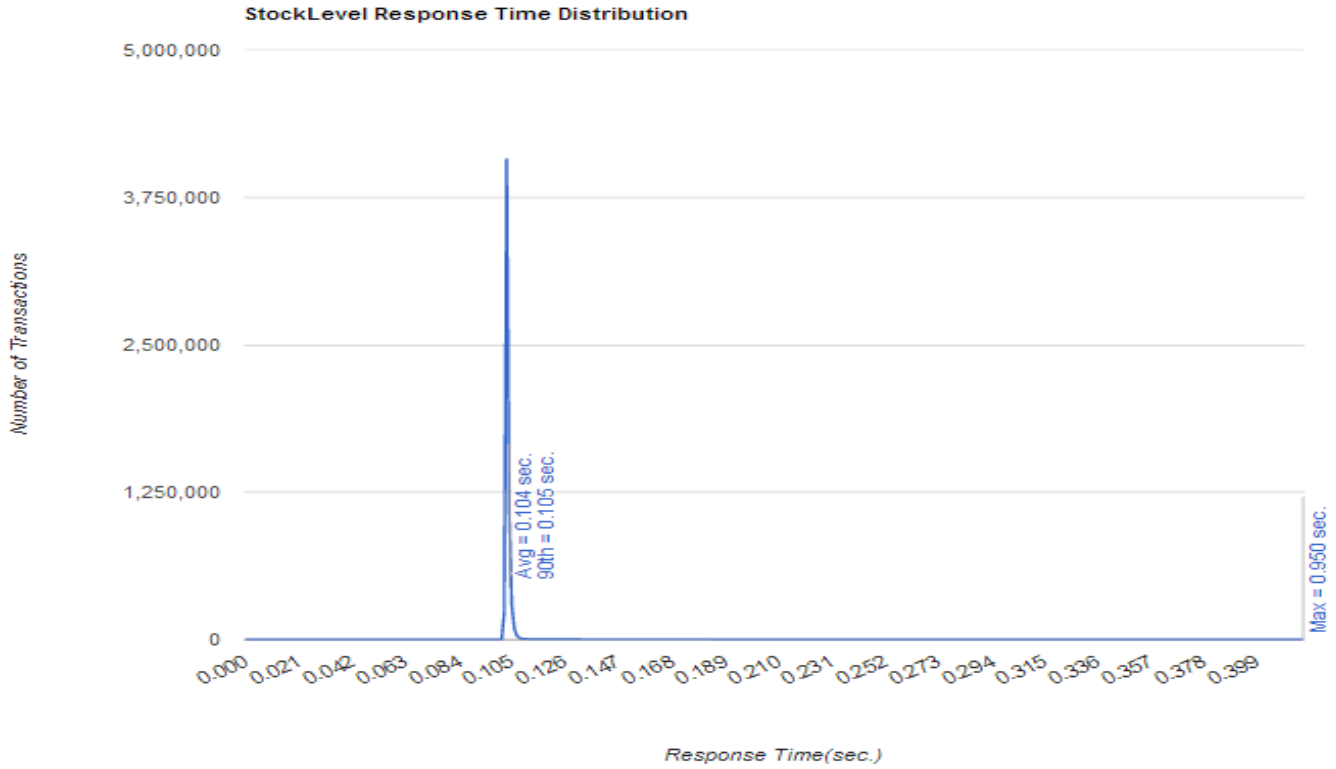


Figure 5.4.1.5: Stock-Level RT Frequency Distribution

5.4.2 Response Time versus throughput

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

Figure 5.4.2.1 shows the Response Time versus throughput curves for the New-Order transaction.

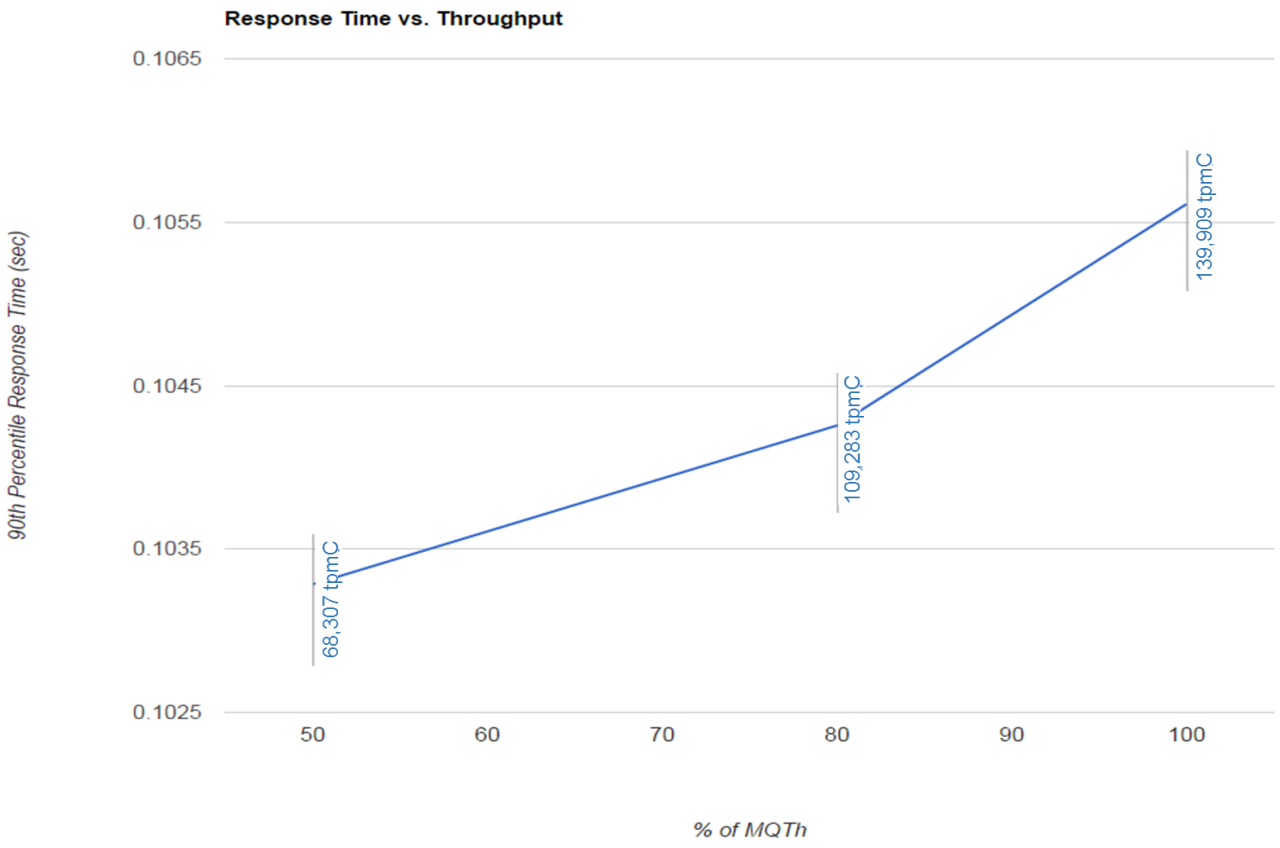


Figure 5.4.2.1: New-Order RT versus Throughput

5.4.3 Think Time frequency distribution

Think Time frequency distribution curves (see Clause 5.6.3) must be reported for each transaction type.

Figure 5.4.3.1 shows the Think Time frequency distribution curves for the New-Order transaction.

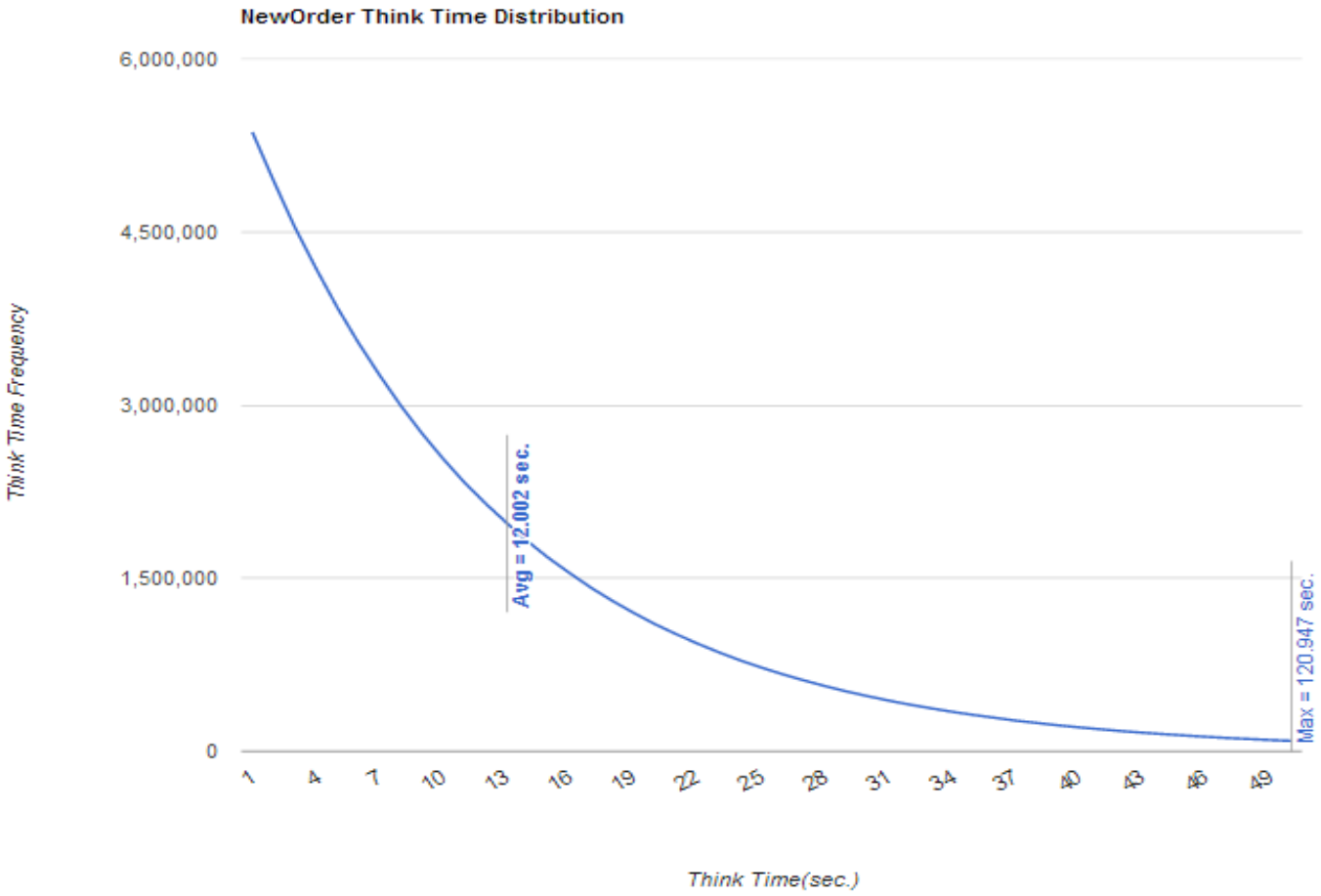


Figure 5.4.3.1: New-Order Think Time Frequency Distribution

5.4.4 Throughput versus elapsed time

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

Figure 5.4.4.1 shows the throughput versus elapsed time for the New-Order transaction. The start and end of the Measurement Interval is included on the figure.

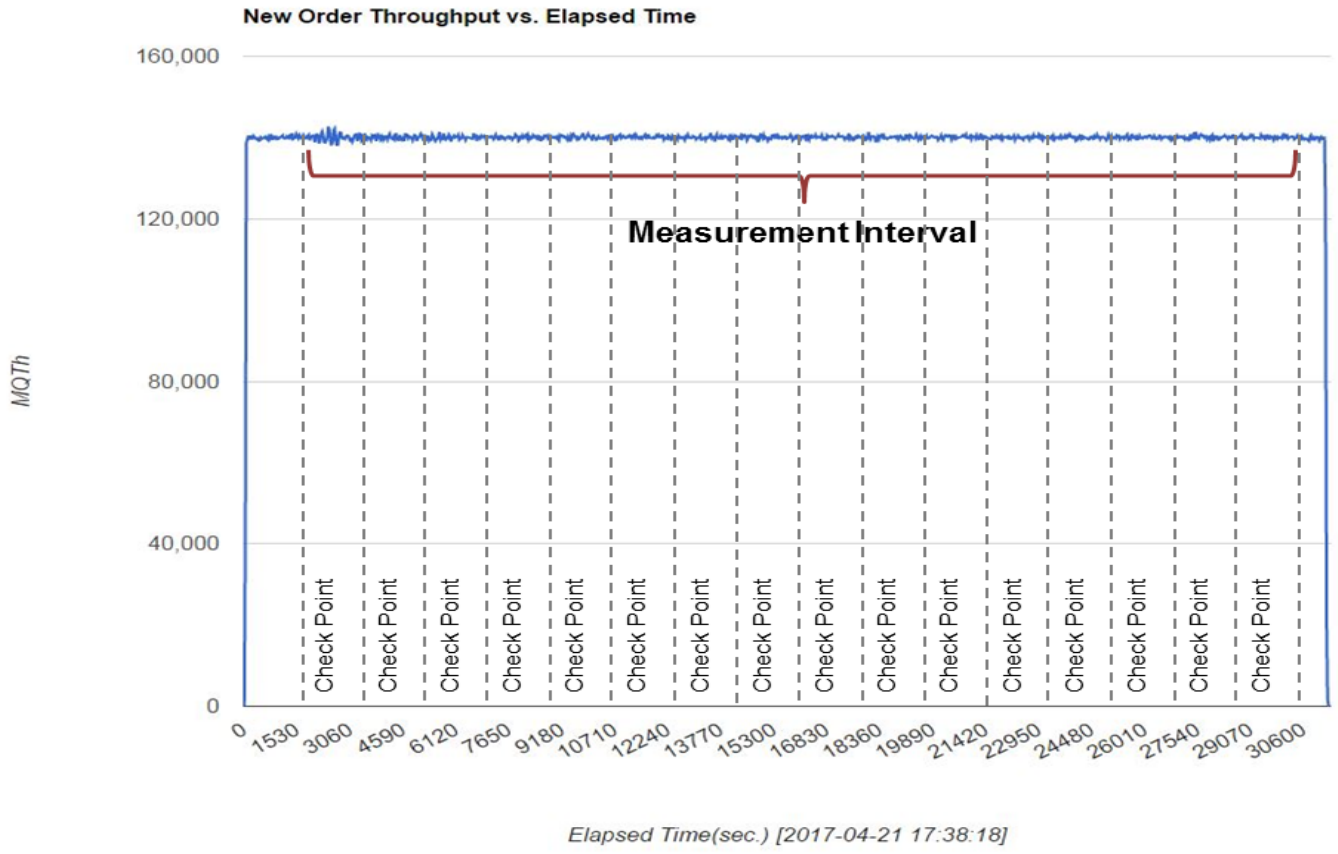


Figure 5.4.4.1: New-Order Throughput versus Elapsed Time

5.5 Steady State Determination

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

Steady state was determined using real time monitor utilities from the RTE. Steady state was further confirmed by a visual analysis of the throughput graph.

5.6 Work Performed During Steady State

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

During the Test Run, emulated users submit TPC-C transactions according to the described mix, keying time and think times. The transactions are implemented in accordance with the requirements of the specification. An emulated user submits transaction input via HTTP and receives acknoledment of the completed transaction. The response time is measured from the start of the transaction until the last byte is received by the RTE.

There are two wait times in each RTE cycle. First, after the menu selection, the RTE waits to simulate keying-in the input data for the transaction. It is called the ‘Keying Time’. Second, after the transaction returns, the RTE waits to simulate reading/studying the transaction output. It is called the ‘Think Time’.

During the test, Goldilocks satisfied all of the ACID properties required by the benchmark specification. Committed transactions write a Redo record in the transaction log, to be used in case of system failure. The Redo records are used for roll-forward recovery during a re-start following a failure. This prevents the system from loosing any commited transactions. Checkpoints periodically occurred about every 27.5 min. and are completed in about 24 min.

5.7 Measurement Period Duration

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

The reported measured interval was 28,800 seconds (480 minutes).

5.8 Transaction Statistics

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

Table 5.4 shows the transaction mix for the reported measurement interval.

Table 5.4: Transaction Mix

Transaction	Mix
New Order	44.982%
Payment	43.010%
Order status	4.002%
Delivery	4.003%
Stock level	4.003%

Table 5.5 shows the various statistics for the transactions.

Table 5.5: Transaction Statistics

Statistic		Value
New Order	Home warehouse order lines	99.001%
	Remote warehouse order lines	0.999%
	Rolled back transactions	1.002%
	Average items per order	10.000
Payment	Home warehouse	84.994%
	Remote warehouse	15.006%
	Accessed by last name	59.998%
Order Status	Accessed by last name	60.025%
Delivery	Skipped transactions	0

5.9 Checkpoints

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

17 complete checkpoints occurred during the Measurement Interval. The checkpoint start time and duration are listed in table 5.6.

Table 5.6: Checkpoints

Event	Start Time	End Time
Start Measurement Interval	21-Apr-2017 18:08:18	N/A
Checkpoint1	21-Apr-2017 18:10:53	21-Apr-2017 18:35:45
Checkpoint2	21-Apr-2017 18:38:18	21-Apr-2017 19:03:08
Checkpoint3	21-Apr-2017 19:05:43	21-Apr-2017 19:30:37
Checkpoint4	21-Apr-2017 19:33:07	21-Apr-2017 19:57:57
Checkpoint5	21-Apr-2017 20:00:34	21-Apr-2017 20:25:27
Checkpoint6	21-Apr-2017 20:27:57	21-Apr-2017 20:52:47
Checkpoint7	21-Apr-2017 20:55:22	21-Apr-2017 21:20:12
Checkpoint8	21-Apr-2017 21:22:48	21-Apr-2017 21:47:40
Checkpoint9	21-Apr-2017 21:50:15	21-Apr-2017 22:15:07
Checkpoint10	21-Apr-2017 22:17:41	21-Apr-2017 22:42:45
Checkpoint11	21-Apr-2017 22:45:07	21-Apr-2017 23:10:10
Checkpoint12	21-Apr-2017 23:12:32	21-Apr-2017 23:37:38
Checkpoint13	21-Apr-2017 23:40:00	22-Apr-2017 00:05:02
Checkpoint14	22-Apr-2017 00:07:26	22-Apr-2017 00:32:44
Checkpoint15	22-Apr-2017 00:34:53	22-Apr-2017 01:00:10
Checkpoint16	22-Apr-2017 01:02:19	22-Apr-2017 01:27:34
Checkpoint17	22-Apr-2017 01:29:45	22-Apr-2017 01:54:54
End Measurement Interval	N/A	22-Apr-2017 02:08:19

Clause 6: SUT, Driver and Communication

6.1 Remote Terminal Emulator (RTE)

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

The RTE software used was internally developed. The RTE simulated web users. It generated random input data based on the benchmark requirements and recorded response times and other statistics for each transaction cycle.

6.2 Emulated Components

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

No components were emulated by the driver system.

6.3 Functional Diagrams

A complete functional diagram of both the benchmark configuration and the configuration of the proposed (target) system must be disclosed. A detailed list of all hardware and software functionality being performed on the Driver System and its interface to the SUT must be disclosed.

The diagram in Figure 0.1 shows the tested and priced benchmark configurations.

6.4 Networks

The network configuration of both the tested services and proposed (target) services which are being represented and a thorough explanation of exactly which parts of the proposed configuration are being replaced with the Driver System must be disclosed.

The bandwidth of the networks used in the tested/priced configuration must be disclosed.

The diagram in Figure 0.1 shows the network configuration between the components of the tested configuration. The RTE and the SUT are connected through a 1Gbit switch.

The network bandwidths are listed in Figure 0.1.

6.5 Operator Intervention

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

No operator intervention is required to sustain eight hours at the reported throughput.

Clause 7: Pricing

7. 1 Hardware and Software Pricing

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

The details of the hardware and software are reported in the front of this report as part of the Executive Summary.

7.2 Three Year Price

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 TPC-C result are reported in the front of this report as part of the Executive Summary.

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 priced system are available on 09-May-2017.

Clause 8: Reporting

8.1 Full Disclosure Report

A Full Disclosure report is required in order for results to be considered compliant with the TPC-C benchmark specification

This document constitute the Full Disclosure Report for the TPC-C benchmark result describes within.

Clause 9: Auditor Attestation

9.1 Auditor Information

The auditor's agency name, address, phone number, and Attestation letter with a brief audit summary report indicating compliance must be included in the full disclosure report. A statement should be included specifying who to contact in order to obtain further information regarding the audit process.

This benchmark was audited by:

InfoSizing

Doug Johnson
63 Lourdes Drive
Leominster, MA 01453-6709

Phone: +1 (978) 343-6562.

www.sizing.com

9.2 Attestation Letter

The auditor's attestation letter is included in the following pages.

InfoSizing
The Right Metric For Sizing IT

TPC Transaction Processing Performance Council
Certified Auditors

Chan Lim (Charlie) Park
Senior Research Engineer
Telecommunications Technology Association (TTA)
Bundang-ro 47, Bundang-gu, Seongnam-city
Gyeonggi-do, 13591, Republic of Korea

May 31, 2017

I verified the TPC Benchmark™ C v5.11.0 performance of the following configuration:

Platform:	Jet-speed™ HHA2212
Operating System:	CentOS 6.6
Database Manager:	Goldilocks v3.1 Standard Edition

The results were:

Performance Metric	**139,909 tpmC**
New-Order 90th %-tile	0.106 Seconds

Tier B (Server)	**Jet-speed™ HHA2212**		
CPUs	2 x Intel Xeon Processor E5-2630 v3 (2.4 GHz, 8-core, 20 MB L3)		
Memory	1.5 TB		
Storage	***Qty***	***Size***	***Type***
	2	300 GB	15K rpm SAS HDD
	10	1.2 TB	10K rpm SAS HDD
	8	64 GB	DRAM SSD

Tier A (Clients)	**TJS104**
CPUs	2 x Intel Xeon Processor E5-2630 v3 (2.4 GHz, 8-core, 20 MB L3)
Memory	32 GB
Storage	2 x 300 GB 15K rpm SAS HDD

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 was properly scaled and populated for 11,000 warehouses
- The ACID properties were met

63 Lourdes Dr. | Leominster, MA 01453 | 978-343-6562 | www.sizing.com

- 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
- All 90% response times were under the specified maximums
- The measurement interval was at least 120 minutes
- Log-switch initiated checkpoints were used during the measurement interval
- The 60-day storage requirement was correctly computed
- The system pricing was verified for major components and maintenance

Additional Audit Notes:

None.

Respectfully Yours,

Doug Johnson, Certified TPC Auditor

63 Lourdes Dr. | Leominster, MA 01453 | 978-343-6562 | www.sizing.com

Appendix A: Source Code

driver.c

```
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include <unistd.h>
#include <fcntl.h>
#include <time.h>
#include <math.h>
#include <errno.h>
#include <assert.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <pthread.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/poll.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#include <tpc.h>
#include <support.h>

extern int   gWarehouseNum;
extern int   gIsAtivate;
extern int   gIsCounting;
extern int   gACIDTest;
extern pid_t gPID;
extern unsigned int gNewOrderCount;

#define USEC_PER_SEC   1000000
#define BROWSER_DELAY  0.1

#define TOTAL_WEIGHT 100000

int   gShmId = -1;
key_t gKeyVal = 1234;
int   gShmOwner = 0;

int gDefaultWeight[5] = { 0, 43001, 4001, 4001, 4001 };
int gConstMinWeight[5] = { 0, };
int gConstMaxWeight[5] = { 0, };
int gMinWeight[5] = { 0, 100000, 100000, 100000, 100000};
int gMaxWeight[5] = { 0, };

TransactionInfo * gTransactionInfo = NULL;
int * gWeight = NULL;
unsigned int * gCount = NULL;
unsigned int * gFailCount = NULL;
unsigned int * gTotal = NULL;
unsigned int * gAtomic = NULL;
int * gShutdown = NULL;

int ConnectTcp( char * aIpAddress,
                short  aPort );

int HttpGetMethod( HttpContext * aHttp );

void InitTransactionMix()
{
    int i;

    gShmId = shmget( gKeyVal,
                     sizeof(TransactionInfo),
                     IPC_EXCL | IPC_CREAT | 0666 );

    if( gShmId != -1 )
    {
        gShmOwner = 1;
    }
    else
    {
        gShmId = shmget( gKeyVal,
                         sizeof(TransactionInfo),
                         IPC_CREAT | 0666 );
    }

    if( gShmId == -1 )
    {
        fprintf( stderr, "shmget error\n" );
        exit(1);
    }

    gTransactionInfo = (TransactionInfo*)shmat( gShmId, NULL, 0 );

    if( gTransactionInfo == (void*)-1 )
    {
        fprintf( stderr, "shmat error\n" );
        exit(1);
    }

    gAtomic    = &gTransactionInfo->mAtomic;
    gTotal     = &gTransactionInfo->mTotal;
    gCount     = gTransactionInfo->mCount;
    gFailCount = gTransactionInfo->mFailCount;
    gWeight    = gTransactionInfo->mWeight;
    gShutdown  = &gTransactionInfo->mStop;

    if( gShmOwner == 1 )
    {
        *gAtomic   = 0;
        *gTotal    = 0;
        *gShutdown = 0;

        for( i = 0; i < 5; i++ )
        {
            gWeight[i]    = gDefaultWeight[i];
            gCount[i]     = 0;
            gFailCount[i] = 0;
        }
    }

    for( i = 1; i < 5; i++ )
    {
        gConstMinWeight[i] = gDefaultWeight[i] - (gDefaultWeight[i]
* 0.05);
        gConstMaxWeight[i] = gDefaultWeight[i] + (gDefaultWeight[i]
* 0.05);
    }
}

void FiniTransactionMix()
{
    if( gShmOwner == 1 )
    {
        printf( "\nTranasction Failure:\n" );
        printf( "  NEW-ORDER    : %u\n", gFailCount[0] );
        printf( "  PAYMENT      : %u\n", gFailCount[1] );
        printf( "  ORDER-STATUS : %u\n", gFailCount[2] );
        printf( "  DELIVERY     : %u\n", gFailCount[3] );
        printf( "  STOCK-LEVEL  : %u\n", gFailCount[4] );

        printf( "\nTranasction Mix:\n" );
        printf( "  PAYMENT (initial/min/adjustment/max)      : %.3f
/ %.3f / %.3f / %.3f\n",
                ((double)gDefaultWeight[1] / 1000.0),
                ((double)gMinWeight[1] / 1000.0),
                ((double)gWeight[1] / 1000.0),
                ((double)gMaxWeight[1] / 1000.0) );
        printf( "  ORDER-STATUS
(initial/min/adjustment/max) :  %.3f /  %.3f /  %.3f /  %.3f\n",
                ((double)gDefaultWeight[2] / 1000.0),
                ((double)gMinWeight[2] / 1000.0),
                ((double)gWeight[2] / 1000.0),
                ((double)gMaxWeight[2] / 1000.0) );
        printf( "  DELIVERY
(initial/min/adjustment/max)     :  %.3f /  %.3f /  %.3f /  %.3f\n",
                ((double)gDefaultWeight[3] / 1000.0),
                ((double)gMinWeight[3] / 1000.0),
                ((double)gWeight[3] / 1000.0),
                ((double)gMaxWeight[3] / 1000.0) );
        printf( "  STOCK-LEVEL
(initial/min/adjustment/max)  :  %.3f /  %.3f /  %.3f /  %.3f\n",
                ((double)gDefaultWeight[4] / 1000.0),
                ((double)gMinWeight[4] / 1000.0),
                ((double)gWeight[4] / 1000.0),
                ((double)gMaxWeight[4] / 1000.0) );
    }

    (void)shmdt( (const void*)gTransactionInfo );

    gTransactionInfo = NULL;
    gWeight    = NULL;
    gCount     = NULL;
    gFailCount = NULL;
    gTotal     = NULL;
    gAtomic    = NULL;

    if( gShmOwner == 1 )
    {
        (void)shmctl( gShmId, IPC_RMID, 0 );
    }

    gShmId = -1;
```

```
}

TransactionType GetTransaction( unsigned int * aSeed )
{
    int sRandom;
    int sDelta;
    unsigned int sTotal;
    double sNewWeight;
    int i;

    if( gIsCounting == 1 )
    {
        sTotal = *gTotal;

        if( ((sTotal % 500) == 0) &&
            (sTotal > 0) &&
            (__sync_bool_compare_and_swap( gAtomic, 0, 1 )) )
        {
            for( i = 1; i < 5; i++ )
            {
                sDelta = (int)(((double)gCount[i] / (double)*gTotal
* TOTAL_WEIGHT) + 0.5) - gDefaultWeight[i];
                sNewWeight = gWeight[i] - sDelta;

                if( sDelta > 0 )
                {
                    gWeight[i] = (sNewWeight < gConstMinWeight[i]) ?
gConstMinWeight[i] : sNewWeight;
                }
                else if( sDelta == 0 )
                {
                    gWeight[i] = gDefaultWeight[i];
                }
                else
                {
                    gWeight[i] = sNewWeight > gConstMaxWeight[i] ?
gConstMaxWeight[i] : sNewWeight;
                }

                gMinWeight[i] = (gWeight[i] < gMinWeight[i] ) ?
gWeight[i] : gMinWeight[i];
                gMaxWeight[i] = (gWeight[i] > gMaxWeight[i] ) ?
gWeight[i] : gMaxWeight[i];
            }

            *gAtomic = 0;
        }
    }

    sRandom = rand_r( aSeed ) % TOTAL_WEIGHT;

    for( i = 1; i < 5; i++ )
    {
        if( sRandom < gWeight[i] )
        {
            goto TRANSACTION_TYPE_SELECTED;
        }

        sRandom -= gWeight[i];
    }

    i = 0;

TRANSACTION_TYPE_SELECTED:

    return i;
}


int GetRemoteWarehouse( unsigned int * aSeed, int aHomeWarehouse )
{
    int sRemoteWarehouse;

    if( gWarehouseNum == 1 )
    {
        return aHomeWarehouse;
    }

    while( (sRemoteWarehouse = RandomNumber( aSeed, 1,
gWarehouseNum )) == aHomeWarehouse );

    return sRemoteWarehouse;
}

double GetThinkTime( unsigned int * aSeed, double aMeanThinkTime )
{
    double sThinkTime;

    errno = 0;
    sThinkTime = -log( ((double)rand_r( aSeed )) / RAND_MAX ) *
aMeanThinkTime;

    if( errno == ERANGE )
    {
        sThinkTime = aMeanThinkTime * 10;
    }

    if( sThinkTime > aMeanThinkTime * 10 )
    {
        sThinkTime = aMeanThinkTime * 10;
    }

    return sThinkTime;
}

int RequestNewOrderMenu( HttpContext * aHttp,
                         int           aW_ID )
{
    snprintf( aHttp->mGetMethodBuf,
              sizeof(aHttp->mGetMethodBuf),
              "/%s/NewOrderInput.html?W_ID=%d",
              aHttp->mReqPage,
              aW_ID );

    if( HttpGetMethod( aHttp ) != 0 )
    {
        if( *gShutdown == 0 )
        {
            printf("HttpGet error \n");
        }

        return -1;
    }

    return 0;
}

int RequestNewOrderTransaction( HttpContext * aHttp,
                                NewOrderReq * aNewOrderReq )
{
    int  sLen = 0;
    int i;

    sLen = snprintf( aHttp->mGetMethodBuf,
                     sizeof(aHttp->mGetMethodBuf),
                     "/%s/NewOrder?W_ID=%d&D_ID=%d&C_ID=%d&",
                     aHttp->mReqPage,
                     aNewOrderReq->mW_ID,
                     aNewOrderReq->mD_ID,
                     aNewOrderReq->mC_ID );

    for( i = 0; i < aNewOrderReq->mOL_CNT; i++ )
    {
        sLen += snprintf( aHttp->mGetMethodBuf + sLen,
                          sizeof(aHttp->mGetMethodBuf) - sLen,
                          "OL_SUPPLY_W_ID_%d=%d&",
                          i,
                          aNewOrderReq->mItem[i].mOL_SUPPLY_W_ID );

        sLen += snprintf( aHttp->mGetMethodBuf + sLen,
                          sizeof(aHttp->mGetMethodBuf) - sLen,
                          "OL_I_ID_%d=%d&",
                          i,
                          aNewOrderReq->mItem[i].mOL_I_ID );

        sLen += snprintf( aHttp->mGetMethodBuf + sLen,
                          sizeof(aHttp->mGetMethodBuf) - sLen,
                          "OL_QUANTITY_%d=%d&",
                          i,
                          aNewOrderReq->mItem[i].mOL_QUANTITY );
    }

    assert( strlen(aHttp->mGetMethodBuf) < sizeof(aHttp-
>mGetMethodBuf) - 1 );

    if( HttpGetMethod( aHttp ) != 0 )
    {
        if( *gShutdown == 0 )
        {
            printf("HttpGet error \n");
        }

        return -1;
    }

    return 0;
}

int GetACIDData( HttpContext * aHttp )
{
    char * sPos = NULL;
    char   sIDString[9];
    int    sID = 0;

    sPos = strstr( aHttp->mRecvBuf, "Order Number:" );
    sPos += 14;

    memcpy( sIDString, sPos, 8 );
    sIDString[8] = '\0';
```

```
    sID = atoi( sIDString );

    return sID;
}

int RequestPaymentMenu( HttpContext * aHttp,
                        int           aW_ID )
{
    snprintf( aHttp->mGetMethodBuf,
              sizeof(aHttp->mGetMethodBuf),
              "/%s/PaymentInput.html?W_ID=%d",
              aHttp->mReqPage,
              aW_ID );

    if( HttpGetMethod( aHttp ) != 0 )
    {
        if( *gShutdown == 0 )
        {
            printf("HttpGet error \n");
        }

        return -1;
    }

    return 0;
}

int RequestPaymentTransaction( HttpContext * aHttp,
                               PaymentReq  * aPaymentReq )
{
    int  sLen = 0;

    sLen = snprintf( aHttp->mGetMethodBuf,
                     sizeof(aHttp->mGetMethodBuf),
                     "/%s/Payment?W_ID=%d&D_ID=%d&",
                     aHttp->mReqPage,
                     aPaymentReq->mW_ID,
                     aPaymentReq->mD_ID );

    if( aPaymentReq->mC_ID != 0 )
    {
        sLen += snprintf( aHttp->mGetMethodBuf + sLen,
                          sizeof(aHttp->mGetMethodBuf) - sLen,
                          "C_ID=%d&C_W_ID=%d&C_D_ID=%d&C_LAST=&",
                          aPaymentReq->mC_ID,
                          aPaymentReq->mC_W_ID,
                          aPaymentReq->mC_D_ID );

    }
    else
    {
        sLen += snprintf( aHttp->mGetMethodBuf + sLen,
                          sizeof(aHttp->mGetMethodBuf) - sLen,
                          "C_ID=&C_W_ID=%d&C_D_ID=%d&C_LAST=%s&",
                          aPaymentReq->mC_W_ID,
                          aPaymentReq->mC_D_ID,
                          aPaymentReq->mC_LAST );
    }

    sLen += snprintf( aHttp->mGetMethodBuf + sLen,
                      sizeof(aHttp->mGetMethodBuf) - sLen,
                      "H_AMOUNT=%.2f",
                      aPaymentReq->mH_AMOUNT );

    assert( strlen(aHttp->mGetMethodBuf) < sizeof(aHttp-
>mGetMethodBuf) - 1 );

    if( HttpGetMethod( aHttp ) != 0 )
    {
        if( *gShutdown == 0 )
        {
            printf("HttpGet error \n");
        }

        return -1;
    }

    return 0;
}

int RequestOrderStatusMenu( HttpContext * aHttp,
                            int           aW_ID )
{
    snprintf( aHttp->mGetMethodBuf,
              sizeof(aHttp->mGetMethodBuf),
              "/%s/OrderStatusInput.html?W_ID=%d",
              aHttp->mReqPage,
              aW_ID );

    if( HttpGetMethod( aHttp ) != 0 )
    {
        if( *gShutdown == 0 )
        {
            printf("HttpGet error \n");
        }

        return -1;
    }

    return 0;
}

int RequestOrderStatusTransaction( HttpContext    * aHttp,
                                   OrderStatusReq *
aOrderStatusReq )
{
    int  sLen = 0;

    sLen = snprintf( aHttp->mGetMethodBuf,
                     sizeof(aHttp->mGetMethodBuf),
                     "/%s/OrderStatus?W_ID=%d&D_ID=%d&",
                     aHttp->mReqPage,
                     aOrderStatusReq->mW_ID,
                     aOrderStatusReq->mD_ID );

    if( aOrderStatusReq->mC_ID != 0 )
    {
        sLen += snprintf( aHttp->mGetMethodBuf + sLen,
                          sizeof(aHttp->mGetMethodBuf) - sLen,
                          "C_ID=%d&C_LAST=",
                          aOrderStatusReq->mC_ID );

    }
    else
    {
        sLen += snprintf( aHttp->mGetMethodBuf + sLen,
                          sizeof(aHttp->mGetMethodBuf) - sLen,
                          "C_ID=&C_LAST=%s",
                          aOrderStatusReq->mC_LAST );
    }

    assert( strlen(aHttp->mGetMethodBuf) < sizeof(aHttp-
>mGetMethodBuf) - 1 );

    if( HttpGetMethod( aHttp ) != 0 )
    {
        if( *gShutdown == 0 )
        {
            printf("HttpGet error \n");
        }

        return -1;
    }

    return 0;
}

int RequestDeliveryMenu( HttpContext * aHttp,
                         int           aW_ID )
{
    snprintf( aHttp->mGetMethodBuf,
              sizeof(aHttp->mGetMethodBuf),
              "/%s/DeliveryInput.html?WID=%d",
              aHttp->mReqPage,
              aW_ID );

    if( HttpGetMethod( aHttp ) != 0 )
    {
        if( *gShutdown == 0 )
        {
            printf("HttpGet error \n");
        }

        return -1;
    }

    return 0;
}

int RequestDeliveryTransaction( HttpContext * aHttp,
                                DeliveryReq * aDeliveryReq )
{
    snprintf( aHttp->mGetMethodBuf,
              sizeof(aHttp->mGetMethodBuf),
              "/%s/Delivery?W_ID=%d&O_CARRIER_ID=%d",
              aHttp->mReqPage,
              aDeliveryReq->mW_ID,
              aDeliveryReq->mO_CARRIER_ID );

    assert( strlen(aHttp->mGetMethodBuf) < sizeof(aHttp-
>mGetMethodBuf) - 1 );

    if( HttpGetMethod( aHttp ) != 0 )
    {
        if( *gShutdown == 0 )
        {
            printf("HttpGet error \n");
        }

        return -1;
```

```
    }

    return 0;
}

int RequestStockLevelMenu( HttpContext * aHttp,
                           int           aW_ID,
                           int           aD_ID )
{
    snprintf( aHttp->mGetMethodBuf,
              sizeof(aHttp->mGetMethodBuf),
              "/%s/StockLevelInput.html?W_ID=%d&D_ID=%d",
              aHttp->mReqPage,
              aW_ID,
              aD_ID );

    if( HttpGetMethod( aHttp ) != 0 )
    {
        if( *gShutdown == 0 )
        {
            printf("HttpGet error \n");
        }

        return -1;
    }

    return 0;
}

int RequestStockLevelTransaction( HttpContext   * aHttp,
                                  StockLevelReq * aStockLevelReq )
{
    snprintf( aHttp->mGetMethodBuf,
              sizeof(aHttp->mGetMethodBuf),
              "/%s/StockLevel?W_ID=%d&D_ID=%d&THRESHOLD=%d",
              aHttp->mReqPage,
              aStockLevelReq->mW_ID,
              aStockLevelReq->mD_ID,
              aStockLevelReq->mThreshold );

    assert( strlen(aHttp->mGetMethodBuf) < sizeof(aHttp-
>mGetMethodBuf) - 1 );

    if( HttpGetMethod( aHttp ) != 0 )
    {
        if( *gShutdown == 0 )
        {
            printf("HttpGet error \n");
        }

        return -1;
    }

    return 0;
}

void GenerateNewOrderData( unsigned int * aSeed,
                           int            aW_ID,
                           NewOrderReq  * aNewOrderReq )
{
    int               sOL_CNT = 0;
    int               sRollback = 0;
    int               sUnusedValue = MAXITEMS + 1;
    NewOrderItemReq * sItem = NULL;

    int i;

    aNewOrderReq->mW_ID        = aW_ID;
    aNewOrderReq->mD_ID        = RandomNumber( aSeed, 1,
DIST_PER_WARE );
    aNewOrderReq->mC_ID        = NURand( aSeed, 1023, 1,
CUST_PER_DIST );
    aNewOrderReq->mOL_CNT      = RandomNumber( aSeed, 5, 15 );
    aNewOrderReq->mInvalidItem = 0;
    sRollback                  = RandomNumber( aSeed, 1, 100 );

#if 0
    if( (aW_ID == 45) && (aNewOrderReq->mD_ID == 7) )
    {
        aNewOrderReq->mD_ID = 11;
    }
#endif

    aNewOrderReq->mRemoteItemCount = 0;

    sOL_CNT = aNewOrderReq->mOL_CNT;

    for( i = 0; i < aNewOrderReq->mOL_CNT; i++ )
    {
        sItem = &aNewOrderReq->mItem[i];

        sItem->mOL_I_ID = NURand( aSeed, 8191, 1, MAXITEMS );

        if( (i == sOL_CNT - 1) && (sRollback == 1) )
        {
            sItem->mOL_I_ID = sUnusedValue;
            aNewOrderReq->mInvalidItem = 1;
        }

        if( RandomNumber( aSeed, 1, 100 ) == 1 )
        {
            sItem->mOL_SUPPLY_W_ID = GetRemoteWarehouse( aSeed,
aW_ID );
            aNewOrderReq->mRemoteItemCount++;
        }
        else
        {
            sItem->mOL_SUPPLY_W_ID = aW_ID;
        }

        sItem->mOL_QUANTITY = RandomNumber( aSeed, 1, 10 );
    }
}

void GeneratePaymentData( unsigned int * aSeed,
                          int            aW_ID,
                          PaymentReq   * aPaymentReq,
                          int          * aIsRemote )
{
    aPaymentReq->mW_ID     = aW_ID;
    aPaymentReq->mD_ID     = RandomNumber( aSeed, 1,
DIST_PER_WARE );
    aPaymentReq->mH_AMOUNT = (double)RandomNumber( aSeed, 100,
500000 ) / (double)100;

#if 0
    if( (aW_ID == 45) && (aPaymentReq->mD_ID == 7) )
    {
        aPaymentReq->mD_ID = 11;
    }
#endif

    if( RandomNumber( aSeed, 1, 100 ) <= 60 )
    {
        /* select by last name */
        aPaymentReq->mC_ID = 0;

        Lastname( NURand( aSeed, 255, 0, 999), aPaymentReq-
>mC_LAST );
    }
    else
    {
        /* select by customer id */
        aPaymentReq->mC_ID = NURand( aSeed, 1023, 1,
CUST_PER_DIST );
    }

    if( RandomNumber( aSeed, 1, 100 ) <= 85 )
    {
        *aIsRemote = 0;

        aPaymentReq->mC_W_ID = aW_ID;
        aPaymentReq->mC_D_ID = aPaymentReq->mD_ID;
    }
    else
    {
        *aIsRemote = 1;

        aPaymentReq->mC_W_ID = GetRemoteWarehouse( aSeed, aW_ID );
        aPaymentReq->mC_D_ID = RandomNumber( aSeed, 1,
DIST_PER_WARE );
    }
}

void GenerateOrderStatusData( unsigned int   * aSeed,
                              int              aW_ID,
                              OrderStatusReq * aOrderStatusReq )
{
    aOrderStatusReq->mW_ID = aW_ID;
    aOrderStatusReq->mD_ID = RandomNumber( aSeed, 1,
DIST_PER_WARE );

    if( RandomNumber( aSeed, 1, 100 ) <= 60 )
    {
        /* select by last name */
        aOrderStatusReq->mC_ID = 0;

        Lastname( NURand( aSeed, 255, 0, 999), aOrderStatusReq-
>mC_LAST );
    }
    else
    {
        /* select by customer id */
        aOrderStatusReq->mC_ID = NURand( aSeed, 1023, 1,
CUST_PER_DIST );
    }
}

void GenerateDeliveryData( unsigned int * aSeed,
                           int            aW_ID,
```

```
                          DeliveryReq  * aDeliveryReq )
{
    aDeliveryReq->mW_ID         = aW_ID;
    aDeliveryReq->mO_CARRIER_ID = RandomNumber( aSeed, 1, 10 );
}

void GenerateStockLevelData( unsigned int  * aSeed,
                             int             aW_ID,
                             int             aD_ID,
                             StockLevelReq * aStockLevelReq )
{
    aStockLevelReq->mW_ID      = aW_ID;
    aStockLevelReq->mD_ID      = aD_ID;
    aStockLevelReq->mThreshold = RandomNumber( aSeed, 10, 20 );
}

#define ERROR_MENU                              \
    sTimestamp2   = sTimestamp1;                \
    sTimestamp3   = sTimestamp1;                \
    sTimestamp4   = sTimestamp1;                \
    sIsRollbacked = 0;                          \
    sCount1       = -1;                         \
    sCount2       = -1;                         \
    sACID1        = 0;                          \
    sThinkTime    = 0;

#define ERROR_TRANSACTION                       \
    sTimestamp4   = sTimestamp3;                \
    sIsRollbacked = 0;                          \
    sCount1       = 0;                          \
    sCount2       = -1;                         \
    sACID1        = 0;                          \
    sThinkTime    = 0;

int thread_main( ThreadArg* aArg )
{
    int sW_ID;
    int sTerminalID;

    FILE * sLogFile = NULL;
    char   sLogFileName[128];

    struct timespec sTimestamp1;
    struct timespec sTimestamp2;
    struct timespec sTimestamp3;
    struct timespec sTimestamp4;

    TransactionType   sTransaction;

    TpccArg        * sTpccArg        = NULL;

    NewOrderReq    * sNewOrderReq    = NULL;
    PaymentReq     * sPaymentReq     = NULL;
    OrderStatusReq * sOrderStatusReq = NULL;
    DeliveryReq    * sDeliveryReq    = NULL;
    StockLevelReq  * sStockLevelReq  = NULL;

    int sIsRemote;

    int sIsRollbacked = 0;
    int sCount1 = 0;
    int sCount2 = 0;
    int sACID1  = 0;
    int sACID2  = 0;

    double sThinkTime = 0.0;

    HttpContext  * sHttp = NULL;

    sW_ID       = aArg->mW_ID;
    sTerminalID = aArg->mTerminalID;

    aArg->mSeed      = gPID + sW_ID * 100 + sTerminalID;
    aArg->mTransSeed = gPID + sW_ID * 100 + (sTerminalID + 50);

    /* seed log file */
    sprintf( sLogFileName, "../log/seed_%05d_%02d.log", sW_ID,
sTerminalID );
    sLogFile = fopen( sLogFileName, "w" );

    if( sLogFile == NULL )
    {
        fprintf( stderr, "fail fopen(%s)\n", sLogFileName );
        goto FINISH_LABEL;
    }

    fprintf( sLogFile, "%u %u", aArg->mSeed, aArg->mTransSeed );
    fclose( sLogFile );
    sLogFile = NULL;

    sprintf( sLogFileName, "../log/N_%05d_%02d.log", sW_ID,
sTerminalID );

    sLogFile = fopen( sLogFileName, "w" );

    if( sLogFile == NULL )
    {
        fprintf( stderr, "fail fopen(%s)\n", sLogFileName );
        goto FINISH_LABEL;
    }

    sTpccArg = (TpccArg*)malloc( sizeof(TpccArg) );
    if( sTpccArg == NULL )
    {
        goto FINISH_LABEL;
    }

    sHttp = (HttpContext *)malloc( sizeof(HttpContext) );
    if( sHttp == NULL )
    {
        goto FINISH_LABEL;
    }

    sHttp->mHostIp  = aArg->mWebIP;
    sHttp->mPort    = aArg->mWebPort;
    sHttp->mReqPage = aArg->mWebPage;
    sHttp->mSocket  = -1;

    sHttp->mSocket = ConnectTcp( sHttp->mHostIp, sHttp->mPort );
    if( sHttp->mSocket < 0 )
    {
        printf("TcpConnection error \n");
        goto FINISH_LABEL;
    }

    sNewOrderReq    = &sTpccArg->mNewOrder;
    sPaymentReq     = &sTpccArg->mPayment;
    sOrderStatusReq = &sTpccArg->mOrderStatus;
    sDeliveryReq    = &sTpccArg->mDelivery;
    sStockLevelReq  = &sTpccArg->mStockLevel;

    clock_gettime( CLOCK_REALTIME, &sTimestamp1 );

    sTransaction = GetTransaction( &aArg->mTransSeed );

    switch( sTransaction )
    {
        case TRANSACTION_TYPE_NEW_ORDER :
            if( RequestNewOrderMenu( sHttp,
                                     sW_ID ) == -1 )
            {
                if( *gShutdown == 0 )
                {
                    __sync_fetch_and_add(  &gFailCount[sTransaction],
1 );
                    printf("RequestNewOrderMenu error \n");

                    ERROR_MENU;
                    break;
                }
                else
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp2);

            GenerateNewOrderData( &aArg->mSeed, sW_ID,
sNewOrderReq );

            sleep( KEYING_TIME_NEW_ORDER );

            clock_gettime( CLOCK_REALTIME, &sTimestamp3);

            if( RequestNewOrderTransaction( sHttp,
                                            sNewOrderReq ) == -1 )
            {
                if( *gShutdown == 0 )
                {
                    __sync_fetch_and_add(  &gFailCount[sTransaction],
1 );
                    printf("RequestNewOrderTransaction error \n");

                    ERROR_TRANSACTION;
                    break;
                }
                else
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp4);

            sIsRollbacked = sNewOrderReq->mInvalidItem;
```

```
            sCount1       = sNewOrderReq->mOL_CNT;
            sCount2       = sNewOrderReq->mRemoteItemCount;
            sACID1        = sNewOrderReq->mD_ID;

            if( gACIDTest == 1 )
            {
                sACID2 = GetACIDData( sHttp );
            }
            else
            {
                sACID2 = 0;
            }

            sThinkTime = GetThinkTime( &aArg->mSeed,
THINK_TIME_NEW_ORDER );

            if( gIsCounting == 1 )
            {
                __sync_fetch_and_add( &gNewOrderCount, 1 );
            }
            break;
        case TRANSACTION_TYPE_PAYMENT :
            if( RequestPaymentMenu( sHttp,
                                    sW_ID ) == -1 )
            {
                if( *gShutdown == 0 )
                {
                    __sync_fetch_and_add(  &gFailCount[sTransaction],
1 );
                    printf("RequestPaymentMenu error \n");

                    ERROR_MENU;
                    break;
                }
                else
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp2);

            GeneratePaymentData( &aArg->mSeed, sW_ID, sPaymentReq,
&sIsRemote );

            sleep( KEYING_TIME_PAYMENT );

            clock_gettime( CLOCK_REALTIME, &sTimestamp3);

            if( RequestPaymentTransaction( sHttp,
                                           sPaymentReq ) == -1 )
            {
                if( *gShutdown == 0 )
                {
                    __sync_fetch_and_add(  &gFailCount[sTransaction],
1 );
                    printf("RequestPaymentTransaction error \n");

                    ERROR_TRANSACTION;
                    break;
                }
                else
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp4);

            sIsRollbacked = 0;
            sCount1       = sIsRemote;
            sCount2       = (sPaymentReq->mC_ID == 0) ? 1 : 0;
            sACID1        = 0;
            sACID2        = 0;

            sThinkTime = GetThinkTime( &aArg->mSeed,
THINK_TIME_PAYMENT );

            break;
        case TRANSACTION_TYPE_ORDER_STATUS :
            if( RequestOrderStatusMenu( sHttp,
                                        sW_ID ) == -1 )
            {
                if( *gShutdown == 0 )
                {
                    __sync_fetch_and_add(  &gFailCount[sTransaction],
1 );
                    printf("RequestOrderStatusMenu error \n");

                    ERROR_MENU;
                    break;
                }
                else
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp2);

            GenerateOrderStatusData( &aArg->mSeed, sW_ID,
sOrderStatusReq );

            sleep( KEYING_TIME_ORDER_STATUS );

            clock_gettime( CLOCK_REALTIME, &sTimestamp3);

            if( RequestOrderStatusTransaction( sHttp,
                                               sOrderStatusReq ) ==
-1 )
            {
                if( *gShutdown == 0 )
                {
                    __sync_fetch_and_add(  &gFailCount[sTransaction],
1 );
                    printf("RequestOrderStatusTransaction error
\n");

                    ERROR_TRANSACTION;
                    break;
                }
                else
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp4);

            sIsRollbacked = 0;
            sCount1       = (sOrderStatusReq->mC_ID == 0) ? 1 : 0;
            sCount2       = 0;
            sACID1        = 0;
            sACID2        = 0;

            sThinkTime = GetThinkTime( &aArg->mSeed,
THINK_TIME_ORDER_STATUS );

            break;
        case TRANSACTION_TYPE_DELIVERY :
            if( RequestDeliveryMenu( sHttp,
                                     sW_ID ) == -1 )
            {
                if( *gShutdown == 0 )
                {
                    __sync_fetch_and_add(  &gFailCount[sTransaction],
1 );
                    printf("RequestDeliveryMenu error \n");

                    ERROR_MENU;
                    break;
                }
                else
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp2);

            GenerateDeliveryData( &aArg->mSeed, sW_ID,
sDeliveryReq );

            sleep( KEYING_TIME_DELIVERY );

            clock_gettime( CLOCK_REALTIME, &sTimestamp3);

            if( RequestDeliveryTransaction( sHttp,
                                            sDeliveryReq ) == -1 )
            {
                if( *gShutdown == 0 )
                {
                    __sync_fetch_and_add(  &gFailCount[sTransaction],
1 );
                    printf("RequestDeliveryTransaction error \n");

                    ERROR_TRANSACTION;
                    break;
                }
                else
```

```
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp4);

            sIsRollbacked = 0;
            sCount1       = 0;
            sCount2       = 0;
            sACID1        = 0;
            sACID2        = 0;

            sThinkTime = GetThinkTime( &aArg->mSeed,
THINK_TIME_DELIVERY );

            break;
        case TRANSACTION_TYPE_STOCK_LEVEL :
            if( RequestStockLevelMenu( sHttp,
                                       sW_ID,
                                       sTerminalID ) == -1 )
            {
                if( *gShutdown == 0 )
                {
                    __sync_fetch_and_add(  &gFailCount[sTransaction],
1 );
                    printf("RequestStockLevelMenu error \n");

                    ERROR_MENU;
                    break;
                }
                else
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp2);

            GenerateStockLevelData( &aArg->mSeed, sW_ID,
sTerminalID, sStockLevelReq );

            sleep( KEYING_TIME_STOCK_LEVEL );

            clock_gettime( CLOCK_REALTIME, &sTimestamp3);

            if( RequestStockLevelTransaction( sHttp,
                                              sStockLevelReq ) == -
1 )
            {
                if( *gShutdown == 0 )
                {
                    __sync_fetch_and_add(  &gFailCount[sTransaction],
1 );
                    printf("RequestStockLevelTransaction error \n");

                    ERROR_TRANSACTION;
                    break;
                }
                else
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp4);

            sIsRollbacked = 0;
            sCount1       = 0;
            sCount2       = 0;
            sACID1        = 0;
            sACID2        = 0;

            sThinkTime = GetThinkTime( &aArg->mSeed,
THINK_TIME_STOCK_LEVEL );

            break;
        default :
            break;
    }

    while( 1 )
    {

    RETRY_LABEL :

        fprintf( sLogFile, "\n*%d> %lf %lf %lf %d %d %d %d %d ",
                 (int)sTransaction,
                 GetTimespecDiff( &sTimestamp1, &sTimestamp2 ),
                 GetTimespecDiff( &sTimestamp2, &sTimestamp3 ),
                 GetTimespecDiff( &sTimestamp3, &sTimestamp4 ),
                 sIsRollbacked,
                 sCount1,
                 sCount2,
                 sACID1,
                 sACID2 );

        usleep( (long)(sThinkTime * USEC_PER_SEC) );

        if( gIsCounting == 1 )
        {
            __sync_fetch_and_add( gTotal, 1 );
            __sync_fetch_and_add( &gCount[sTransaction], 1 );
        }

        clock_gettime( CLOCK_REALTIME, &sTimestamp1 );

        fprintf( sLogFile, "%lf %ld %ld ",
                 GetTimespecDiff( &sTimestamp4, &sTimestamp1 ),
                 sTimestamp1.tv_sec,
                 sTimestamp1.tv_nsec );

        if( (gIsAtivate == 0) || (*gShutdown == 1) )
        {
            break;
        }

        sTransaction = GetTransaction( &aArg->mTransSeed );

        switch( sTransaction )
        {
            case TRANSACTION_TYPE_NEW_ORDER :
                if( RequestNewOrderMenu( sHttp,
                                         sW_ID ) == -1 )
                {
                    if( *gShutdown == 0 )
                    {

__sync_fetch_and_add( &gFailCount[sTransaction], 1 );
                        printf("RequestNewOrderMenu error \n");

                        ERROR_MENU;

                        goto RETRY_LABEL;
                    }
                    else
                    {
                        goto FINISH_LABEL;
                    }
                }

                usleep( BROWSER_DELAY * USEC_PER_SEC );

                clock_gettime( CLOCK_REALTIME, &sTimestamp2);

                GenerateNewOrderData( &aArg->mSeed, sW_ID,
sNewOrderReq );

                sleep( KEYING_TIME_NEW_ORDER );

                clock_gettime( CLOCK_REALTIME, &sTimestamp3);

                if( RequestNewOrderTransaction( sHttp,
                                                sNewOrderReq ) == -
1 )
                {
                    if( *gShutdown == 0 )
                    {

__sync_fetch_and_add( &gFailCount[sTransaction], 1 );
                        printf("RequestNewOrderTransaction error
\n");

                        ERROR_TRANSACTION;

                        goto RETRY_LABEL;
                    }
                    else
                    {
                        goto FINISH_LABEL;
                    }
                }

                usleep( BROWSER_DELAY * USEC_PER_SEC );

                clock_gettime( CLOCK_REALTIME, &sTimestamp4);

                sIsRollbacked = sNewOrderReq->mInvalidItem;
                sCount1       = sNewOrderReq->mOL_CNT;
                sCount2       = sNewOrderReq->mRemoteItemCount;
                sACID1        = sNewOrderReq->mD_ID;

                if( gACIDTest == 1 )
                {
```

```
                sACID2 = GetACIDData( sHttp );
            }
            else
            {
                sACID2 = 0;
            }

            sThinkTime = GetThinkTime( &aArg->mSeed,
THINK_TIME_NEW_ORDER );

            if( gIsCounting == 1 )
            {
                __sync_fetch_and_add( &gNewOrderCount, 1 );
            }
            break;
        case TRANSACTION_TYPE_PAYMENT :
            if( RequestPaymentMenu( sHttp,
                                    sW_ID ) == -1 )
            {
                if( *gShutdown == 0 )
                {

__sync_fetch_and_add( &gFailCount[sTransaction], 1 );
                    printf("RequestPaymentMenu error \n");

                    ERROR_MENU;

                    goto RETRY_LABEL;
                }
                else
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp2);

            GeneratePaymentData( &aArg->mSeed, sW_ID,
sPaymentReq, &sIsRemote );

            sleep( KEYING_TIME_PAYMENT );

            clock_gettime( CLOCK_REALTIME, &sTimestamp3);

            if( RequestPaymentTransaction( sHttp,
                                           sPaymentReq ) == -1 )
            {
                if( *gShutdown == 0 )
                {

__sync_fetch_and_add( &gFailCount[sTransaction], 1 );
                    printf("RequestPaymentTransaction error
\n");

                    ERROR_TRANSACTION;

                    goto RETRY_LABEL;
                }
                else
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp4);

            sIsRollbacked = 0;
            sCount1       = sIsRemote;
            sCount2       = (sPaymentReq->mC_ID == 0) ? 1 : 0;
            sACID1        = 0;
            sACID2        = 0;

            sThinkTime = GetThinkTime( &aArg->mSeed,
THINK_TIME_PAYMENT );

            break;
        case TRANSACTION_TYPE_ORDER_STATUS :
            if( RequestOrderStatusMenu( sHttp,
                                        sW_ID ) == -1 )
            {
                if( *gShutdown == 0 )
                {

__sync_fetch_and_add( &gFailCount[sTransaction], 1 );
                    printf("RequestOrderStatusMenu error \n");

                    ERROR_MENU;

                    goto RETRY_LABEL;
                }
                else
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp2);

            GenerateOrderStatusData( &aArg->mSeed, sW_ID,
sOrderStatusReq );

            sleep( KEYING_TIME_ORDER_STATUS );

            clock_gettime( CLOCK_REALTIME, &sTimestamp3);

            if( RequestOrderStatusTransaction( sHttp,
                                               sOrderStatusReq )
== -1 )
            {
                if( *gShutdown == 0 )
                {

__sync_fetch_and_add( &gFailCount[sTransaction], 1 );
                    printf("RequestOrderStatusTransaction error
\n");

                    ERROR_TRANSACTION;

                    goto RETRY_LABEL;
                }
                else
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp4);

            sIsRollbacked = 0;
            sCount1       = (sOrderStatusReq->mC_ID == 0) ? 1 :
0;
            sCount2       = 0;
            sACID1        = 0;
            sACID2        = 0;

            sThinkTime = GetThinkTime( &aArg->mSeed,
THINK_TIME_ORDER_STATUS );

            break;
        case TRANSACTION_TYPE_DELIVERY :
            if( RequestDeliveryMenu( sHttp,
                                     sW_ID ) == -1 )
            {
                if( *gShutdown == 0 )
                {

__sync_fetch_and_add( &gFailCount[sTransaction], 1 );
                    printf("RequestDeliveryMenu error \n");

                    ERROR_MENU;

                    goto RETRY_LABEL;
                }
                else
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp2);

            GenerateDeliveryData( &aArg->mSeed, sW_ID,
sDeliveryReq );

            sleep( KEYING_TIME_DELIVERY );

            clock_gettime( CLOCK_REALTIME, &sTimestamp3);

            if( RequestDeliveryTransaction( sHttp,
                                            sDeliveryReq ) == -
1 )
            {
                if( *gShutdown == 0 )
                {

__sync_fetch_and_add( &gFailCount[sTransaction], 1 );
                    printf("RequestDeliveryTransaction error
\n");

                    ERROR_TRANSACTION;
```

```
                    goto RETRY_LABEL;
                }
                else
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp4);

            sIsRollbacked = 0;
            sCount1       = 0;
            sCount2       = 0;
            sACID1        = 0;
            sACID2        = 0;

            sThinkTime = GetThinkTime( &aArg->mSeed,
THINK_TIME_DELIVERY );

            break;
        case TRANSACTION_TYPE_STOCK_LEVEL :
            if( RequestStockLevelMenu( sHttp,
                                       sW_ID,
                                       sTerminalID ) == -1 )
            {
                if( *gShutdown == 0 )
                {

__sync_fetch_and_add( &gFailCount[sTransaction], 1 );
                    printf("RequestStockLevelMenu error \n");

                    ERROR_MENU;

                    goto RETRY_LABEL;
                }
                else
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp2);

            GenerateStockLevelData( &aArg->mSeed, sW_ID,
sTerminalID, sStockLevelReq );

            sleep( KEYING_TIME_STOCK_LEVEL );

            clock_gettime( CLOCK_REALTIME, &sTimestamp3);

            if( RequestStockLevelTransaction( sHttp,
                                              sStockLevelReq )
== -1 )
            {
                if( *gShutdown == 0 )
                {

__sync_fetch_and_add( &gFailCount[sTransaction], 1 );
                    printf("RequestStockLevelTransaction error
\n");

                    ERROR_TRANSACTION;

                    goto RETRY_LABEL;
                }
                else
                {
                    goto FINISH_LABEL;
                }
            }

            usleep( BROWSER_DELAY * USEC_PER_SEC );

            clock_gettime( CLOCK_REALTIME, &sTimestamp4);

            sIsRollbacked = 0;
            sCount1       = 0;
            sCount2       = 0;
            sACID1        = 0;
            sACID2        = 0;

            sThinkTime = GetThinkTime( &aArg->mSeed,
THINK_TIME_STOCK_LEVEL );

            break;
        default :
            break;
    }
}
```

```
    if( sHttp->mSocket != -1 )
    {
        if( close(sHttp->mSocket) != 0 )
        {
            printf("close error %d, %d\n", sHttp->mSocket, errno);
        }
        sHttp->mSocket = -1;
    }

    if( sLogFile != NULL )
    {
        fclose( sLogFile );
        sLogFile = NULL;
    }

    fprintf( stdout, "." );
    fflush( stdout );

    free( (char*)sTpccArg );
    sTpccArg = NULL;

    free( (char*)sHttp );
    sHttp = NULL;

    return 0;

FINISH_LABEL :

    gIsAtivate = 0;

    if( sHttp->mSocket != -1 )
    {
        close( sHttp->mSocket );
        sHttp->mSocket = -1;
    }

    if( sLogFile != NULL )
    {
        fclose( sLogFile );
        sLogFile = NULL;
    }

    if( sTpccArg != NULL )
    {
        free( (char*)sTpccArg );
        sTpccArg = NULL;
    }

    if( sHttp != NULL )
    {
        free( (char*)sHttp );
        sHttp = NULL;
    }

    return -1;
}

int ConnectTcp( char * aIpAddress,
                short  aPort )
{
    struct sockaddr_in sServerAddr;
    int sSocket;

    sServerAddr.sin_family = AF_INET;
    sServerAddr.sin_addr.s_addr = inet_addr( aIpAddress );
    sServerAddr.sin_port = htons( aPort );

    sSocket = socket( AF_INET, SOCK_STREAM, 0 );
    if( sSocket == -1 )
    {
        printf("socket error %d\n", errno);
        return -1;
    }

RETRY:
    if( connect( sSocket, (struct sockaddr *)&sServerAddr ,
sizeof(sServerAddr)) < 0 )
    {
        if( errno == ETIMEDOUT )
        {
            printf("connection timeout => retry \n");
            goto RETRY;
        }
        printf("connect error %d\n", errno);
        return -1;
    }

    return sSocket;
}

int SendTcp( int    aSocket,
             char * aBuff,
             int    aMsgSize )
{
    int sSentLen = 0;
```

```
    int sLen = 0;

    while( 1 )
    {
        do
        {
            errno = 0;
            sLen = send( aSocket, aBuff + sSentLen, aMsgSize -
sSentLen, MSG_NOSIGNAL );
        } while( (sLen == -1) && (errno == EINTR) );

        if( sLen < 0 )
        {
            printf("send error \n");
            return -1;
        }
        sSentLen += sLen;
        if( sSentLen == aMsgSize )
        {
            break;
        }
    }

    return 0;
}

int RecvTcp( int    aSocket,
             char * aBuff,
             int    aMsgSize )
{
    int sReceivedLen = 0;
    int sLen = 0;
    struct pollfd sPollFd[1];
    int sSigFdNum;

    while( 1 )
    {
        sPollFd[0].fd = aSocket;
        sPollFd[0].events = POLLIN;
        sPollFd[0].revents = 0;

        do
        {
            errno = 0;
            sSigFdNum = poll( sPollFd, 1, 1000 /* millisecond */ );
        } while( (sSigFdNum == -1) && (errno == EINTR) );

        if( sSigFdNum == 0 )
        {
            /**
             * timeout
             */

            if( *gShutdown == 1 )
            {
                return -1;
            }

            continue;
        }
        else if( sSigFdNum < 0 )
        {
            printf("poll error [errno %d]\n", errno);
            return -1;
        }

        do
        {
            errno = 0;
            sLen = recv( aSocket, aBuff + sReceivedLen, aMsgSize -
sReceivedLen, 0 );
        } while( (sLen == -1) && (errno == EINTR) );

        if( sLen == 0 )
        {
            aBuff[sReceivedLen] = 0x00;
            printf("recv tcp : disconnect [%s]\n", aBuff);
            return -1;
        }
        else if( sLen < 0 )
        {
            if( errno == EAGAIN )
            {
                continue;
            }

            printf("recv error %d \n", errno);
            return -1;
        }

        sReceivedLen += sLen;
        if( sReceivedLen == aMsgSize )
        {
            break;
        }
    }

    return 0;
}


int CheckDisconnect( int    aSocket )
{
    int sLen = 0;
    struct pollfd sPollFd[1];
    int sSigFdNum;
    char   sBuff[1];

    sPollFd[0].fd = aSocket;
    sPollFd[0].events = POLLIN;
    sPollFd[0].revents = 0;

    do
    {
        errno = 0;
        sSigFdNum = poll( sPollFd, 1, 0 /* millisecond */ );
    } while( (sSigFdNum == -1) && (errno == EINTR) );

    if( sSigFdNum == 0 )
    {
        /**
         * timeout
         */
        return 0;
    }
    else if( sSigFdNum < 0 )
    {
        printf("poll error [errno %d]\n", errno);
        return -1;
    }

    do
    {
        errno = 0;
        sLen = recv( aSocket, sBuff, 1, MSG_PEEK );
    } while( (sLen == -1) && (errno == EINTR) );

    if( sLen == 0 )
    {
        // disconnect
        return -1;
    }
    else if( sLen < 0 )
    {
        if( errno == EAGAIN )
        {
            return 0;
        }

        printf("recv error %d \n", errno);
        return -1;
    }

    return 0;
}

int RecvHttpHeader( int    aSocket,
                    char * aHeader,
                    size_t aBufSize,
                    int  * aBodyLen )
{
    int    sReceivedLen = 0;
    char * sPos = NULL;

    while( 1 )
    {
        assert( sReceivedLen < aBufSize );

        if( RecvTcp( aSocket, aHeader + sReceivedLen, 1 ) < 0 )
        {
            aHeader[sReceivedLen] = 0x00;
            return -1;
        }

        if( (aHeader[sReceivedLen]     == '\n') &&
            (aHeader[sReceivedLen - 1] == '\r') &&
            (aHeader[sReceivedLen - 2] == '\n') &&
            (aHeader[sReceivedLen - 3] == '\r') )
        {
            sPos = strstr( aHeader, "Content-Length: " );

            if( sPos != NULL )
            {
                sPos += strlen( "Content-Length: " );

                *aBodyLen = atoi( sPos );
            }
            else
            {
```

```
                sPos = strstr( aHeader, "Transfer-Encoding:
chunked" );
                assert( sPos != NULL );

                *aBodyLen = 0;
            }

            break;
        }

        sReceivedLen += 1;
    }

    return 0;
}


int RecvChunkSize( int    aSocket,
                   int  * aChunkLen )
{
    int sReceivedLen = 0;
    char sChunkSizeBuf[16];

    while( 1 )
    {
        assert( sReceivedLen < sizeof(sChunkSizeBuf) );

        if( RecvTcp( aSocket, sChunkSizeBuf + sReceivedLen, 1 ) <
0 )
        {
            sChunkSizeBuf[sReceivedLen] = 0x00;
            return -1;
        }

        if( sChunkSizeBuf[sReceivedLen] == '\n' &&
sChunkSizeBuf[sReceivedLen - 1] == '\r' )
        {
            if( sReceivedLen == 1 )
            {
                // case : data == CRLF => retry receive
                sReceivedLen = 0;
                continue;
            }


            sChunkSizeBuf[sReceivedLen - 1] = 0x00;

            // hexadecimal -> decimal
            *aChunkLen = (int)strtol( sChunkSizeBuf, NULL, 16 );

            break;
        }

        sReceivedLen += 1;
    }

    return 0;
}


int HttpGetMethod( HttpContext * aHttp )
{
    int sRtn = 0;
    int sBodyLen;
    int sChunkLen;

    assert( strlen(aHttp->mGetMethodBuf) < 2048 );  // http get
spec

    sprintf( aHttp->mSendBuf,
             "GET %s HTTP/1.1\n"
             "User-Agent: tpcc\n"
             "Host: %s\n"
             "Connection: Keep-Alive\n"
             "\n",
             aHttp->mGetMethodBuf,
             aHttp->mHostIp );

    if( SendTcp( aHttp->mSocket, aHttp->mSendBuf, strlen(aHttp-
>mSendBuf) ) != 0 )
    {
        printf("sendWait error\n");
        sRtn = -1;
        goto END;
    }

    if( RecvHttpHeader( aHttp->mSocket, aHttp->mRecvBuf,
sizeof(aHttp->mRecvBuf), &sBodyLen ) != 0 )
    {
        sRtn = -1;
        goto END;
    }

    if( sBodyLen > 0 )
    {
        // content-length
        assert( sBodyLen < sizeof(aHttp->mRecvBuf) );   // check
buf overflow

        if( RecvTcp( aHttp->mSocket, aHttp->mRecvBuf, sBodyLen ) !=
0 )
        {
            sRtn = -1;
            goto END;
        }

        aHttp->mRecvBuf[sBodyLen] = 0x00;
    }
    else
    {
        // chunked
        sBodyLen = 0;

        while( 1 )
        {
            if( RecvChunkSize( aHttp->mSocket, &sChunkLen ) != 0 )
            {
                sRtn = -1;
                goto END;
            }

            if( sChunkLen == 0 )
            {
                aHttp->mRecvBuf[sBodyLen] = 0x00;
                break;
            }

            assert( sBodyLen + sChunkLen < sizeof(aHttp-
>mRecvBuf) );   // check buf overflow
            if( RecvTcp( aHttp->mSocket, aHttp->mRecvBuf + sBodyLen,
sChunkLen ) != 0 )
            {
                sRtn = -1;
                goto END;
            }
            sBodyLen += sChunkLen;
        }
    }

    if( strstr(aHttp->mRecvBuf, "Error") != NULL )
    {
        sRtn = -1;
        printf("request : [%s]\n", aHttp->mGetMethodBuf);
        printf("response : [%s]\n\n", aHttp->mRecvBuf);
    }

END:

    if( sRtn != 0 )
    {
        if( *gShutdown == 0 )
        {
            printf( "HttpError : get : %s \n", aHttp-
>mGetMethodBuf );
        }
    }

    if( CheckDisconnect( aHttp->mSocket ) < 0 )
    {
        close( aHttp->mSocket );

        if( *gShutdown == 0 )
        {
            aHttp->mSocket = ConnectTcp( aHttp->mHostIp, aHttp-
>mPort );
            if( aHttp->mSocket < 0 )
            {
                printf("Tcp connect error \n");
            }
        }
    }

    return sRtn;
}
```

free space.c

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

#include <time.h>
#include <signal.h>

#include <math.h>
#include <sys/time.h>

#include <goldilocks.h>
```

```
#define GL_TRY( stmt, aExpression )                                 \
    do                                                              \
    {                                                               \
        if( !(SQL_SUCCEEDED( aExpression ) ) )                      \
        {                                                           \
            printf("%s:%d error: ", __FILE__, __LINE__ );           \
            printError(SQL_HANDLE_STMT, stmt);                      \
            goto GLIESE_FINISH_LABEL;                               \
        }                                                           \
    } while( 0 )

#define GLIESE_SQL_TRY( aExpression )                               \
    do                                                              \
    {                                                               \
        if( !(SQL_SUCCEEDED( aExpression ) ) )                      \
        {                                                           \
            printf("%s:%d: error : ", __FILE__, __LINE__ );         \
            goto GLIESE_FINISH_LABEL;                               \
        }                                                           \
    } while( 0 )


#define GLIESE_SUCCESS                          \
    goto GLIESE_SUCCESS_FINISH;                 \
GLIESE_SUCCESS_FINISH:

#define GLIESE_FINISH                           \
    goto GLIESE_FINISH_LABEL;                   \
GLIESE_FINISH_LABEL:

#define SQL_MAX_MESSAGE_LENGTH 512
#define SQL_BUFF_SIZE          2048 * 4

void printError(SQLSMALLINT aHandleType, SQLHANDLE aHandle)
{
    SQLSMALLINT   i = 1;
    SQLRETURN     rc;
    SQLCHAR       sSQLState[6];
    SQLINTEGER    sNativeError;
    SQLSMALLINT   sTextLength;
    SQLCHAR       sMessageText[SQL_MAX_MESSAGE_LENGTH];

    while( 1 )
    {
        rc = SQLGetDiagRec( aHandleType,
                            aHandle,
                            i,
                            sSQLState,
                            (SQLINTEGER*)&sNativeError,
                            sMessageText,
                            SQL_MAX_MESSAGE_LENGTH,
                            &sTextLength );
        if( rc == SQL_NO_DATA ) {
            break;
        }

        printf("[%2d]SQL STATE    : %s\n", i, sSQLState );
        printf("[%2d]NATIVE ERROR : %d\n", i, sNativeError );
        printf("[%2d]MESSAGE      : %s\n", i, sMessageText );
        i++;
    }
}

/* ODBC variables */
SQLHENV     gEnv    = NULL;
SQLHDBC     gDbc    = NULL;
SQLHSTMT    gStmt01 = NULL;
SQLHSTMT    gStmt02 = NULL;
SQLHSTMT    gStmt03 = NULL;
SQLHSTMT    gStmt04 = NULL;
SQLHSTMT    gStmt05 = NULL;
SQLHSTMT    gStmt06 = NULL;

typedef struct _tablespace_volume
{
    /*
     * table column
     */
    SQLCHAR         name                    [128 + 1];      //
tablespace name
    SQLCHAR         id                      [  8 + 1];      //
tablespace id
    SQLCHAR         tot                     [128 + 1];      //
tablespace alloc total

    /*
     * table column indicator
     */
    SQLLEN          indname                 ;
    SQLLEN          indid                   ;
    SQLLEN          indtot                  ;
} tablespace_volume_t;
```

```
typedef struct
{
    /*
     * table column
     */
    SQLCHAR         name                    [128 + 1];      //
tablespace name
    SQLCHAR         tbsId                   [  8 + 1];      //
tablespace name
    SQLCHAR         tot                     [128 + 1];      //
tablespace name
    SQLCHAR         used                    [ 25 + 1];      // used
space
    SQLCHAR         free                    [ 25 + 1];      // free
space
    SQLCHAR         actual_used             [ 25 + 1];      //
actual used space

    /*
     * table column indicator
     */
    SQLLEN          indname                 ;
    SQLLEN          indtbsId                ;
    SQLLEN          indtot                  ;
    SQLLEN          indused                 ;
    SQLLEN          indfree                 ;
    SQLLEN          indactualused           ;
} space_used_t;


typedef struct _object
{
    /*
     * table column
     */
    SQLBIGINT  physicalId                   ;           //
PHYSICAL_ID   BIGINT
    SQLCHAR    id                           [128 + 1];  // ID
    SQLCHAR    name                         [128 + 1];  //
TABLE_NAME or INDEX_NAME   CHARACTER VARYING(128)
    SQLCHAR    schema                       [128 + 1];  // SCHEMA
NAME   CHARACTER VARYING(128)
    SQLCHAR    tbsId                        [  8 + 1];  //
Tablespace ID

    /*
     * table column indicator
     */
    SQLLEN     indphysicalId                ;  // PHYSICAL_ID
BIGINT
    SQLLEN     indid                        ;  // ID
    SQLLEN     indname                      ;  // TABLE_NAME or
INDEX_NAME   CHARACTER VARYING(128)
    SQLLEN     indschema                    ;  // SCHEMA NAME
CHARACTER VARYING(128)
    SQLLEN     indtbsId                     ;  // tablespace id
} object_t;


struct sigaction act_new;
struct sigaction act_old;

void sig_handler( int signo, siginfo_t *siginfo, void *context)
{
    printf( "signal received: si_signo(%d), si_errno(%d),
si_code(%d)\n", signo, siginfo->si_errno, siginfo->si_code );

    switch ( signo )
    {
        case SIGINT:
        case SIGQUIT:
        case SIGABRT:
        case SIGSEGV:
        case SIGTERM:
        case SIGUSR2:
            printf( "signal received: si_signo(%d), si_errno(%d),
si_code(%d)\n", signo, siginfo->si_errno, siginfo->si_code );

            if ( gStmt01 != NULL )
                (void) SQLFreeHandle( SQL_HANDLE_STMT, gStmt01 );
            if ( gStmt02 != NULL )
                (void) SQLFreeHandle( SQL_HANDLE_STMT, gStmt02 );
            if ( gStmt03 != NULL )
                (void) SQLFreeHandle( SQL_HANDLE_STMT, gStmt03 );
            if ( gStmt04 != NULL )
                (void) SQLFreeHandle( SQL_HANDLE_STMT, gStmt04 );
            if ( gStmt05 != NULL )
                (void) SQLFreeHandle( SQL_HANDLE_STMT, gStmt05 );
            if ( gStmt06 != NULL )
                (void) SQLFreeHandle( SQL_HANDLE_STMT, gStmt06 );

            if ( gDbc != NULL )
                SQLDisconnect( gDbc );

            if ( gDbc != NULL )
```

```
                SQLFreeHandle( SQL_HANDLE_DBC, gDbc );

            if ( gEnv != NULL )
                SQLFreeHandle( SQL_HANDLE_ENV, gEnv );

            break;
    }
}



SQLRETURN doJob()
{
    int sState = 0;

    int i = 0;

    long fetchCount01 = 0;
    long fetchCount02 = 0;
    long fetchCount03 = 0;
    long fetchCount04 = 0;
    long fetchCount05 = 0;
    long fetchCount06 = 0;


    int ret01 = 0;
    int ret02 = 0;
    int ret03 = 0;
    int ret04 = 0;
    int ret05 = 0;
    int ret06 = 0;

    char    sSQL01[SQL_BUFF_SIZE] = {0x00,};
    char    sSQL02[SQL_BUFF_SIZE] = {0x00,};
    char    sSQL03[SQL_BUFF_SIZE] = {0x00,};
    char    sSQL04[SQL_BUFF_SIZE] = {0x00,};
    char    sSQL05[SQL_BUFF_SIZE] = {0x00,};
    char    sSQL06[SQL_BUFF_SIZE] = {0x00,};

    tablespace_volume_t                     tablespace_alloc_size;
    space_used_t                            tablespace_used;

    space_used_t                            table_used;
    space_used_t                            index_used;

    object_t                                obj_table;
    object_t                                obj_index;

    GL_TRY( gStmt01, SQLAllocHandle( SQL_HANDLE_STMT, gDbc,
&gStmt01 ) );
    GL_TRY( gStmt02, SQLAllocHandle( SQL_HANDLE_STMT, gDbc,
&gStmt02 ) );
    GL_TRY( gStmt03, SQLAllocHandle( SQL_HANDLE_STMT, gDbc,
&gStmt03 ) );
    GL_TRY( gStmt04, SQLAllocHandle( SQL_HANDLE_STMT, gDbc,
&gStmt04 ) );
    GL_TRY( gStmt05, SQLAllocHandle( SQL_HANDLE_STMT, gDbc,
&gStmt05 ) );
    GL_TRY( gStmt06, SQLAllocHandle( SQL_HANDLE_STMT, gDbc,
&gStmt06 ) );

    sState = 1;

    sprintf(sSQL01,
            " SELECT                                "
            "       PHYSICAL_ID,                    "
            "       TABLE_ID,                       "
            "       TABLE_NAME,                     "
            "       SCHEMA_NAME,                    "
            "       TABLESPACE_ID                   "
            " FROM  DEFINITION_SCHEMA.TABLES A,     "
            "       DEFINITION_SCHEMA.SCHEMATA B    "
            " WHERE 1=1                             "
            " AND   A.OWNER_ID <> 1                 "
            " AND   A.SCHEMA_ID = B.SCHEMA_ID       "
            " AND   PHYSICAL_ID IS NOT NULL         "
            " ORDER BY A.TABLESPACE_ID              "
        );
    GL_TRY( gStmt01, SQLPrepare( gStmt01, (SQLCHAR*) sSQL01,
SQL_NTS ) );

    i = 1;
    GL_TRY( gStmt01, SQLBindCol(gStmt01, i++, SQL_C_SBIGINT,
&obj_table.physicalId, sizeof(obj_table.physicalId),
&obj_table.indphysicalId));
    GL_TRY( gStmt01, SQLBindCol(gStmt01, i++, SQL_C_CHAR,
obj_table.id, sizeof(obj_table.id), &obj_table.indid) );
    GL_TRY( gStmt01, SQLBindCol(gStmt01, i++, SQL_C_CHAR,
obj_table.name, sizeof(obj_table.name), &obj_table.indname) );
    GL_TRY( gStmt01, SQLBindCol(gStmt01, i++, SQL_C_CHAR,
obj_table.schema, sizeof(obj_table.schema), &obj_table.indschema) );
    GL_TRY( gStmt01, SQLBindCol(gStmt01, i++, SQL_C_CHAR,
obj_table.tbsId, sizeof(obj_table.tbsId), &obj_table.indtbsId) );

    sprintf(sSQL02,
            " SELECT                                "
            "       PHYSICAL_ID,                    "
            "       INDEX_ID,                       "
            "       INDEX_NAME,                     "
            "       SCHEMA_NAME,                    "
            "       TABLESPACE_ID                   "
            " FROM  INDEXES A,                      "
            "       DEFINITION_SCHEMA.SCHEMATA B    "
            " WHERE 1=1                             "
            " AND   A.OWNER_ID <> 1                 "
            " AND   A.SCHEMA_ID = B.SCHEMA_ID       "
            " AND   A.TABLE_ID = ?                  "
            " ORDER BY A.INDEX_ID                   "
        );
    GL_TRY( gStmt02, SQLPrepare( gStmt02, (SQLCHAR*) sSQL02,
SQL_NTS ) );

    i = 1;
    GL_TRY( gStmt02, SQLBindCol(gStmt02, i++, SQL_C_SBIGINT,
&obj_index.physicalId, sizeof(obj_index.physicalId),
&obj_index.indphysicalId));
    GL_TRY( gStmt02, SQLBindCol(gStmt02, i++, SQL_C_CHAR,
obj_index.id, sizeof(obj_index.id), &obj_index.indid) );
    GL_TRY( gStmt02, SQLBindCol(gStmt02, i++, SQL_C_CHAR,
obj_index.name, sizeof(obj_index.name), &obj_index.indname) );
    GL_TRY( gStmt02, SQLBindCol(gStmt02, i++, SQL_C_CHAR,
obj_index.schema, sizeof(obj_index.schema), &obj_index.indschema) );
    GL_TRY( gStmt02, SQLBindCol(gStmt02, i++, SQL_C_CHAR,
obj_index.tbsId, sizeof(obj_index.tbsId), &obj_index.indtbsId) );

    i = 1;
    GL_TRY( gStmt02, SQLBindParameter(gStmt02, i++, SQL_PARAM_INPUT,
SQL_C_CHAR, SQL_VARCHAR, sizeof(obj_table.id), 0, obj_table.id,
sizeof(obj_table.id), &obj_table.indid) );

    sprintf(sSQL03,
            " SELECT NAME,          "
            "        ID             "
            " FROM   X$TABLESPACE   "
            " WHERE  1=1            "
        );
    GL_TRY( gStmt03, SQLPrepare( gStmt03, (SQLCHAR*) sSQL03,
SQL_NTS ) );

    i = 1;
    GL_TRY( gStmt03, SQLBindCol(gStmt03, i++, SQL_C_CHAR,
&tablespace_alloc_size.name, sizeof(tablespace_alloc_size.name),
&tablespace_alloc_size.indname));
    GL_TRY( gStmt03, SQLBindCol(gStmt03, i++, SQL_C_CHAR,
&tablespace_alloc_size.id, sizeof(tablespace_alloc_size.id),
&tablespace_alloc_size.indid));

    memset ( &tablespace_alloc_size, 0x00,
sizeof(tablespace_volume_t));
    memset ( &tablespace_used, 0x00, sizeof(space_used_t));
    memset ( &table_used, 0x00, sizeof(space_used_t));
    memset ( &index_used, 0x00, sizeof(space_used_t));

    memset ( &obj_table, 0x00, sizeof(object_t));
    memset ( &obj_index, 0x00, sizeof(object_t));

    printf("\n");
    printf("-------------------------------------------------------
-----------------------------------------\n");
    printf("ID TABLESPACE_NAME                    TOTAL(KB)
USED(KB)         FREE(KB)\n");
    printf("-------------------------------------------------------
-----------------------------------------\n");

    /*
      SQL_sSQL03
    */
    GL_TRY( gStmt03, SQLExecute( gStmt03 ) );
    fetchCount03 = 0;

    while (1)
    {
        ret03 = SQLFetch( gStmt03 );

        if ( ret03 == SQL_NO_DATA )
        {
            if ( fetchCount03 == 0 )
            {
                printf ("SQL03(%d) TABLESPACE0! >x=@4O4Y.\n",
__LINE__);
                break;
            }
            else
            {
                break;
            }
        }
        GL_TRY( gStmt03, ret03 );

        sprintf(sSQL04,
```

```
                " SELECT NAME,       "
                "        TO_CHAR((USED_CNT + FREE_CNT) * EXTSIZE /
1024, '999,999,999,990') TOTAL, "
                "        TO_CHAR(USED_CNT * EXTSIZE / 1024,
'999,999,999,990') USED,                "
                "        TO_CHAR(FREE_CNT * EXTSIZE / 1024,
'999,999,999,990') FREE_SIZE            "
                " FROM
(                                       "
                "       SELECT COUNT(STATE) USED_CNT
"
                "       FROM D$TABLESPACE_EXTENT('%s')
"
                "       WHERE STATE <> 'F'
"
                "      ) A,
"
                "
(                                       "
                "       SELECT COUNT(STATE) FREE_CNT
"
                "       FROM D$TABLESPACE_EXTENT('%s')
"
                "       WHERE STATE = 'F'
"
                "      ) B,
"
                "
(                                       "
                "       SELECT NAME, EXTSIZE
"
                "       FROM X$TABLESPACE
"
                "       WHERE NAME = '%s'
"
                "      ) C
"
                "
"
                , tablespace_alloc_size.name
                , tablespace_alloc_size.name
                , tablespace_alloc_size.name
            );

        i = 1;
        GL_TRY( gStmt04, SQLBindCol(gStmt04, i++, SQL_C_CHAR,
&tablespace_used.name, sizeof(tablespace_used.name),
&tablespace_used.indname));
        GL_TRY( gStmt04, SQLBindCol(gStmt04, i++, SQL_C_CHAR,
&tablespace_used.tot, sizeof(tablespace_used.tot),
&tablespace_used.indtot));
        GL_TRY( gStmt04, SQLBindCol(gStmt04, i++, SQL_C_CHAR,
&tablespace_used.used, sizeof(tablespace_used.used),
&tablespace_used.indused));
        GL_TRY( gStmt04, SQLBindCol(gStmt04, i++, SQL_C_CHAR,
&tablespace_used.free, sizeof(tablespace_used.free),
&tablespace_used.indfree));

        GL_TRY( gStmt04, SQLExecDirect( gStmt04, (unsigned
char*)sSQL04,  SQL_NTS ) );
        fetchCount04 = 0;

        ret04 = SQLFetch( gStmt04 );
        if (ret04 == SQL_NO_DATA)
        {
            if ( fetchCount04 == 0 )
            {
                memcpy(&tablespace_used.used, "E", 1);
                memcpy(&tablespace_used.free, "E", 1);
            }
            else
            {
            }
        }
        GL_TRY( gStmt04, ret04 );
        fetchCount04++;
        GL_TRY( gStmt04, SQLCloseCursor( gStmt04 ) );

        printf("%2s %-27s %15s %15s %15s\n",
tablespace_alloc_size.id, tablespace_alloc_size.name,
tablespace_used.tot, tablespace_used.used, tablespace_used.free);

        fetchCount03++;
    }

    GL_TRY( gStmt03, SQLCloseCursor( gStmt03 ) );

    printf("\n");
    printf("-------------------------------------------------------
-----------------------------------------\n");
    printf("ID TABLE_NAME                                TOTAL(KB)
USED(KB)         FREE(KB)  ACTUAL USED(KB)\n");
    printf("-------------------------------------------------------
-----------------------------------------\n");
```

```
    /*
      SQL_sSQL01
      - table list all
    */

    GL_TRY( gStmt01, SQLExecute( gStmt01 ) );
    fetchCount01 = 0;

    while (1) {

        ret01 = SQLFetch( gStmt01 );

        if ( ret01 == SQL_NO_DATA ) {
            if ( fetchCount01 == 0 )
            {
                printf ("SQL01(%d) Table@L >x=@4O4Y.\n", __LINE__ );
                goto GLIESE_SUCCESS_FINISH;
            }
            else
            {
                break;
            }
        }
        GL_TRY( gStmt01, ret01 );
        fetchCount01++;

        sprintf(sSQL05,
                " SELECT
"
                "     TABLE_NAME,
"
                "     TABLESPACE_ID,
"
                "     TO_CHAR((USED1 + FREE1) / 1024,
'999,999,999,990') TOTAL,               "
                "     TO_CHAR(USED1 / 1024, '999,999,999,990') USED,
"
                "     TO_CHAR(FREE1 / 1024, '999,999,999,990')
FREE_SIZE,                      "
                "     TO_CHAR(ACT_USED / 1024, '999,999,999,990')
ACTUAL_USED                     "
                " FROM
(
    "
                "     SELECT (COUNT(*) * 8192) USED1
"
                "     FROM  D$MEMORY_SEGMENT_BITMAP('TABLE,%s.%s')
"
                "     WHERE 1=1
"
                "     AND   FREENESS <> 'FR'
"
                " ) A,
"
                "
(
        "
                "     SELECT (COUNT(*) * 8192) FREE1
"
                "     FROM  D$MEMORY_SEGMENT_BITMAP('TABLE,%s.%s')
"
                "     WHERE 1=1
"
                "     AND   FREENESS = 'FR'
"
                " ) B,
"
                "
(
        "
                "     SELECT TABLE_NAME, TABLESPACE_ID
"
                "     FROM DEFINITION_SCHEMA.TABLES
"
                "     WHERE 1=1
"
                "     AND   PHYSICAL_ID = %ld
"
                " ) C,
"
                "
(
        "
                "     SELECT (SUM(8192 - LOW_WATER_MARK)) ACT_USED
"
                "     FROM D$PAGE_SIGNPOST('TABLE,%s')
"
                "     WHERE 1=1
"
                " ) D
"
                , obj_table.schema
                , obj_table.name
                , obj_table.schema
                , obj_table.name
```

```
                , obj_table.physicalId
                , obj_table.name
            );

        i = 1;
        GL_TRY( gStmt05, SQLBindCol(gStmt05, i++, SQL_C_CHAR,
&table_used.name, sizeof(table_used.name), &table_used.indname));
        GL_TRY( gStmt05, SQLBindCol(gStmt05, i++, SQL_C_CHAR,
&table_used.tbsId, sizeof(table_used.tbsId), &table_used.indtbsId));
        GL_TRY( gStmt05, SQLBindCol(gStmt05, i++, SQL_C_CHAR,
&table_used.tot, sizeof(table_used.tot), &table_used.indtot));
        GL_TRY( gStmt05, SQLBindCol(gStmt05, i++, SQL_C_CHAR,
&table_used.used, sizeof(table_used.used), &table_used.indused));
        GL_TRY( gStmt05, SQLBindCol(gStmt05, i++, SQL_C_CHAR,
&table_used.free, sizeof(table_used.free), &table_used.indfree));
        GL_TRY( gStmt05, SQLBindCol(gStmt05, i++, SQL_C_CHAR,
&table_used.actual_used, sizeof(table_used.actual_used),
&table_used.indactualused));

        GL_TRY( gStmt05, SQLExecDirect( gStmt05, (unsigned
char*)sSQL05,  SQL_NTS ) );
        fetchCount05 = 0;

        ret05 = SQLFetch( gStmt05 );
        if (ret05 == SQL_NO_DATA)
        {
            if ( fetchCount05 == 0 )
            {
                memcpy(&table_used.tot, "E", 1);
                memcpy(&table_used.used, "E", 1);
                memcpy(&table_used.free, "E", 1);
            }
            else
            {
            }
        }
        GL_TRY( gStmt05, ret05 );
        fetchCount05++;
        GL_TRY( gStmt05, SQLCloseCursor( gStmt05 ) );

        printf("%2s %-27s %15s %15s %15s %15s\n", table_used.tbsId,
table_used.name, table_used.tot, table_used.used, table_used.free,
table_used.actual_used);

        /*
          SQL_sSQL02
          - index list all
        */
        memset ( &index_used, 0x00, sizeof(space_used_t));

        GL_TRY( gStmt02, SQLExecute( gStmt02 ) );
        fetchCount02 = 0;

        while (1)
        {
            ret02 = SQLFetch( gStmt02 );

            if ( ret02 == SQL_NO_DATA )
            {
                if ( fetchCount02 == 0 )
                {
                    printf ("   (has no indexes)\n");
                    break;
                }
                else
                {
                    break;
                }
            }
            GL_TRY( gStmt02, ret02 );
            fetchCount02++;

            sprintf(sSQL06,
                    " SELECT
"
                    "     INDEX_NAME,
"
                    "     TABLESPACE_ID,
"
                    "     TO_CHAR((USED1 + FREE1) / 1024,
'999,999,999,990') TOTAL,                "
                    "     TO_CHAR(USED1 / 1024, '999,999,999,990')
USED,                         "
                    "     TO_CHAR(FREE1 / 1024, '999,999,999,990')
FREE_SIZE,                    "
                    "     TO_CHAR(ACT_USED / 1024,
'999,999,999,990') ACTUAL_USED                "
                    " FROM
(
    "
                    "     SELECT (COUNT(*) * 8192) USED1
"
                    "     FROM
D$MEMORY_SEGMENT_BITMAP('INDEX,%s.%s')
"
                    "     WHERE 1=1
"
                    "     AND   FREENESS <> 'FR'
"
                    " ) A,
"
                    "
(
    "
                    "     SELECT (COUNT(*) * 8192) FREE1
"
                    "     FROM
D$MEMORY_SEGMENT_BITMAP('INDEX,%s.%s')
"
                    "     WHERE 1=1
"
                    "     AND   FREENESS = 'FR'
"
                    " ) B,
"
                    "
(
    "
                    "     SELECT INDEX_NAME, TABLESPACE_ID
"
                    "     FROM INDEXES
"
                    "     WHERE 1=1
"
                    "     AND   PHYSICAL_ID = %ld
"
                    " ) C,
"
                    "
(
    "
                    "     SELECT (SUM(8192 - LOW_WATER_MARK))
ACT_USED                      "
                    "     FROM D$PAGE_SIGNPOST('INDEX,%s')
"
                    "     WHERE 1=1
"
                    " ) D
"
                    , obj_index.schema
                    , obj_index.name
                    , obj_index.schema
                    , obj_index.name
                    , obj_index.physicalId
                    , obj_index.name
                );

            i = 1;
            GL_TRY( gStmt06, SQLBindCol(gStmt06, i++, SQL_C_CHAR,
&index_used.name, sizeof(index_used.name), &index_used.indname));
            GL_TRY( gStmt06, SQLBindCol(gStmt06, i++, SQL_C_CHAR,
&index_used.tbsId, sizeof(index_used.tbsId), &index_used.indtbsId));
            GL_TRY( gStmt06, SQLBindCol(gStmt06, i++, SQL_C_CHAR,
&index_used.tot, sizeof(index_used.tot), &index_used.indtot));
            GL_TRY( gStmt06, SQLBindCol(gStmt06, i++, SQL_C_CHAR,
&index_used.used, sizeof(index_used.used), &index_used.indused));
            GL_TRY( gStmt06, SQLBindCol(gStmt06, i++, SQL_C_CHAR,
&index_used.free, sizeof(index_used.free), &index_used.indfree));
            GL_TRY( gStmt06, SQLBindCol(gStmt06, i++, SQL_C_CHAR,
&index_used.actual_used, sizeof(index_used.actual_used),
&index_used.indactualused));

            GL_TRY( gStmt06, SQLExecDirect( gStmt06, (unsigned
char*)sSQL06,  SQL_NTS ) );
            fetchCount06 = 0;

            ret06 = SQLFetch( gStmt06 );
            if (ret06 == SQL_NO_DATA)
            {
                if ( fetchCount06 == 0 )
                {
                    memcpy(&index_used.tot, "E", 1);
                    memcpy(&index_used.used, "E", 1);
                    memcpy(&index_used.free, "E", 1);
                }
                else
                {
                }
            }
            GL_TRY( gStmt06, ret06 );
            fetchCount06++;
            GL_TRY( gStmt06, SQLCloseCursor( gStmt06 ) );

            printf("%2s %-27s %15s %15s %15s %15s\n",
index_used.tbsId, index_used.name, index_used.tot, index_used.used,
index_used.free, index_used.actual_used);

        } // while (1) index
        GL_TRY( gStmt02, SQLCloseCursor( gStmt02 ) );
```

```
        printf("-------------------------------------------------
-----------------------------------------------\n");
    } // while (1)
    GL_TRY( gStmt01, SQLCloseCursor( gStmt01 ) );

    GLIESE_SUCCESS;

    sState = 0;
    GL_TRY( gStmt01, SQLFreeHandle( SQL_HANDLE_STMT, gStmt01 ) );
    GL_TRY( gStmt02, SQLFreeHandle( SQL_HANDLE_STMT, gStmt02 ) );
    GL_TRY( gStmt03, SQLFreeHandle( SQL_HANDLE_STMT, gStmt03 ) );
    GL_TRY( gStmt04, SQLFreeHandle( SQL_HANDLE_STMT, gStmt04 ) );
    GL_TRY( gStmt05, SQLFreeHandle( SQL_HANDLE_STMT, gStmt05 ) );
    GL_TRY( gStmt06, SQLFreeHandle( SQL_HANDLE_STMT, gStmt06 ) );

    gStmt01 = NULL;
    gStmt02 = NULL;
    gStmt03 = NULL;
    gStmt04 = NULL;
    gStmt05 = NULL;
    gStmt06 = NULL;

    return (SQL_SUCCESS);

    GLIESE_FINISH;

    switch(sState)
    {
        case 1:

            (void) SQLFreeHandle( SQL_HANDLE_STMT, gStmt01 );
            (void) SQLFreeHandle( SQL_HANDLE_STMT, gStmt02 );
            (void) SQLFreeHandle( SQL_HANDLE_STMT, gStmt03 );
            (void) SQLFreeHandle( SQL_HANDLE_STMT, gStmt04 );
            (void) SQLFreeHandle( SQL_HANDLE_STMT, gStmt05 );
            (void) SQLFreeHandle( SQL_HANDLE_STMT, gStmt06 );

            gStmt01 = NULL;
            gStmt02 = NULL;
            gStmt03 = NULL;
            gStmt04 = NULL;
            gStmt05 = NULL;
            gStmt06 = NULL;
        default:
            break;
    }

    return (SQL_ERROR);
}


int main(int argc, char* argv[])
{

    int sState = 0;

    act_new.sa_sigaction = sig_handler;
    act_new.sa_flags = SA_SIGINFO;
    sigemptyset(&act_new.sa_mask);

    if (sigaction(SIGINT, &act_new, &act_old) < 0)
    {
        printf("sigaction SIGINT fail.\n");
        return 1;
    }
    if (sigaction(SIGQUIT, &act_new, &act_old) < 0)
    {
        printf("sigaction SIGQUIT fail.\n");
        return 1;
    }
    if (sigaction(SIGABRT, &act_new, &act_old) < 0)
    {
        printf("sigaction SIGABRT fail.\n");
        return 1;
    }
    if (sigaction(SIGSEGV, &act_new, &act_old) < 0)
    {
        printf("sigaction SIGSEGV fail.\n");
        return 1;
    }
    if (sigaction(SIGTERM, &act_new, &act_old) < 0)
    {
        printf("sigaction SIGTERM fail.\n");
        return 1;
    }
    if (sigaction(SIGUSR2, &act_new, &act_old) < 0)
    {
        printf("sigaction SIGUSR2 fail.\n");
        return 1;
    }

    printf("\n");

printf("************************************************************
************************************\n");
    printf("*
*\n");
    printf("*  FREE SPACE IN MEMORY REPORT OF GOLDILOCKS DATABASE
*\n");
    printf("*
*\n");
    printf("*  Copyright 2010-2014, SunjeSoft, Inc. All rights
reserved.                                    *\n");
    printf("*
*\n");

printf("************************************************************
************************************\n");

    GLIESE_SQL_TRY( SQLAllocHandle( SQL_HANDLE_ENV,
                                    NULL,
                                    &gEnv ) );
    sState = 1;

    GLIESE_SQL_TRY( SQLSetEnvAttr( gEnv,
                                   SQL_ATTR_ODBC_VERSION,
                                   (SQLPOINTER)SQL_OV_ODBC3,
                                   0 ) );

    GLIESE_SQL_TRY( SQLAllocHandle( SQL_HANDLE_DBC,
                                    gEnv,
                                    &gDbc ) );
    sState = 2;

    GLIESE_SQL_TRY( SQLConnect( gDbc,
                                (SQLCHAR*)"GOLDILOCKS",
                                SQL_NTS,
                                (SQLCHAR*)"test",
                                SQL_NTS,
                                (SQLCHAR*)"test",
                                SQL_NTS) );
    sState = 3;

    GLIESE_SQL_TRY( doJob() );
    printf("\n");

    sState = 2;
    GLIESE_SQL_TRY( SQLDisconnect( gDbc ) );

    sState = 1;
    GLIESE_SQL_TRY( SQLFreeHandle( SQL_HANDLE_DBC, gDbc ) );

    gDbc = NULL;

    sState = 0;
    GLIESE_SQL_TRY( SQLFreeHandle( SQL_HANDLE_ENV, gEnv ) );

    gEnv = NULL;

    return SQL_SUCCESS;

    GLIESE_FINISH;

    if( gDbc != NULL)
    {
        printError( SQL_HANDLE_DBC, gDbc );
    }
    if( gEnv != NULL)
    {
        printError( SQL_HANDLE_ENV, gEnv );
    }

    switch( sState )
    {
        case 3:
            (void)SQLDisconnect( gDbc );
        case 2:
            (void)SQLFreeHandle( SQL_HANDLE_DBC, gDbc );
        case 1:
            (void)SQLFreeHandle( SQL_HANDLE_ENV, gEnv );
        default:
            break;
    }

    printf("MAIN_CLOSE\n");

    return SQL_ERROR;
}
```

load.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/time.h>
```

```
#include <sys/timeb.h>

#include <goldilocks.h>

#include <tpc.h>
#include <support.h>
#include <spt_proc.h>

#define NNULL ((void *)0)

#define STMT_COUNT (11)

#define swap_int(a,b) {int tmp; tmp=a; a=b; b=tmp;}

static int nums[CUST_PER_DIST];
static int perm_count;

SQLHENV  env;
SQLHDBC  dbc;
SQLHSTMT stmt[STMT_COUNT];

/* Global SQL Variables */
SQLCHAR timestamp[21];
long    count_ware;
int     fd;
unsigned int seed;

int     particle_flg = 0; /* "1" means particle mode */
long    min_ware = 1;
long    max_ware;

/* ITEM */
SQLINTEGER i_id;
SQLINTEGER i_im_id;
SQLCHAR    i_name[25];
SQLREAL    i_price;
SQLCHAR    i_data[51];

/* WAREHOUSE */
SQLINTEGER w_id;
SQLCHAR    w_name[11];
SQLCHAR    w_street_1[21];
SQLCHAR    w_street_2[21];
SQLCHAR    w_city[21];
SQLCHAR    w_state[3];
SQLCHAR    w_zip[10];
SQLREAL    w_tax;
SQLREAL    w_ytd;

/* STOCK */
SQLINTEGER s_i_id;
SQLINTEGER s_w_id;
SQLINTEGER s_quantity;
SQLCHAR    s_dist_01[25];
SQLCHAR    s_dist_02[25];
SQLCHAR    s_dist_03[25];
SQLCHAR    s_dist_04[25];
SQLCHAR    s_dist_05[25];
SQLCHAR    s_dist_06[25];
SQLCHAR    s_dist_07[25];
SQLCHAR    s_dist_08[25];
SQLCHAR    s_dist_09[25];
SQLCHAR    s_dist_10[25];
SQLCHAR    s_data[51];

/* DISTRICT */
SQLINTEGER d_id;
SQLINTEGER d_w_id;
SQLCHAR    d_name[11];
SQLCHAR    d_street_1[21];
SQLCHAR    d_street_2[21];
SQLCHAR    d_city[21];
SQLCHAR    d_state[3];
SQLCHAR    d_zip[10];
SQLREAL    d_tax;
SQLREAL    d_ytd;
SQLINTEGER d_next_o_id;

/* CUSTOMER */
SQLINTEGER c_id;
SQLINTEGER c_d_id;
SQLINTEGER c_w_id;
SQLCHAR    c_first[17];
SQLCHAR    c_middle[3];
SQLCHAR    c_last[17];
SQLCHAR    c_street_1[21];
SQLCHAR    c_street_2[21];
SQLCHAR    c_city[21];
SQLCHAR    c_state[3];
SQLCHAR    c_zip[10];
SQLCHAR    c_phone[17];
SQLCHAR    c_credit[3];
SQLREAL    c_credit_lim;
SQLREAL    c_discount;
SQLREAL    c_balance;
SQLCHAR    c_data[501];

/* HISTORY */
SQLREAL    h_amount;
SQLCHAR    h_data[25];

/* ORDERS */
SQLINTEGER o_id;
SQLINTEGER o_c_id;
SQLINTEGER o_d_id;
SQLINTEGER o_w_id;
SQLINTEGER o_carrier_id;
SQLINTEGER o_ol_cnt;

/* ORDER-LINE */
SQLINTEGER ol_number;
SQLINTEGER ol_i_id;
SQLINTEGER ol_supply_w_id;
SQLINTEGER ol_quantity;
SQLREAL    ol_amount;
SQLCHAR    ol_dist_info[25];

/*
 * p. 54
 *
 * make a ``random a-string'': a string of random alphanumeric
 * characters of a random length of minimum x, maximum y, and
 * mean (y+x)/2
 */
int MakeAlphaString (unsigned int * seed, int x, int y, char str[])
{
    static char *alphanum = "0123456789"
        "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
        "abcdefghijklmnopqrstuvwxyz";
    int arrmax = 61;  /* index of last array element */
    register int i, len;

    len = RandomNumber(seed, x, y);

    for (i = 0; i < len; i++)
        str[i] = alphanum[RandomNumber(seed, 0, arrmax)];

    return len;
}

/*
 * like MakeAlphaString, only numeric characters only
 */
int MakeNumberString (unsigned int * seed, int x, int y, char str[])
{
    static char *numeric = "0123456789";
    int arrmax = 9;
    register int i, len;

    len = RandomNumber(seed, x, y);

    for (i = 0; i < len; i++)
        str[i] = numeric[RandomNumber(seed, 0, arrmax)];

    return len;
}

/*==========================================================================
=+
  | ROUTINE NAME
  |      MakeAddress()
  | DESCRIPTION
  |      Build an Address
  | ARGUMENTS

+============================================================*
/
void
MakeAddress(str1, str2, city, state, zip)
    char           *str1;
    char           *str2;
    char           *city;
    char           *state;
    char           *zip;
{
    str1[ MakeAlphaString(&seed, 10, 20, str1) ] = 0;       /*
Street 1 */
    str2[ MakeAlphaString(&seed, 10, 20, str2) ] = 0;       /*
Street 2 */
    city[ MakeAlphaString(&seed, 10, 20, city) ] = 0;       /* City
*/
    state[ MakeAlphaString(&seed, 2, 2, state) ] = 0;       /*
State */
    MakeNumberString(&seed, 4, 4, zip);                 /* Zip */
    zip[4] = '1';
    zip[5] = '1';
    zip[6] = '1';
    zip[7] = '1';
    zip[8] = '1';
    zip[9] = 0;
```

```
}

/*
 * permute the list of customer ids for the order table
 */
void InitPermutation (unsigned int * seed )
{
    int *cur;
    int i,j;

    perm_count = 0;

    /* initialize with consecutive values [1..ORD_PER_DIST] */
    for (i = 0, cur = nums; i < ORD_PER_DIST; i++, cur++) {
        *cur = i + 1;
    }

    /* now, shuffle */
    for (i = 0; i < ORD_PER_DIST-1; i++) {
        j = (int)RandomNumber(seed, i+1, ORD_PER_DIST-1);
        swap_int(nums[i], nums[j]);
    }
}

int GetPermutation (void)
{
    if ( perm_count >= ORD_PER_DIST ) {
        fprintf(stderr, "GetPermutation: past end of list!\n");
        abort();
    }
    return nums[perm_count++];
}

/*================================================================
=+
  | ROUTINE NAME
  |      Error()
  | DESCRIPTION
  |      Handles an error from a SQL call.
  | ARGUMENTS

+================================================================*
/
void
Error(aStmt)
    SQLHSTMT aStmt;
{
    SQLCHAR       sSQLState[6];
    SQLINTEGER    sNaiveError;
    SQLSMALLINT   sTextLength;
    SQLCHAR       sMessageText[SQL_MAX_MESSAGE_LENGTH];
    SQLRETURN     sReturn;

    if( aStmt != NULL )
    {
        sReturn = SQLGetDiagRec( SQL_HANDLE_STMT,
                                 aStmt,
                                 1,
                                 sSQLState,
                                 &sNaiveError,
                                 sMessageText,
                                 SQL_MAX_MESSAGE_LENGTH,
                                 &sTextLength );

        if( SQL_SUCCEEDED( sReturn ) )
        {

printf("\n============================================\n" );
            printf("SQL_DIAG_SQLSTATE     : %s\n", sSQLState );
            printf("SQL_DIAG_NATIVE       : %d\n", sNaiveError );
            printf("SQL_DIAG_MESSAGE_TEXT : %s\n", sMessageText );

printf("============================================\n" );
        }
    }

    sReturn = SQLGetDiagRec( SQL_HANDLE_DBC,
                             dbc,
                             1,
                             sSQLState,
                             &sNaiveError,
                             sMessageText,
                             SQL_MAX_MESSAGE_LENGTH,
                             &sTextLength );

    if( SQL_SUCCEEDED( sReturn ) )
    {

printf("\n============================================\n" );
        printf("SQL_DIAG_SQLSTATE     : %s\n", sSQLState );
        printf("SQL_DIAG_NATIVE       : %d\n", sNaiveError );
        printf("SQL_DIAG_MESSAGE_TEXT : %s\n", sMessageText );
        printf("============================================\n" );
    }
    exit(-1);
}

/*================================================================
=+
  | ROUTINE NAME
  |      LoadItems
  | DESCRIPTION
  |      Loads the Item table
  | ARGUMENTS
  |      none
  *
+================================================================*/
void
LoadItems()
{
    int idatasiz;
    int orig[MAXITEMS];
    int pos;
    int i;

    printf("Loading Item \n");

    for( i = 0; i < MAXITEMS; i++ )
    {
        orig[i] = 0;
    }

    for( i = 0; i < MAXITEMS / 10; i++ )
    {
        do
        {
            pos = RandomNumber(&seed, 0L, MAXITEMS - 1);
        } while (orig[pos]);
        orig[pos] = 1;
    }

    for( i_id = 1; i_id <= MAXITEMS; i_id++ )
    {
        /* Generate Item Data */
        i_im_id = RandomNumber(&seed, 1L, 10000L);

        i_name[ MakeAlphaString(&seed, 14, 24, (char*)i_name) ] = 0;

        i_price = ((int) RandomNumber(&seed, 100L, 10000L)) / 100.0;

        idatasiz = MakeAlphaString(&seed, 26, 50, (char*)i_data);
        i_data[idatasiz] = 0;

        if( orig[i_id - 1] )
        {
            pos = RandomNumber(&seed, 0L, idatasiz - 8);
            i_data[pos] = 'O';
            i_data[pos + 1] = 'R';
            i_data[pos + 2] = 'I';
            i_data[pos + 3] = 'G';
            i_data[pos + 4] = 'I';
            i_data[pos + 5] = 'N';
            i_data[pos + 6] = 'A';
            i_data[pos + 7] = 'L';
        }

        SQL_TRY( SQLExecute( stmt[0] ) );

        if( !(i_id % 100) )
        {
            printf(".");
            fflush(stdout);

            if( !(i_id % 5000) )
            {
                printf(" %d\n", i_id);
            }
        }
    }

    SQL_TRY( SQLEndTran( SQL_HANDLE_DBC,
                         dbc,
                         SQL_COMMIT ) );

    printf("Item Done. \n");

    return;

    SQL_FINISH;

    fprintf(stderr, "error at ITEM\n");

    Error( stmt );

    (void)SQLEndTran( SQL_HANDLE_DBC,
                      dbc,
                      SQL_ROLLBACK );
}
```

```
/*==============================================================
=+
  | ROUTINE NAME
  |      Stock
  | DESCRIPTION
  |      Loads the Stock table
  | ARGUMENTS
  |      w_id - warehouse id

+==============================================================*
/
int
Stock(aW_id)
    SQLINTEGER aW_id;
{
    int sdatasiz;
    int orig[MAXITEMS];
    int pos;
    int i;

    printf("Loading Stock Wid=%d\n", aW_id);

    s_w_id = aW_id;

    for( i = 0; i < MAXITEMS; i++ )
    {
        orig[i] = 0;
    }

    for( i = 0; i < MAXITEMS / 10; i++ )
    {
        do
        {
            pos = RandomNumber(&seed, 0L, MAXITEMS - 1);
        } while (orig[pos]);
        orig[pos] = 1;
    }

    for( s_i_id = 1; s_i_id <= MAXITEMS; s_i_id++ )
    {
        /* Generate Stock Data */
        s_quantity = RandomNumber(&seed, 10L, 100L);

        s_dist_01[ MakeAlphaString(&seed, 24, 24,
(char*)s_dist_01) ] = 0;
        s_dist_02[ MakeAlphaString(&seed, 24, 24,
(char*)s_dist_02) ] = 0;
        s_dist_03[ MakeAlphaString(&seed, 24, 24,
(char*)s_dist_03) ] = 0;
        s_dist_04[ MakeAlphaString(&seed, 24, 24,
(char*)s_dist_04) ] = 0;
        s_dist_05[ MakeAlphaString(&seed, 24, 24,
(char*)s_dist_05) ] = 0;
        s_dist_06[ MakeAlphaString(&seed, 24, 24,
(char*)s_dist_06) ] = 0;
        s_dist_07[ MakeAlphaString(&seed, 24, 24,
(char*)s_dist_07) ] = 0;
        s_dist_08[ MakeAlphaString(&seed, 24, 24,
(char*)s_dist_08) ] = 0;
        s_dist_09[ MakeAlphaString(&seed, 24, 24,
(char*)s_dist_09) ] = 0;
        s_dist_10[ MakeAlphaString(&seed, 24, 24,
(char*)s_dist_10) ] = 0;
        sdatasiz = MakeAlphaString(&seed, 26, 50, (char*)s_data);
        s_data[sdatasiz] = 0;

        if( orig[s_i_id - 1] )
        {
            pos = RandomNumber(&seed, 0L, sdatasiz - 8);

            s_data[pos] = 'O';
            s_data[pos + 1] = 'R';
            s_data[pos + 2] = 'I';
            s_data[pos + 3] = 'G';
            s_data[pos + 4] = 'I';
            s_data[pos + 5] = 'N';
            s_data[pos + 6] = 'A';
            s_data[pos + 7] = 'L';
        }

        SQL_TRY( SQLExecute( stmt[2] ) );

        if( !(s_i_id % 100) )
        {
            printf(".");
            fflush(stdout);

            if( !(s_i_id % 5000) )
            {
                printf(" %d\n", s_i_id);
            }
        }
    }

    printf(" Stock Done.\n");

    return 1;

    SQL_FINISH;

    fprintf(stderr, "error at Stock\n");

    Error( stmt[2] );

    return 0;
}

/*==============================================================
=+
  | ROUTINE NAME
  |      District
  | DESCRIPTION
  |      Loads the District table
  | ARGUMENTS
  |      w_id - warehouse id

+==============================================================*
/
int
District(aW_id)
    SQLINTEGER aW_id;
{
    printf("Loading District\n");

    d_w_id      = aW_id;
    d_ytd       = 30000.0;
    d_next_o_id = 3001;

    for( d_id = 1; d_id <= DIST_PER_WARE; d_id++ )
    {
        /* Generate District Data */
        d_name[ MakeAlphaString(&seed, 6L, 10L, (char*)d_name) ] =
0;
        MakeAddress(d_street_1, d_street_2, d_city, d_state, d_zip);

        d_tax = ((float) RandomNumber(&seed, 0L, 2000L)) / 10000.0;

        SQL_TRY( SQLExecute( stmt[3] ) );
    }

    return 1;

    SQL_FINISH;

    fprintf(stderr, "error at District\n");

    Error( stmt[3] );

    return 0;
}

/*==============================================================
=+
  | ROUTINE NAME
  |      LoadWare
  | DESCRIPTION
  |      Loads the Warehouse table
  |      Loads Stock, District as Warehouses are created
  | ARGUMENTS
  |      none

+==============================================================*
/
void
LoadWare()
{
    printf("Loading Warehouse \n");

    for( w_id = min_ware; w_id <= max_ware; w_id++ )
    {
        /* Generate Warehouse Data */

        w_name[ MakeAlphaString(&seed, 6, 10, (char*)w_name) ] = 0;

        MakeAddress(w_street_1, w_street_2, w_city, w_state, w_zip);

        w_tax = ((float) RandomNumber(&seed, 0L, 2000L)) / 10000.0;
        w_ytd = 300000.00;

        SQL_TRY( SQLExecute( stmt[1] ) );

        /** Make Rows associated with Warehouse **/
        SQL_TRY( Stock(w_id) == 1 );
        SQL_TRY( District(w_id) == 1 );

        SQL_TRY( SQLEndTran( SQL_HANDLE_DBC,
                             dbc,
                             SQL_COMMIT ) );
    }
```

```
    return;

    SQL_FINISH;

    fprintf(stderr, "error at WAREHOUSE\n");

    Error( stmt[1] );

    (void)SQLEndTran( SQL_HANDLE_DBC,
                      dbc,
                      SQL_ROLLBACK );
}

/*
================================================================+
   | ROUTINE NAME
   |      Customer
   | DESCRIPTION
   |      Loads Customer Table
   |      Also inserts corresponding history record
   | ARGUMENTS
   |      d_id - district id
   |      w_id - warehouse id

+================================================================
*/
void
Customer(aD_id, aW_id)
    SQLINTEGER aD_id;
    SQLINTEGER aW_id;
{
    SQLHSTMT sStmt;
    int      orig[CUST_PER_DIST];
    int      pos;
    int      i;

    printf("Loading Customer for DID=%d, WID=%d\n", aD_id, aW_id);

    for( i = 0; i < CUST_PER_DIST; i++ )
    {
        orig[i] = 0;
    }

    for( i = 0; i < CUST_PER_DIST / 10; i++ )
    {
        do
        {
            pos = RandomNumber(&seed, 0L, CUST_PER_DIST - 1);
        } while (orig[pos]);
        orig[pos] = 1;
    }

    for( c_id = 1; c_id <= CUST_PER_DIST; c_id++ )
    {
        /* Generate Customer Data */
        c_d_id = aD_id;
        c_w_id = aW_id;

        c_first[ MakeAlphaString(&seed, 8, 16, (char*)c_first) ] =
0;
        c_middle[0] = 'O';
        c_middle[1] = 'E';
        c_middle[2] = 0;

        if( c_id <= 1000 )
        {
            Lastname(c_id - 1, (char*)c_last);
        }
        else
        {
            Lastname(NURand(&seed, 255, 0, 999), (char*)c_last);
        }

        MakeAddress(c_street_1, c_street_2, c_city, c_state, c_zip);
        c_phone[ MakeNumberString(&seed, 16, 16, (char*)c_phone) ]
= 0;

        if( orig[c_id - 1] )
        {
            c_credit[0] = 'B';
        }
        else
        {
            c_credit[0] = 'G';
        }
        c_credit[1] = 'C';
        c_credit[2] = 0;

        c_credit_lim = 50000.0;
        c_discount = ((float) RandomNumber(&seed, 0L, 5000L)) /
10000.0;
        c_balance = -10.0;

        c_data[ MakeAlphaString(&seed, 300, 500, (char*)c_data) ] =
0;

        sStmt = stmt[4];
        SQL_TRY( SQLExecute( sStmt ) );

        h_amount = 10.0;

        h_data[ MakeAlphaString(&seed, 12, 24, (char*)h_data) ] = 0;

        sStmt = stmt[5];
        SQL_TRY( SQLExecute( sStmt ) );

        if( !(c_id % 100) )
        {
            printf(".");
            fflush(stdout);

            if( !(c_id % 1000) )
            {
                printf(" %d\n", c_id);
            }
        }
    }

    SQL_TRY( SQLEndTran( SQL_HANDLE_DBC,
                         dbc,
                         SQL_COMMIT ) );

    printf("Customer Done.\n");

    return;

    SQL_FINISH;

    fprintf(stderr, "error at Curstomer\n");

    Error( sStmt );
}

/*================================================================
=+
  | ROUTINE NAME
  |      LoadCust
  | DESCRIPTION
  |      Loads the Customer Table
  | ARGUMENTS
  |      none

+================================================================*
/
void
LoadCust()
{
    SQLINTEGER sW_id;
    SQLINTEGER sD_id;

    for( sW_id = min_ware; sW_id <= max_ware; sW_id++ )
    {
        for( sD_id = 1; sD_id <= DIST_PER_WARE; sD_id++ )
        {
            Customer(sD_id, sW_id);
        }
    }

    SQL_TRY( SQLEndTran( SQL_HANDLE_DBC,
                         dbc,
                         SQL_COMMIT ) );

    return;

    SQL_FINISH;

    fprintf(stderr, "error at LoadCust\n");

    Error( NULL );

    (void)SQLEndTran( SQL_HANDLE_DBC,
                      dbc,
                      SQL_ROLLBACK );
}

/*================================================================
=+
  | ROUTINE NAME
  |      Orders
  | DESCRIPTION
  |      Loads the Orders table
  |      Also loads the Order_Line table on the fly
  | ARGUMENTS
  |      d_id - district id
  |      w_id - warehouse id

+================================================================*
/
void
Orders(aD_id, aW_id)
```

```
    SQLINTEGER aD_id;
    SQLINTEGER aW_id;
{
    SQLHSTMT sStmt;

    printf("Loading Orders for D=%d, W=%d\n", aD_id, aW_id);

    o_d_id = aD_id;
    o_w_id = aW_id;

    InitPermutation(&seed);   /* initialize permutation of customer
numbers */

    for( o_id = 1; o_id <= ORD_PER_DIST; o_id++ )
    {
        /* Generate Order Data */
        o_c_id = GetPermutation();
        o_carrier_id = RandomNumber(&seed, 1L, 10L);
        o_ol_cnt = RandomNumber(&seed, 5L, 15L);

        if( o_id > 2100 )
        {
            /* the last 900 orders have not been delivered) */

            sStmt = stmt[6];
            SQL_TRY( SQLExecute( sStmt ) );

            sStmt = stmt[7];
            SQL_TRY( SQLExecute( sStmt ) );
        }
        else
        {
            sStmt = stmt[8];
            SQL_TRY( SQLExecute( sStmt ) );
        }

        for( ol_number = 1; ol_number <= o_ol_cnt; ol_number++ )
        {
            /* Generate Order Line Data */
            ol_i_id = RandomNumber(&seed, 1L, MAXITEMS);
            ol_supply_w_id = o_w_id;
            ol_quantity = 5;

            ol_dist_info[ MakeAlphaString(&seed, 24, 24,
(char*)ol_dist_info) ] = 0;

            if (o_id > 2100)
            {
                ol_amount = (float) (RandomNumber(&seed, 1L,
999999L)) / 100.0;

                sStmt = stmt[9];
                SQL_TRY( SQLExecute( sStmt ) );
            }
            else
            {
                ol_amount = 0.0;

                sStmt = stmt[10];
                SQL_TRY( SQLExecute( sStmt ) );
            }
        }

        if( !(o_id % 100) )
        {
            printf(".");
            fflush(stdout);

            if(!(o_id % 1000) )
            {
                printf(" %d\n", o_id);
            }
        }
    }

    SQL_TRY( SQLEndTran( SQL_HANDLE_DBC,
                         dbc,
                         SQL_COMMIT ) );

    printf("Orders Done.\n");

    return;

    SQL_FINISH;

    fprintf(stderr, "error at Orders\n");

    Error( sStmt );
}

/*================================================================
=+
  | ROUTINE NAME
  |      LoadOrd
  | DESCRIPTION
  |      Loads the Orders and Order_Line Tables
  | ARGUMENTS
  |      none

+================================================================*
/
void
LoadOrd()
{
    SQLINTEGER sW_id;
    SQLINTEGER sD_id;

    for( sW_id = min_ware; sW_id <= max_ware; sW_id++ )
    {
        for( sD_id = 1; sD_id <= DIST_PER_WARE; sD_id++ )
        {
            Orders(sD_id, sW_id);
        }
    }

    SQL_TRY( SQLEndTran( SQL_HANDLE_DBC,
                         dbc,
                         SQL_COMMIT ) );

    return;

    SQL_FINISH;

    fprintf(stderr, "error at LoadOrd\n");

    Error( NULL );

    (void)SQLEndTran( SQL_HANDLE_DBC,
                      dbc,
                      SQL_ROLLBACK );
}

/*================================================================
=+
  | ROUTINE NAME
  |      main()
  | ARGUMENTS
  |      Warehouses n [Debug] [Help]

+================================================================*
/
int
main(argc, argv)
    int             argc;
    char           *argv[];
{
    struct timeval  tv;
    int             sState = 0;
    int             i;

    /* initialize */
    count_ware = 0;

    printf("*************************************\n");
    printf("***  ###easy###  TPC-C Data Loader  ***\n");
    printf("*************************************\n");

    /* Parse args */
    if (argc != 4)
    {
        if (argc != 2)
        {
            fprintf( stderr,
                     "\n usage: tpcc_load [warehouse]\n"
                     "      OR\n"
                     "        tpcc_load [warehouse] [min_wh]
[max_wh]\n" );
            exit(1);
        }
    }
    else
    {
        particle_flg = 1;
    }

    if( (count_ware = atoi(argv[1])) <= 0 )
    {
        fprintf(stderr, "\n expecting positive number of
warehouses\n");
        exit(1);
    }

    if(particle_flg == 1 )
    {
        min_ware = atoi(argv[2]);
        max_ware = atoi(argv[3]);
    }
    else
    {
        min_ware = 1;
```

```
        max_ware = count_ware;
    }

    SetCValue( 4, 5, 6 );

    printf("<Parameters>\n");
    printf("  [warehouse]: %ld\n", count_ware);

    if(particle_flg==1)
    {
        printf("     [MIN WH]: %ld\n", min_ware);
        printf("     [MAX WH]: %ld\n", max_ware);
    }

    fd = open("/dev/urandom", O_RDONLY);
    if (fd == -1)
    {
        fd = open("/dev/random", O_RDONLY);
        if (fd == -1)
        {
            gettimeofday(&tv, NNULL);
            seed = (tv.tv_sec ^ tv.tv_usec) * tv.tv_sec *
tv.tv_usec ^ tv.tv_sec;
        }
        else
        {
            read(fd, &seed, sizeof(seed));
            close(fd);
        }
    }
    else
    {
        read(fd, &seed, sizeof(seed));
        close(fd);
    }

    /* Initialize timestamp (for date columns) */
    gettimestamp((char*)timestamp, sizeof(timestamp));

    if( !(SQL_SUCCEEDED( SQLAllocHandle( SQL_HANDLE_ENV,
                                         NULL,
                                         &env ) ) ) )
    {
        fprintf(stderr, "error at
SQLAllocHandle(SQL_HANDLE_ENV)\n");
        exit(1);
    }
    sState = 1;

    if( !(SQL_SUCCEEDED( SQLSetEnvAttr( env,
                                        SQL_ATTR_ODBC_VERSION,
                                        (SQLPOINTER)SQL_OV_ODBC3,
                                        0 ) ) ) )
    {
        fprintf(stderr, "error at
SQLSetEnvAttr(SQL_ATTR_ODBC_VERSION)\n");
        exit(1);
    }

    if( !(SQL_SUCCEEDED( SQLAllocHandle( SQL_HANDLE_DBC,
                                         env,
                                         &dbc ) ) ) )
    {
        fprintf(stderr, "error at
SQLAllocHandle(SQL_HANDLE_DBC)\n");
        exit(1);
    }

    sState = 2;

    SQL_TRY( SQLConnect( dbc,
                         (SQLCHAR*)"GOLDILOCKS",
                         SQL_NTS,
                         (SQLCHAR*)"test",
                         SQL_NTS,
                         (SQLCHAR*)"test",
                         SQL_NTS) );
    sState = 3;

    SQL_TRY( SQLSetConnectAttr( dbc,
                                SQL_ATTR_AUTOCOMMIT,
                                (SQLPOINTER)SQL_AUTOCOMMIT_OFF,
                                SQL_IS_UINTEGER ) );

    memset( stmt, 0x00, sizeof( SQLHSTMT ) * STMT_COUNT );

    for( i = 0; i < STMT_COUNT; i++ )
    {
        SQL_TRY( SQLAllocHandle( SQL_HANDLE_STMT,
                                 dbc,
                                 &stmt[i] ) );
    }

    SQL_TRY( SQLPrepare( stmt[0],
                         (SQLCHAR*)"INSERT INTO item
values(?,?,?,?,?)",
                         SQL_NTS ) );

    SQL_TRY( SQLBindParameter( stmt[0],
                               1,
                               SQL_PARAM_INPUT,
                               SQL_C_SLONG,
                               SQL_INTEGER,
                               0,
                               0,
                               &i_id,
                               0,
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[0],
                               2,
                               SQL_PARAM_INPUT,
                               SQL_C_SLONG,
                               SQL_INTEGER,
                               0,
                               0,
                               &i_im_id,
                               0,
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[0],
                               3,
                               SQL_PARAM_INPUT,
                               SQL_C_CHAR,
                               SQL_VARCHAR,
                               sizeof( i_name ) - 1,
                               0,
                               i_name,
                               sizeof( i_name ),
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[0],
                               4,
                               SQL_PARAM_INPUT,
                               SQL_C_FLOAT,
                               SQL_NUMERIC,
                               5,
                               2,
                               &i_price,
                               0,
                               NULL ) );


    SQL_TRY( SQLBindParameter( stmt[0],
                               5,
                               SQL_PARAM_INPUT,
                               SQL_C_CHAR,
                               SQL_VARCHAR,
                               sizeof( i_data ) - 1,
                               0,
                               i_data,
                               sizeof( i_data ),
                               NULL ) );



    SQL_TRY( SQLPrepare( stmt[1],
                         (SQLCHAR*)"INSERT INTO warehouse
values(?,?,?,?,?,?,?,?,?)",
                         SQL_NTS ) );

    SQL_TRY( SQLBindParameter( stmt[1],
                               1,
                               SQL_PARAM_INPUT,
                               SQL_C_SLONG,
                               SQL_INTEGER,
                               0,
                               0,
                               &w_id,
                               0,
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[1],
                               2,
                               SQL_PARAM_INPUT,
                               SQL_C_CHAR,
                               SQL_VARCHAR,
                               sizeof( w_name ) - 1,
                               0,
                               w_name,
                               sizeof( w_name ),
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[1],
                               3,
                               SQL_PARAM_INPUT,
                               SQL_C_CHAR,
                               SQL_VARCHAR,
                               sizeof( w_street_1 ) - 1,
```

```
                            0,
                            w_street_1,
                            sizeof( w_street_1 ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[1],
                            4,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_VARCHAR,
                            sizeof( w_street_2 ) - 1,
                            0,
                            w_street_2,
                            sizeof( w_street_2 ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[1],
                            5,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_VARCHAR,
                            sizeof( w_city ) - 1,
                            0,
                            w_city,
                            sizeof( w_city ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[1],
                            6,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( w_state ) - 1,
                            0,
                            w_state,
                            sizeof( w_state ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[1],
                            7,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( w_zip ) - 1,
                            0,
                            w_zip,
                            sizeof( w_zip ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[1],
                            8,
                            SQL_PARAM_INPUT,
                            SQL_C_FLOAT,
                            SQL_NUMERIC,
                            4,
                            4,
                            &w_tax,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[1],
                            9,
                            SQL_PARAM_INPUT,
                            SQL_C_FLOAT,
                            SQL_NUMERIC,
                            12,
                            2,
                            &w_ytd,
                            0,
                            NULL ) );



    SQL_TRY( SQLPrepare( stmt[2],
                        (SQLCHAR*)"INSERT INTO stock
values(?,?,?,?,?,?,?,?,?,?,?,?,0,0,0,?)",
                        SQL_NTS ) );

    SQL_TRY( SQLBindParameter( stmt[2],
                            1,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &s_i_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[2],
                            2,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &s_w_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[2],
                            3,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_NUMERIC,
                            4,
                            0,
                            &s_quantity,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[2],
                            4,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( s_dist_01 ) - 1,
                            0,
                            s_dist_01,
                            sizeof( s_dist_01 ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[2],
                            5,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( s_dist_02 ) - 1,
                            0,
                            s_dist_02,
                            sizeof( s_dist_02 ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[2],
                            6,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( s_dist_03 ) - 1,
                            0,
                            s_dist_03,
                            sizeof( s_dist_03 ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[2],
                            7,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( s_dist_04 ) - 1,
                            0,
                            s_dist_04,
                            sizeof( s_dist_04 ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[2],
                            8,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( s_dist_05 ) - 1,
                            0,
                            s_dist_05,
                            sizeof( s_dist_05 ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[2],
                            9,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( s_dist_06 ) - 1,
                            0,
                            s_dist_06,
                            sizeof( s_dist_06 ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[2],
                            10,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( s_dist_07 ) - 1,
                            0,
                            s_dist_07,
                            sizeof( s_dist_07 ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[2],
                            11,
```

```
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( s_dist_08 ) - 1,
                            0,
                            s_dist_08,
                            sizeof( s_dist_08 ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[2],
                            12,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( s_dist_09 ) - 1,
                            0,
                            s_dist_09,
                            sizeof( s_dist_09 ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[2],
                            13,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( s_dist_10 ) - 1,
                            0,
                            s_dist_10,
                            sizeof( s_dist_10 ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[2],
                            14,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( s_data ) - 1,
                            0,
                            s_data,
                            sizeof( s_data ),
                            NULL ) );



    SQL_TRY( SQLPrepare( stmt[3],
                        (SQLCHAR*)"INSERT INTO district
values(?,?,?,?,?,?,?,?,?,?,?)",
                        SQL_NTS ) );

    SQL_TRY( SQLBindParameter( stmt[3],
                            1,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &d_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[3],
                            2,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &d_w_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[3],
                            3,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_VARCHAR,
                            sizeof( d_name ) - 1,
                            0,
                            d_name,
                            sizeof( d_name ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[3],
                            4,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_VARCHAR,
                            sizeof( d_street_1 ) - 1,
                            0,
                            d_street_1,
                            sizeof( d_street_1 ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[3],
                            5,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_VARCHAR,
                            sizeof( d_street_2 ) - 1,
                            0,
                            d_street_2,
                            sizeof( d_street_2 ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[3],
                            6,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_VARCHAR,
                            sizeof( d_city ) - 1,
                            0,
                            d_city,
                            sizeof( d_city ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[3],
                            7,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( d_state ) - 1,
                            0,
                            d_state,
                            sizeof( d_state ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[3],
                            8,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( d_zip ) - 1,
                            0,
                            d_zip,
                            sizeof( d_zip ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[3],
                            9,
                            SQL_PARAM_INPUT,
                            SQL_C_FLOAT,
                            SQL_NUMERIC,
                            4,
                            4,
                            &d_tax,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[3],
                            10,
                            SQL_PARAM_INPUT,
                            SQL_C_FLOAT,
                            SQL_NUMERIC,
                            12,
                            2,
                            &d_ytd,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[3],
                            11,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &d_next_o_id,
                            0,
                            NULL ) );



    SQL_TRY( SQLPrepare( stmt[4],
                        (SQLCHAR*)"INSERT INTO customer
values(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?, 10.0, 1, 0,?)",
                        SQL_NTS ) );

    SQL_TRY( SQLBindParameter( stmt[4],
                            1,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &c_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[4],
```

```
                            2,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &c_d_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[4],
                            3,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &c_w_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[4],
                            4,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_VARCHAR,
                            sizeof( c_first ) - 1,
                            0,
                            c_first,
                            sizeof( c_first ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[4],
                            5,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( c_middle ) - 1,
                            0,
                            c_middle,
                            sizeof( c_middle ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[4],
                            6,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_VARCHAR,
                            sizeof( c_last ) - 1,
                            0,
                            c_last,
                            sizeof( c_last ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[4],
                            7,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_VARCHAR,
                            sizeof( c_street_1 ) - 1,
                            0,
                            c_street_1,
                            sizeof( c_street_1 ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[4],
                            8,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_VARCHAR,
                            sizeof( c_street_2 ) - 1,
                            0,
                            c_street_2,
                            sizeof( c_street_2 ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[4],
                            9,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_VARCHAR,
                            sizeof( c_city ) - 1,
                            0,
                            c_city,
                            sizeof( c_city ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[4],
                            10,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( c_state ) - 1,
                            0,
                            c_state,
                            sizeof( c_state ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[4],
                            11,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( c_zip ) - 1,
                            0,
                            c_zip,
                            sizeof( c_zip ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[4],
                            12,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( c_phone ) - 1,
                            0,
                            c_phone,
                            sizeof( c_phone ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[4],
                            13,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_VARCHAR,
                            sizeof( timestamp ) - 1,
                            0,
                            timestamp,
                            sizeof( timestamp ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[4],
                            14,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( c_credit ) - 1,
                            0,
                            c_credit,
                            sizeof( c_credit ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[4],
                            15,
                            SQL_PARAM_INPUT,
                            SQL_C_FLOAT,
                            SQL_NUMERIC,
                            12,
                            2,
                            &c_credit_lim,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[4],
                            16,
                            SQL_PARAM_INPUT,
                            SQL_C_FLOAT,
                            SQL_NUMERIC,
                            4,
                            4,
                            &c_discount,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[4],
                            17,
                            SQL_PARAM_INPUT,
                            SQL_C_FLOAT,
                            SQL_NUMERIC,
                            12,
                            2,
                            &c_balance,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[4],
                            18,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_VARCHAR,
                            sizeof( c_data ) - 1,
                            0,
                            c_data,
                            sizeof( c_data ),
                            NULL ) );


    SQL_TRY( SQLPrepare( stmt[5],
                         (SQLCHAR*)"INSERT INTO history
values(?,?,?,?,?,?,?,?)",
```

```
                        SQL_NTS ) );

    SQL_TRY( SQLBindParameter( stmt[5],
                               1,
                               SQL_PARAM_INPUT,
                               SQL_C_SLONG,
                               SQL_INTEGER,
                               0,
                               0,
                               &c_id,
                               0,
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[5],
                               2,
                               SQL_PARAM_INPUT,
                               SQL_C_SLONG,
                               SQL_INTEGER,
                               0,
                               0,
                               &c_d_id,
                               0,
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[5],
                               3,
                               SQL_PARAM_INPUT,
                               SQL_C_SLONG,
                               SQL_INTEGER,
                               0,
                               0,
                               &c_w_id,
                               0,
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[5],
                               4,
                               SQL_PARAM_INPUT,
                               SQL_C_SLONG,
                               SQL_INTEGER,
                               0,
                               0,
                               &c_d_id,
                               0,
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[5],
                               5,
                               SQL_PARAM_INPUT,
                               SQL_C_SLONG,
                               SQL_INTEGER,
                               0,
                               0,
                               &c_w_id,
                               0,
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[5],
                               6,
                               SQL_PARAM_INPUT,
                               SQL_C_CHAR,
                               SQL_VARCHAR,
                               sizeof( timestamp ) - 1,
                               0,
                               timestamp,
                               sizeof( timestamp ),
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[5],
                               7,
                               SQL_PARAM_INPUT,
                               SQL_C_FLOAT,
                               SQL_NUMERIC,
                               6,
                               2,
                               &h_amount,
                               0,
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[5],
                               8,
                               SQL_PARAM_INPUT,
                               SQL_C_CHAR,
                               SQL_VARCHAR,
                               sizeof( h_data ) - 1,
                               0,
                               h_data,
                               sizeof( h_data ),
                               NULL ) );



    SQL_TRY( SQLPrepare( stmt[6],
                         (SQLCHAR*)"INSERT INTO orders
values(?,?,?,?,?,NULL,?, 1)",
                        SQL_NTS ) );

    SQL_TRY( SQLBindParameter( stmt[6],
                               1,
                               SQL_PARAM_INPUT,
                               SQL_C_SLONG,
                               SQL_INTEGER,
                               0,
                               0,
                               &o_id,
                               0,
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[6],
                               2,
                               SQL_PARAM_INPUT,
                               SQL_C_SLONG,
                               SQL_INTEGER,
                               0,
                               0,
                               &o_d_id,
                               0,
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[6],
                               3,
                               SQL_PARAM_INPUT,
                               SQL_C_SLONG,
                               SQL_INTEGER,
                               0,
                               0,
                               &o_w_id,
                               0,
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[6],
                               4,
                               SQL_PARAM_INPUT,
                               SQL_C_SLONG,
                               SQL_INTEGER,
                               0,
                               0,
                               &o_c_id,
                               0,
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[6],
                               5,
                               SQL_PARAM_INPUT,
                               SQL_C_CHAR,
                               SQL_VARCHAR,
                               sizeof( timestamp ) - 1,
                               0,
                               timestamp,
                               sizeof( timestamp ),
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[6],
                               6,
                               SQL_PARAM_INPUT,
                               SQL_C_SLONG,
                               SQL_NUMERIC,
                               2,
                               0,
                               &o_ol_cnt,
                               0,
                               NULL ) );



    SQL_TRY( SQLPrepare( stmt[7],
                         (SQLCHAR*)"INSERT INTO new_order
values(?,?,?)",
                        SQL_NTS ) );

    SQL_TRY( SQLBindParameter( stmt[7],
                               1,
                               SQL_PARAM_INPUT,
                               SQL_C_SLONG,
                               SQL_INTEGER,
                               0,
                               0,
                               &o_id,
                               0,
                               NULL ) );

    SQL_TRY( SQLBindParameter( stmt[7],
                               2,
                               SQL_PARAM_INPUT,
                               SQL_C_SLONG,
                               SQL_INTEGER,
                               0,
                               0,
                               &o_d_id,
                               0,
```

```
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[7],
                            3,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &o_w_id,
                            0,
                            NULL ) );



    SQL_TRY( SQLPrepare( stmt[8],
                        (SQLCHAR*)"INSERT INTO orders
values(?,?,?,?,?,?,?, 1)",
                        SQL_NTS ) );

    SQL_TRY( SQLBindParameter( stmt[8],
                            1,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &o_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[8],
                            2,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &o_d_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[8],
                            3,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &o_w_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[8],
                            4,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &o_c_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[8],
                            5,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_VARCHAR,
                            sizeof( timestamp ) - 1,
                            0,
                            timestamp,
                            sizeof( timestamp ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[8],
                            6,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &o_carrier_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[8],
                            7,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_NUMERIC,
                            2,
                            0,
                            &o_ol_cnt,
                            0,
                            NULL ) );



    SQL_TRY( SQLPrepare( stmt[9],
                        (SQLCHAR*)"INSERT INTO order_line
values(?,?,?,?,?,?, NULL,?,?,?)",
                        SQL_NTS ) );

    SQL_TRY( SQLBindParameter( stmt[9],
                            1,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &o_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[9],
                            2,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &o_d_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[9],
                            3,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &o_w_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[9],
                            4,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &ol_number,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[9],
                            5,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &ol_i_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[9],
                            6,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &ol_supply_w_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[9],
                            7,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_NUMERIC,
                            2,
                            0,
                            &ol_quantity,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[9],
                            8,
                            SQL_PARAM_INPUT,
                            SQL_C_FLOAT,
                            SQL_NUMERIC,
                            6,
                            2,
                            &ol_amount,
                            0,
```

```
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[9],
                            9,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( ol_dist_info ) - 1,
                            0,
                            ol_dist_info,
                            sizeof( ol_dist_info ),
                            NULL ) );



    SQL_TRY( SQLPrepare( stmt[10],
                        (SQLCHAR*)"INSERT INTO order_line
values(?,?,?,?,?,?,?,?,?,?)",
                        SQL_NTS ) );

    SQL_TRY( SQLBindParameter( stmt[10],
                            1,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &o_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[10],
                            2,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &o_d_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[10],
                            3,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &o_w_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[10],
                            4,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &ol_number,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[10],
                            5,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &ol_i_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[10],
                            6,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_INTEGER,
                            0,
                            0,
                            &ol_supply_w_id,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[10],
                            7,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_VARCHAR,
                            sizeof( timestamp ),
                            0,
                            timestamp,
                            sizeof( timestamp ),
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[10],
                            8,
                            SQL_PARAM_INPUT,
                            SQL_C_SLONG,
                            SQL_NUMERIC,
                            2,
                            0,
                            &ol_quantity,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[10],
                            9,
                            SQL_PARAM_INPUT,
                            SQL_C_FLOAT,
                            SQL_NUMERIC,
                            6,
                            2,
                            &ol_amount,
                            0,
                            NULL ) );

    SQL_TRY( SQLBindParameter( stmt[10],
                            10,
                            SQL_PARAM_INPUT,
                            SQL_C_CHAR,
                            SQL_CHAR,
                            sizeof( ol_dist_info ) - 1,
                            0,
                            ol_dist_info,
                            sizeof( ol_dist_info ),
                            NULL ) );


    /* exec sql begin transaction; */

    printf("TPCC Data Load Started...\n");

    if( min_ware == 1 )
    {
        LoadItems ();
    }
    LoadWare();
    LoadCust();
    LoadOrd();

    SQL_TRY( SQLEndTran( SQL_HANDLE_DBC,
                         dbc,
                         SQL_COMMIT ) );

    for( i = 0; i < STMT_COUNT; i++ )
    {
        SQL_TRY( SQLFreeHandle( SQL_HANDLE_STMT,
                                stmt[i] ) );
        stmt[i] = NULL;
    }

    sState = 2;
    SQL_TRY( SQLDisconnect( dbc ) );

    sState = 1;
    SQL_TRY( SQLFreeHandle( SQL_HANDLE_DBC,
                            dbc ) );

    sState = 0;
    SQL_TRY( SQLFreeHandle( SQL_HANDLE_ENV,
                            env ) );

    printf("\n...DATA LOADING COMPLETED SUCCESSFULLY.\n");
    exit(0);

    SQL_FINISH;

    fprintf(stderr, "error at main\n");

    Error( NULL );

    (void)SQLEndTran( SQL_HANDLE_DBC,
                      dbc,
                      SQL_ROLLBACK );

    for( i = 0; i < STMT_COUNT; i++ )
    {
        if( stmt[i] != NULL )
        {
            SQL_TRY( SQLFreeHandle( SQL_HANDLE_STMT,
                                    stmt[i] ) );
        }
    }

    switch( sState )
    {
        case 3:
```

```
            (void)SQLDisconnect( dbc );
        case 2:
            (void)SQLFreeHandle( SQL_HANDLE_DBC,
                                 dbc );
        case 1:
            (void)SQLFreeHandle( SQL_HANDLE_ENV,
                                 env );
        default:
            break;
    }

    exit(0);
}
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/time.h>
#include <signal.h>
#include <pthread.h>
#include <fcntl.h>
#include <float.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <errno.h>
#include <assert.h>

#include <tpc.h>
#include <support.h>

extern int thread_main (ThreadArg* aArg);
extern void InitTransactionMix();
extern void FiniTransactionMix();

#define STRING_MAX_LEN 128

int   gWarehouseNum   = 0;
int   gStartWarehouse = 0;
int   gEndWarehouse   = 0;
int   gACIDTest       = 0;
pid_t gPID            = 0;
int   gTimeCount      = 0;

unsigned int gNewOrderCount = 0;

int gPrintInterval = 0;

int gIsAtivate  = 1;
int gIsCounting = 0;

float ShowTpmC()
{
    float sTpmC = 0.0;

    gTimeCount += gPrintInterval;

    sTpmC = (float)gNewOrderCount * 60.0
        / (float)((gTimeCount / gPrintInterval) * gPrintInterval);

    printf("| %5d | %12.3f |\n",
           gTimeCount,
           sTpmC );

    fflush(stdout);

    return sTpmC;
}

int main( int aArgc, char * aArgv[] )
{
    char sBaseUrl[128] = {0,};
    int  sWebPort;
    char * sPosIp;
    char * sPosPort;
    char * sPosPage;

    char sMasterIP[128];
    int  sPort;

    int  sUserNum;
    int  sTerminalNum;

    int  sRampupTime;
    int  sRampdownTime;
    int  sMeasureTime;

    char sStart[8];
    char sEnd[8];

    int  sSockFd;
    struct sockaddr_in sAddr;
    socklen_t sLen;
    int sSize;
    int sRemainSize;
    int sRecvSize;
    int sSendSize;

    ProtocolInfo   sInfo;
    ProtocolTpmC   sTpmC;

    pthread_t * sTerminalThreads = NULL;
    ThreadArg * sTerminalArgs    = NULL;

    int sWarehouseIdx;
    int sTerminalIdx;
    int sUserIdx;

    int sPrintCount = 0;
    int sError = 0;

    char   sTimestamp[21];

    int i;
    int c;

    printf("*********************************\n");
    printf("*** SUNJESOFT TPC-C Benchmark  ***\n");
    printf("*********************************\n");

    /* initialize */
    memset( sStart, 0x00, sizeof(sStart) );
    strcpy( sEnd, "END" );

    strcpy( sBaseUrl, "http://192.168.0.125/tpcc-0.1/tpcc" );

    strcpy( sMasterIP, "127.0.0.1" );
    sPort = 6821;

    gWarehouseNum   = 1;
    gStartWarehouse = 1;
    gEndWarehouse   = 1;
    gACIDTest       = 0;
    gPID            = 0;

    sTerminalNum   = 10;

#ifdef _DEBUG
    printf("test : sTerminalNum   = 1; \n");
    sTerminalNum   = 1;
#endif

    sRampupTime    = 60;
    sRampdownTime  = 60;

    sMeasureTime   = 60;
    gPrintInterval = 60;

    while( (c = getopt( aArgc, aArgv, "hm:p:n:x:u:")) != -1 )
    {
        switch( c )
        {
            case 'm':
                strcpy( sMasterIP, optarg );
                break;
            case 'p':
                sPort = atoi( optarg );
                break;
            case 'n':
                gStartWarehouse = atoi( optarg );
                break;
            case 'x':
                gEndWarehouse = atoi( optarg );
                break;
            case 'u':
                strcpy( sBaseUrl, optarg );
                break;
            case 'h':
                printf("Usage: %s -m master_ip -p port -n
min_warehouse_id -x max_warehouse_id -u WebUrl\n", aArgv[0]);
                exit(0);
            default:
                printf( "?? getopt returned character code
0%o ??\n", c );
        }
    }

    if( optind < aArgc )
    {
        printf( "non-option ARGV-elements: " );

        while( optind < aArgc )
        {
            printf( "%s ", aArgv[optind++] );
        }
```

```
    printf( "\n" );
}

sPosIp = strstr( sBaseUrl, "http://" );
if( sPosIp != NULL )
{
    sPosIp += 7;
}
else
{
    sPosIp = sBaseUrl;
    assert( (sPosIp[0] >= '0') && (sPosIp[0] <= '9') );
}

sPosPage = strchr( sPosIp, '/' );
if( sPosPage != NULL )
{
    *sPosPage = 0x00;
    sPosPage++;
}

sPosPort = strchr( sPosIp, ':' );
if( sPosPort != NULL )
{
    *sPosPort = 0x00;
    sWebPort = atoi( sPosPort + 1 );
}
else
{
    sWebPort = 80;
}
printf("IP: %s, %d, %s\n", sPosIp, sWebPort, sPosPage);

InitTransactionMix();

sSockFd = socket( AF_INET, SOCK_STREAM, 0 );
if( sSockFd == -1 )
{
    fprintf( stderr, "socket error\n" );
    exit(1);
}

sAddr.sin_family = AF_INET;
sAddr.sin_addr.s_addr = inet_addr(sMasterIP);
sAddr.sin_port = htons(sPort);

sLen = sizeof( sAddr );

if( connect( sSockFd, (struct sockaddr*)&sAddr, sLen ) == -1 )
{
    fprintf( stderr, "connect error\n" );
    exit(1);
}

sRemainSize = sizeof(ProtocolInfo);
sRecvSize   = 0;

while( sRecvSize < sRemainSize )
{
    do
    {
        sSize = read( sSockFd,
                      (void*)(((char*)&sInfo) + sRecvSize),
                      sRemainSize );
    } while( (sSize == -1) && (errno == EINTR) );

    if( sSize == -1 )
    {
        fprintf( stderr, "read node info error\n" );
        exit(1);
    }

    if( sSize == 0 )
    {
        fprintf( stderr, "read node info EOF error\n" );
        return 0;
    }

    sRemainSize -= sSize;
    sRecvSize   += sSize;
}

gWarehouseNum  = sInfo.mWarehouse;
sRampupTime    = sInfo.mRampup;
sRampdownTime  = sInfo.mRampdown;
sMeasureTime   = sInfo.mMeasure;
gPrintInterval = sInfo.mPrintInterval;
gACIDTest      = sInfo.mACIDTest;
gPID           = sInfo.mPID;

sRemainSize = strlen( "START" );
sRecvSize   = 0;

while( sRecvSize < sRemainSize )
{
        do
        {
            sSize = read( sSockFd,
                          (void*)(sStart + sRecvSize),
                          sRemainSize );
        } while( (sSize == -1) && (errno == EINTR) );

        if( sSize == -1 )
        {
            fprintf( stderr, "read start error\n" );
            exit(1);
        }

        if( sSize == 0 )
        {
            fprintf( stderr, "read start EOF error\n" );
            return 0;
        }

        sRemainSize -= sSize;
        sRecvSize   += sSize;
    }

    if( strcmp( sStart, "START" ) != 0 )
    {
        fprintf( stderr, "invalid start event\n" );
        exit(1);
    }

    gettimestamp( sTimestamp, sizeof(sTimestamp) );
    printf("\nTEST PERFORMED ON %s\n\n", sTimestamp);

    printf("<Parameters>\n");
    printf("               [warehouse]: %d\n", gWarehouseNum );
    printf("     [lower warehouse id]: %d\n", gStartWarehouse);
    printf("     [upper warehouse id]: %d\n", gEndWarehouse);
    printf("                  [rampup]: %d (sec.)\n", sRampupTime );
    printf("                [rampdown]: %d (sec.)\n",
sRampdownTime );
    printf("                 [measure]: %d (sec.)\n", sMeasureTime);


    /* set up threads */

    sUserNum = (gEndWarehouse - gStartWarehouse + 1 ) *
sTerminalNum;

    sTerminalThreads = malloc( sizeof(pthread_t) * sUserNum  );
    if( sTerminalThreads == NULL )
    {
        fprintf( stderr, "error at malloc(pthread_t)\n" );
        exit(1);
    }

    sTerminalArgs = malloc( sizeof(ThreadArg) * sUserNum );
    if( sTerminalArgs == NULL )
    {
        fprintf( stderr, "error at malloc(ThreadArg)\n" );
        exit(1);
    }

    memset( sTerminalArgs, 0x00, sizeof(ThreadArg) * sUserNum );

    for( sWarehouseIdx = gStartWarehouse; sWarehouseIdx <=
gEndWarehouse; sWarehouseIdx++ )
    {
        for( sTerminalIdx = 0; sTerminalIdx < sTerminalNum;
sTerminalIdx++ )
        {
            sUserIdx = ( sWarehouseIdx - gStartWarehouse ) *
sTerminalNum + sTerminalIdx;

            sTerminalArgs[sUserIdx].mW_ID       = sWarehouseIdx;
            sTerminalArgs[sUserIdx].mTerminalID = sTerminalIdx + 1;

            sTerminalArgs[sUserIdx].mWebIP = sPosIp;
            sTerminalArgs[sUserIdx].mWebPort = sWebPort;
            sTerminalArgs[sUserIdx].mWebPage = sPosPage;

            pthread_create( &sTerminalThreads[sUserIdx],
                            NULL,
                            (void*)thread_main,
                            (void*)&sTerminalArgs[sUserIdx] );
        }
    }

    printf( "\nRAMP-UP TIME.(%d sec.)\n", sRampupTime );
    fflush( stdout );

    sleep( sRampupTime );

    printf( "\nMEASURING START.\n\n" );
    fflush(stdout);

    printf( "========================\n" );
```

```
    printf( "|  SEC  |     tpmC     |\n" );
    printf( "========================\n" );
    fflush(stdout);

    gIsCounting = 1;

    sPrintCount = sMeasureTime / gPrintInterval;

    for( i = 0; i < sPrintCount; i++ )
    {
        sleep( gPrintInterval );

        if( gIsAtivate == 0 )
        {
            sError = 1;
            break;
        }

        sTpmC.mIdx  = i;
        sTpmC.mTpmC = ShowTpmC();

        sRemainSize = sizeof(ProtocolTpmC);
        sSendSize   = 0;

        while( sSendSize < sRemainSize )
        {
            do
            {
                sSize = write( sSockFd,
                               (void*)(((char*)&sTpmC) + sSendSize),
                               sRemainSize );
            } while( (sSize == -1) && (errno == EINTR) );

            if( sSize  == -1 )
            {
                fprintf( stderr, "write tpmc error\n" );
                exit(1);
            }

            sRemainSize -= sSize;
            sSendSize   += sSize;
        }
    }

    gIsCounting = 0;

    printf( "========================\n" );

    printf( "\nRAMP-DOWN TIME.(%d sec.)\n", sRampdownTime );
    fflush( stdout );

    sleep( sRampdownTime );

    printf( "\nSTOPPING THREADS" );
    fflush( stdout );

    gIsAtivate = 0;

    /* wait threads' ending and thread_main (ThreadArg* aArg);close
connections*/
    for( i = 0; i < sUserNum; i++ )
    {
        pthread_join( sTerminalThreads[i], NULL );
    }

    sRemainSize = strlen(sEnd);
    sSendSize   = 0;

    if( sError == 0 )
    {
        while( sSendSize < sRemainSize )
        {
            do
            {
                sSize = write( sSockFd,
                               (void*)(sEnd + sSendSize),
                               sRemainSize );
            } while( (sSize == -1) && (errno == EINTR) );

            if( sSize == -1 )
            {
                fprintf( stderr, "send end event error\n" );
                return 0;
            }

            sRemainSize -= sSize;
            sSendSize   += sSize;
        }
    }

    FiniTransactionMix();

    if( close( sSockFd ) == -1 )
    {
        fprintf( stderr, "clsoe socket error\n" );
        exit(1);
    }

    free( sTerminalThreads );
    free( sTerminalArgs );

    exit(0);
}
```

Makefile

```
#
# "make all" to build necessary executables.
#

CC              = gcc

LIBSERVER            = -L$(GOLDILOCKS_HOME)/lib -lgoldilocksa -ldl
-lpthread -lrt -lm
LIBCLIENT            = -L$(GOLDILOCKS_HOME)/lib -lpthread -lrt -lm

INC             = -I. -I../include -I$(GOLDILOCKS_HOME)/include

CFLAGS          = -Wall -O3
LINK_FLAG       = -rdynamic

.SUFFIXES:
.SUFFIXES: .o .c

.c.o:
         $(CC) $(CFLAGS) $(INC) $(DEFS) -c $*.c

all: free_space tpcc_load tpcc_result tpcc_master tpcc_client

free_space : free_space.o
         $(CC) $(LINK_FLAG) free_space.o $(LIBSERVER) -
o ../bin/free_space

tpcc_load : load.o support.o
         $(CC) $(LINK_FLAG) load.o support.o $(LIBSERVER) -
o ../bin/tpcc_load

tpcc_client : main.o support.o driver.o
         $(CC) $(LINK_FLAG) main.o support.o driver.o $(LIBCLIENT)
-o ../bin/tpcc_client

tpcc_result : result.o support.o
         $(CC) $(LINK_FLAG) result.o support.o $(LIBCLIENT) -
o ../bin/tpcc_result

tpcc_master : master.o support.o
         $(CC) $(LINK_FLAG) master.o support.o $(LIBCLIENT) -
o ../bin/tpcc_master

clean :
         rm -f
*.o ../bin/free_space ../bin/tpcc_load ../bin/tpcc_master ../bin/tp
cc_client ../bin/tpcc_result
```

master.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <pthread.h>
#include <errno.h>
#include <arpa/inet.h>

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

#include <tpc.h>
#include <support.h>

pthread_mutex_t gTestMutex = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t  gTestCondition = PTHREAD_COND_INITIALIZER;

int gNodeNum = 0;

int   gWarehouseNum  = 0;
int   gRampupTime    = 0;
int   gRampdownTime  = 0;
int   gMeasureTime   = 0;
int   gPrintInterval = 0;
int   gACIDTest      = 0;
int   gError         = 0;
pid_t gPID           = 0;

unsigned int gReadyNodeNum = 0;
```

```
float ** gTpmC = NULL;

int                * gNodeFd   = NULL;
struct sockaddr_in * gNodeAddr = NULL;

int thread_main( void * aArg )
{
    int sNodeID;
    int sSockFd;
    int sSize;
    int sRemainSize;
    int sRecvSize;
    int sSendSize;

    char           sStart[8];
    char           sEnd[8];

    ProtocolInfo   sInfo;
    ProtocolTpmC   sTpmC;

    int sPrintCount = 0;

    int i;

    strcpy( sStart, "START" );
    memset( sEnd, 0x00, sizeof(sEnd) );

    sNodeID = (int)(long)aArg;
    sSockFd = gNodeFd[sNodeID];

    sInfo.mWarehouse     = gWarehouseNum;
    sInfo.mRampup        = gRampupTime;
    sInfo.mRampdown      = gRampdownTime;
    sInfo.mMeasure       = gMeasureTime;
    sInfo.mPrintInterval = gPrintInterval;
    sInfo.mACIDTest      = gACIDTest;
    sInfo.mPID           = gPID;

    sRemainSize = sizeof(ProtocolInfo);
    sSendSize   = 0;

    while( sSendSize < sRemainSize )
    {
        do
        {
            sSize = write( sSockFd,
                           (void*)(((char*)&sInfo) + sSendSize),
                           sRemainSize );
        } while( (sSize == -1) && (errno == EINTR) );

        if( sSize == -1 )
        {
            fprintf( stderr, "send %d node info error\n", sNodeID );
            gError = 1;
            return 0;
        }

        sRemainSize -= sSize;
        sSendSize   += sSize;
    }

    __sync_fetch_and_add( &gReadyNodeNum, 1 );

    pthread_cond_wait( &gTestCondition, &gTestMutex );
    pthread_mutex_unlock( &gTestMutex );

    sRemainSize = strlen(sStart);
    sSendSize   = 0;

    while( sSendSize < sRemainSize )
    {
        do
        {
            sSize = write( sSockFd,
                           (void*)(sStart + sSendSize),
                           sRemainSize );
        } while( (sSize == -1) && (errno == EINTR) );

        if( sSize == -1 )
        {
            fprintf( stderr, "send %d start event error\n",
sNodeID );
            gError = 1;
            return 0;
        }

        sRemainSize -= sSize;
        sSendSize   += sSize;
    }

    sPrintCount = gMeasureTime / gPrintInterval;

    for( i = 0; i < sPrintCount; i++ )
    {
        if( gError == 1 )
        {
            break;
        }

        sRemainSize = sizeof(ProtocolTpmC);
        sRecvSize   = 0;

        while( sRecvSize < sRemainSize )
        {
            do
            {
                sSize = read( sSockFd,
                              (void*)(((char*)&sTpmC) + sRecvSize),
                              sRemainSize );
            } while( (sSize == -1) && (errno == EINTR) );

            if( sSize == -1 )
            {
                fprintf( stderr, "read %d node tpmC error\n",
sNodeID );
                gError = 1;
                return 0;
            }

            if( sSize == 0 )
            {
                fprintf( stderr, "read %d node tpmC EOF error\n",
sNodeID );
                gError = 1;
                return 0;
            }

            sRemainSize -= sSize;
            sRecvSize   += sSize;
        }

        gTpmC[sTpmC.mIdx][sNodeID] = sTpmC.mTpmC;
    }

    if( gError == 0 )
    {
        sRemainSize = strlen( "END" );
        sRecvSize   = 0;

        while( sRecvSize < sRemainSize )
        {
            do
            {
                sSize = read( sSockFd,
                              (void*)(sEnd + sRecvSize),
                              sRemainSize );
            } while( (sSize == -1) && (errno == EINTR) );

            if( sSize == -1 )
            {
                fprintf( stderr, "read %d end error\n", sNodeID );
                gError = 1;
                return 0;
            }

            if( sSize == 0 )
            {
                fprintf( stderr, "read %d end EOF error\n",
sNodeID );
                gError = 1;
                return 0;
            }

            sRemainSize -= sSize;
            sRecvSize   += sSize;
        }

        if( strcmp( sEnd, "END" ) != 0 )
        {
            fprintf( stderr, "invalid %d end event\n", sNodeID );
            gError = 1;
            return 0;
        }
    }

    return 0;
}

int main( int aArgc, char * aArgv[] )
{
    int sPort;

    int sSockFd;
    struct sockaddr_in sAddr;
    socklen_t sAddrLen = 0;
    int sValue = 1;

    int sPrintCount = 0;
    float sTpmC;
```

```
    pthread_t * sNodeThreads = NULL;

    char   sTimestamp[21];
    FILE * sInfoFile = NULL;

    struct timespec sTestStartTime;
    struct timespec sTestEndTime;
    struct timespec sMeasureStartTime;
    struct timespec sMeasureEndTime;

    int c;
    int i;
    int j;

    printf("*********************************\n");
    printf("*** SUNJESOFT TPC-C Benchmark  ***\n");
    printf("*********************************\n");

    /* initialize */
    gNodeNum       = 1;
    sPort          = 6821;

    gWarehouseNum  = 1;
    gRampupTime    = 60;
    gRampdownTime  = 60;
    gMeasureTime   = 60;
    gPrintInterval = 60;
    gPID           = getpid();

    while( (c = getopt( aArgc, aArgv, "n:p:w:r:o:l:i:dh")) != -1 )
    {
        switch( c )
        {
            case 'n':
                gNodeNum = atoi( optarg );
                break;
            case 'p':
                sPort = atoi( optarg );
                break;
            case 'w':
                gWarehouseNum = atoi( optarg );
                break;
            case 'r':
                gRampupTime = atoi( optarg );
                break;
            case 'o':
                gRampdownTime = atoi( optarg );
                break;
            case 'l':
                gMeasureTime = atoi( optarg );
                break;
            case 'i':
                gPrintInterval = atoi( optarg );
                break;
            case 'd':
                gACIDTest = 1;
                break;
            case 'h':
                printf("Usage: %s -n nodes -p port -w warehouses -r
rampup_time -o rampdown_time -l running_time -i report_interval -
d\n", aArgv[0]);
                exit(0);
            default:
                printf( "?? getopt returned character code
0%o ??\n", c );
        }
    }

    if( optind < aArgc )
    {
        printf( "non-option ARGV-elements: " );

        while( optind < aArgc )
        {
            printf( "%s ", aArgv[optind++] );
        }
        printf( "\n" );
    }

    sPrintCount = gMeasureTime / gPrintInterval;

    gTpmC = (float**)malloc( sizeof(float*) * sPrintCount );
    if( gTpmC == NULL )
    {
        fprintf( stderr, "malloc tpmc array error\n" );
        exit(1);
    }

    for( i = 0; i < sPrintCount; i++ )
    {
        gTpmC[i] = (float*)malloc( sizeof(float) * gNodeNum );

        if( gTpmC[i] == NULL )
        {
            fprintf( stderr, "malloc tpmc array error\n" );
            exit(1);
        }

        for( j = 0; j < gNodeNum; j++ )
        {
            gTpmC[i][j] = -1.0;
        }
    }

    gNodeFd = (int*)malloc( sizeof(int) * gNodeNum );
    if( gNodeFd == NULL )
    {
        fprintf( stderr, "malloc node fd error\n" );
        exit(1);
    }

    sNodeThreads = (pthread_t*)malloc( sizeof(pthread_t) *
gNodeNum );
    if( sNodeThreads == NULL )
    {
        fprintf( stderr, "malloc pthread_t error\n" );
        exit(1);
    }

    gNodeAddr = (struct sockaddr_in*)malloc( sizeof(struct
sockaddr_in) * gNodeNum );
    if( gNodeAddr == NULL )
    {
        fprintf( stderr, "malloc node addr error\n" );
        exit(1);
    }

    sSockFd = socket( AF_INET, SOCK_STREAM, 0);
    if( sSockFd < 0 )
    {
        fprintf( stderr, "socket error\n" );
        exit(1);
    }

    if( setsockopt( sSockFd, SOL_SOCKET, SO_REUSEADDR, &sValue,
sizeof(sValue) )== -1 )
    {
        fprintf( stderr, "setsockopt error\n" );
        exit(1);
    }

    bzero( &sAddr, sizeof(sAddr) );
    sAddr.sin_family = AF_INET;
    sAddr.sin_addr.s_addr = htonl(INADDR_ANY);
    sAddr.sin_port = htons(sPort);

    if( bind( sSockFd, (struct sockaddr*)&sAddr, sizeof(sAddr) ) ==
-1 )
    {
        fprintf( stderr, "bind error\n" );
        exit(1);
    }

    if( listen( sSockFd, gNodeNum) == -1 )
    {
        fprintf( stderr, "listen error\n" );
        exit(1);
    }

    sAddrLen = sizeof(struct sockaddr_in);

    printf( "WATING " );
    fflush( stdout );

    for( i = 0; i < gNodeNum; i++ )
    {
        gNodeFd[i] = accept( sSockFd, (struct sockaddr
*)&gNodeAddr[i], &sAddrLen );
        if( gNodeFd[i] == -1 )
        {
            fprintf( stderr, "accept %d error\n", i );
            exit(1);
        }

        pthread_create( &sNodeThreads[i], NULL, (void*)thread_main,
(void*)(long)i );
    }

    while( 1 )
    {
        if( gReadyNodeNum == gNodeNum )
        {
            break;
        }

        printf( "." );
        fflush( stdout );

        sleep( 1 );
```

```
}

printf( "\n" );

pthread_cond_broadcast( &gTestCondition );

clock_gettime( CLOCK_REALTIME, &sTestStartTime );

gettimestamp( sTimestamp, sizeof(sTimestamp) );
printf("\nTEST PERFORMED ON %s\n\n", sTimestamp);

printf("<Parameters>\n");
printf("     [warehouse]: %d\n", gWarehouseNum );
printf("     [rampup]   : %d (sec.)\n", gRampupTime );
printf("     [rampdown] : %d (sec.)\n", gRampdownTime );
printf("     [measure]  : %d (sec.)\n", gMeasureTime);

printf( "\nRAMP-UP TIME.(%d sec.)\n", gRampupTime );
fflush( stdout );

sleep( gRampupTime );

printf( "\nMEASURING START.\n\n" );
fflush( stdout );

clock_gettime( CLOCK_REALTIME, &sMeasureStartTime );

sleep( gPrintInterval );

printf( "========================\n" );
printf( "|  SEC  |     tpmC     |\n" );
printf( "========================\n" );
fflush( stdout );

for( i = 0; i < sPrintCount; i++ )
{
    if( gError == 1 )
    {
        clock_gettime( CLOCK_REALTIME, &sMeasureEndTime );
        break;
    }

    sleep( gPrintInterval );

    sTpmC = 0.0;
    for( j = 0; j < gNodeNum; j++ )
    {
        if( gTpmC[i][j] == -1.0 )
        {
            printf( "tpmC is not set yet!\n" );
            fflush( stdout );

            gTpmC[i][j] = 0;
        }

        sTpmC += gTpmC[i][j];
    }

    printf("| %5d | %12.3f |\n",
           (i + 1) * gPrintInterval,
           sTpmC );
    fflush( stdout );

    if( i == sPrintCount - 2 )
    {
        clock_gettime( CLOCK_REALTIME, &sMeasureEndTime );
    }
}

printf( "========================\n" );

printf( "\nRAMP-DOWN TIME.(%d sec.)\n", gRampdownTime );
fflush( stdout );

sleep( gRampdownTime );

for( i = 0; i < gNodeNum; i++ )
{
    pthread_join( sNodeThreads[i], NULL );
}

clock_gettime( CLOCK_REALTIME, &sTestEndTime );

sInfoFile = fopen( "../log/info.log", "w" );
if( sInfoFile == NULL )
{
    fprintf( stderr, "fail fopen(../log/info.log)\n" );
    exit(1);
}

fprintf( sInfoFile, "%d %d %ld %ld %ld %ld %ld %ld %ld %ld",
         gACIDTest,
         gWarehouseNum,
         sTestStartTime.tv_sec,
         sTestStartTime.tv_nsec,
         sMeasureStartTime.tv_sec,
         sMeasureStartTime.tv_nsec,
         sMeasureEndTime.tv_sec,
         sMeasureEndTime.tv_nsec,
         sTestEndTime.tv_sec,
         sTestEndTime.tv_nsec );

    fclose( sInfoFile );
    sInfoFile = NULL;

    free( gNodeAddr );
    gNodeAddr = NULL;

    for( i = 0; i < gNodeNum; i++ )
    {
        if( close( gNodeFd[i] ) == -1 )
        {
            fprintf( stderr, "close %d error\n", i );
            return 0;
        }
    }

    free( gNodeFd );
    gNodeFd = NULL;

    for( i = 0; i < sPrintCount; i++ )
    {
        free( gTpmC[i] );
        gTpmC[i] = NULL;
    }

    free( gTpmC );
    gTpmC = NULL;

    free( sNodeThreads );
    sNodeThreads = NULL;

    pthread_mutex_destroy( &gTestMutex );
    pthread_cond_destroy( &gTestCondition );

    return 0;
}
```

result.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <math.h>
#include <sys/time.h>
#include <signal.h>
#include <pthread.h>
#include <fcntl.h>
#include <float.h>

#include <tpc.h>
#include <support.h>

#define REPORT_HTML_PART_1
\
    "<html>\n"
\
    "  <head>\n"
\
    "    <script type=\"text/javascript\"
src=\"https://www.gstatic.com/charts/loader.js\"></script>\n" \
    "    <script type=\"text/javascript\">\n"
\
    "      google.charts.load('current',
{'packages':['corechart']});\n"                              \
    "      google.charts.setOnLoadCallback(drawChart);\n"
\
    "\n"
\
    "      function drawChart() {\n"
\
    "        var data = new google.visualization.DataTable();\n"

#define REPORT_HTML_PART_1_RT
\
    "        data.addColumn('string', 'Response Time(sec.)');\n"
\
    "        data.addColumn('number', 'Number of Transactions');\n"
\
    "        data.addColumn({type:'string', role:'annotation'});\n"
\
    "        data.addColumn({type:'string', role:'annotation'});\n"
\
    "        data.addColumn({type:'string', role:'annotation'});\n"
\
    "        data.addRows([\n"
```

```
#define REPORT_HTML_PART_1_THINK
\
    "        data.addColumn('string', 'Think Time(sec.)');\n"
\
    "        data.addColumn('number', 'Think Time Frequency');\n"
\
    "        data.addColumn({type:'string', role:'annotation'});\n"
\
    "        data.addColumn({type:'string', role:'annotation'});\n"
\
    "        data.addRows([\n"

#define REPORT_HTML_PART_1_THROUGHPUT                           \
    "        data.addColumn('string', 'Elapsed Time(sec.)');\n" \
    "        data.addColumn('number', 'MQTh');\n"               \
    "        data.addRows([\n"

#define REPORT_HTML_PART_2                      \
    "\n"                                        \
    "        ]);\n"                             \
    "\n"                                        \
    "        var options = {\n"                 \
    "          hAxis: { title : '%s%s' },\n"    \
    "          vAxis: { title : '%s' },\n"      \
    "          title: '"

#define REPORT_HTML_PART_3
\
    "',\n"
\
    "          annotations : { style: 'line' },\n"
\
    "          legend: { position: 'none' }\n"
\
    "        };\n"
\
    "\n"
\
    "        var chart = new
google.visualization.LineChart(document.getElementById('draw_chart'
));\n" \
    "\n"
\
    "        chart.draw(data, options);\n"
\
    "      }\n"
\
    "    </script>\n"
\
    "  </head>\n"
\
    "  <body>\n"
\
    "    <div id=\"draw_chart\" style=\"width: 1200px; height:
900px\"></div>\n"                       \
    "  </body>\n"
\
    "</html>\n"

#define RT_MAXREC      2
#define RT_REC_PER_SEC 1000000

#define USEC_PER_SEC   1000000

#define THINK_MAXREC      50

#define THROUGHPUT_INTERVAL_SIZE 30

#define MENU_IDX 5
#define DELIVERY_DEFERRED_IDX 6

typedef struct TransactionData
{
    long   mCount;
    long   mSuccess;
    long   mFailures;
    double mKeySum;
    double mKeyMin;
    double mKeyMax;
    double mKeyAvg;
    double mRTSum;
    double mRTMin;
    double mRTMax;
    double mRTAvg;
    double mRT90th;
    int    mRT90thLine;
    double mThinkSum;
    double mThinkMin;
    double mThinkMax;
    double mThinkAvg;
} TransactionData;

char * gTitle[] =
{
    "New Order Response Time Frequency Distribution",
    "Payment Response Time Distribution",
    "Order Status Response Time Distribution",
    "Interactive Delivery Response Time Distribution",
    "StockLevel Response Time Distribution",
    "Menu Response Time Distribution",
    "Deferred Delivery Response Time Distribution"
};

char * gHAixs[] =
{
    "Response Time(sec.)",
    "Think Time(sec.)",
    "Elapsed Time(sec.)"
};

char * gVAixs[] =
{
    "Number of Transactions",
    "Think Time Frequency",
    "MQTh"
};

int gRT90th[] = { 5, 5, 5, 5, 20, 0, 80 };

int gRTGraph[7][RT_MAXREC * RT_REC_PER_SEC];
int gThinkGraph[THINK_MAXREC];
int * gThroghtputGraph = NULL;

long gRollbackCount            = 0;
long gCommitCount              = 0;
long gTotalItemsCount          = 0;
long gRemoteItemCount          = 0;
long gRemotePaymentCount       = 0;
long gPaymentByC_LASTCount     = 0;
long gOrderStatusByC_LASTCount = 0;

int sortACIDItem( const void * aP1, const void * aP2 )
{
    ACIDItem * sP1 = (ACIDItem*)aP1;
    ACIDItem * sP2 = (ACIDItem*)aP2;

    double sTimestamp1;
    double sTimestamp2;

    sTimestamp1 = (double)(sP1->mTransactionEndTimeSec * 1000000.0)
+
                           (sP1->mTransactionEndTimeNano / 1000.0 );
    sTimestamp2 = (double)(sP2->mTransactionEndTimeSec * 1000000.0)
+
                           (sP2->mTransactionEndTimeNano / 1000.0 );

    if( sTimestamp1 < sTimestamp2 )
    {
        return 1;
    }
    else if( sTimestamp1 > sTimestamp2 )
    {
        return -1;
    }
    else
    {
        return 0;
    }
}

int main( int aArgc, char * aArgv[] )
{
    int sWarehouseNum;
    int sTerminalNum;
    int sACIDTest;

    int sWarehouseIdx;
    int sTerminalIdx;

    FILE * sGraphFile = NULL;
    FILE * sInfoFile = NULL;
    FILE * sLogFile = NULL;
    FILE * sACIDFile = NULL;
    char   sFileName[128];

    FILE * sRTFile[5] = { NULL, };

    struct timespec sTestStartTime;
    struct timespec sTestEndTime;
    struct timespec sMeasureStartTime;
    struct timespec sMeasureEndTime;

    long sTestStartTimeSec;
    long sTestStartTimeNano;
    long sTestEndTimeSec;
    long sTestEndTimeNano;
    long sTestIntervalCount;

    long sMeasureStartTimeSec;
    long sMeasureStartTimeNano;
```

```
    long sMeasureEndTimeSec;
    long sMeasureEndTimeNano;
    int  sWarmupInterval = 0;
    int  sMeasureInterval = 0;

    int    sTransactionType;
    double sMenuTime;
    double sKeyingTime;
    double sResponseTime;
    double sThinkTime;
    int    sIsRollbacked;
    int    sCount1;
    int    sCount2;
    int    sACID1;
    int    sACID2;
    long   sTransactionEndTimeSec;
    long   sTransactionEndTimeNano;
    struct timespec sTransactionStartTime;
    struct timespec sTransactionEndTime;

    long sTransactionCount = 0;
    long sNewOrderCount = 0;
    TransactionData sData[7];

    int sDeferred = 0;
    int sIdx;
    int sStartYY;
    int sStartMM;
    int sStartDD;
    int sStartHH;
    int sStartMI;
    int sStartSS;
    int sStartMili;

    int sWID;
    int sCarrierID;

    int sD1;
    int sD2;
    int sD3;
    int sD4;
    int sD5;
    int sD6;
    int sD7;
    int sD8;
    int sD9;
    int sD10;

    int sEndYY;
    int sEndMM;
    int sEndDD;
    int sEndHH;
    int sEndMI;
    int sEndSS;
    int sEndMili;

    ACIDItem   sACIDItem;
    ACIDItem * sACIDItemArray = NULL;
    int        sACIDItemCount = 0;

    int sLoop;
    char sTestStartTimeString[32];
    int sLine;
    int sPrintMax;
    long sTemp;
    int i;
    int j;
    int c;

    struct tm sTmTestStart;
    struct tm sTmTestEnd;
    struct tm sTmDeliveryStart;
    struct tm sTmDeliveryEnd;

#ifdef DEBUG
    struct tm sTmTransEnd;
#endif

    while( (c = getopt( aArgc, aArgv, "w:m:dh")) != -1 )
    {
        switch( c )
        {
            case 'w':
                sWarmupInterval = atoi( optarg );
                break;
            case 'm':
                sMeasureInterval = atoi( optarg );
                break;
            case 'd':
                sDeferred = 1;
                break;
            case 'h':
                printf("Usage: %s -w warmup_interval -m
measure_interval\n", aArgv[0]);
                exit(0);
            default:
                printf( "?? getopt returned character code
0%o ??\n", c );
        }
    }

    if( optind < aArgc )
    {
        printf( "non-option ARGV-elements: " );

        while( optind < aArgc )
        {
            printf( "%s ", aArgv[optind++] );
        }
        printf( "\n" );
    }


    for( i = 0; i < 7; i++ )
    {
        for( j = 0; j < RT_MAXREC * RT_REC_PER_SEC; j++ )
        {
            gRTGraph[i][j] = 0;
        }
    }

    for( i = 0; i < THINK_MAXREC; i++ )
    {
        gThinkGraph[i] = 0;
    }

    memset( (void*)sData, 0x00, sizeof(TransactionData) * 7 );

    for( i = 0; i < 7; i++ )
    {
        sData[i].mKeyMin   = DBL_MAX;
        sData[i].mRTMin    = DBL_MAX;
        sData[i].mThinkMin = DBL_MAX;
    }

    sInfoFile = fopen( "../log/info.log", "r" );
    if( sInfoFile == NULL )
    {
        fprintf( stderr, "fail fopen(../log/info.log)\n" );
        exit(1);
    }

    sTerminalNum = 10;

    fscanf( sInfoFile, "%d %d %ld %ld %ld %ld %ld %ld %ld %ld",
            &sACIDTest,
            &sWarehouseNum,
            &sTestStartTimeSec,
            &sTestStartTimeNano,
            &sMeasureStartTimeSec,
            &sMeasureStartTimeNano,
            &sMeasureEndTimeSec,
            &sMeasureEndTimeNano,
            &sTestEndTimeSec,
            &sTestEndTimeNano );

    fclose( sInfoFile );

    if( sWarmupInterval != 0 )
    {
        sMeasureStartTimeSec  = sTestStartTimeSec + sWarmupInterval;
        sMeasureStartTimeNano = sTestStartTimeNano;
    }

    if( sMeasureInterval != 0 )
    {
        sMeasureEndTimeSec  = sMeasureStartTimeSec +
sMeasureInterval;
        sMeasureEndTimeNano = sMeasureStartTimeNano;
    }

    if( sACIDTest == 1 )
    {
        sACIDFile = fopen( "../log/ACID.log", "w+" );
        if( sACIDFile == NULL )
        {
            fprintf( stderr, "fail fopen(../log/ACID.log)\n" );
            exit(1);
        }
    }

    sTestStartTime.tv_sec     = (time_t)sTestStartTimeSec;
    sTestStartTime.tv_nsec    = sTestStartTimeNano;
    sMeasureStartTime.tv_sec  = (time_t)sMeasureStartTimeSec;
    sMeasureStartTime.tv_nsec = sMeasureStartTimeNano;
    sMeasureEndTime.tv_sec    = (time_t)sMeasureEndTimeSec;
    sMeasureEndTime.tv_nsec   = sMeasureEndTimeNano;
    sTestEndTime.tv_sec       = (time_t)sTestEndTimeSec;
    sTestEndTime.tv_nsec      = sTestEndTimeNano;
```

```
    localtime_r((time_t *)&sTestStartTime.tv_sec, &sTmTestStart);
    localtime_r((time_t *)&sTestEndTime.tv_sec, &sTmTestEnd);

    sTestIntervalCount = (int)(GetTimespecDiff( &sTestStartTime,
&sTestEndTime ) /
                               (double)USEC_PER_SEC /
                               (double)THROUGHPUT_INTERVAL_SIZE );
    sTestIntervalCount++;

    gThroghtputGraph = (int*)malloc( sizeof(int) *
sTestIntervalCount );
    if( gThroghtputGraph == NULL )
    {
        fprintf( stderr, "fail malloc throughput graph\n" );
        exit(1);
    }

    memset( gThroghtputGraph, 0x00, sizeof(int) *
sTestIntervalCount );

    for( i = 0; i < 5; i++ )
    {
        sprintf( sFileName, "../log/DEBUG_RT_%d.txt", i );

        sRTFile[i] = fopen( sFileName, "w" );
        if( sRTFile[i] == NULL )
        {
            fprintf( stderr, "fail fopen(%s)\n", sFileName );
            exit(1);
        }
    }

    if( sDeferred == 1 )
    {
        /* Delivery (Deferred) */
        for( sIdx = 1; sIdx < 4; sIdx++ )
        {
            sprintf( sFileName, "../log/run1-dylog%d.txt", sIdx );

            sLogFile = fopen( sFileName, "r" );
            if( sLogFile == NULL )
            {
                fprintf( stderr, "fail fopen(%s)\n", sFileName );
                exit(1);
            }

            while( feof( sLogFile ) == 0 )
            {
                fscanf( sLogFile,
                        "%d-%d-
%d %d:%d:%d.%d %d %d %d|1|%d|2|%d|3|%d|4|%d|5|%d|6|%d|7|%d|8|%d|9|%
d|10| %d-%d-%d %d:%d:%d.%d\n",
                        &sStartYY,
                        &sStartMM,
                        &sStartDD,
                        &sStartHH,
                        &sStartMI,
                        &sStartSS,
                        &sStartMili,
                        &sWID,
                        &sCarrierID,
                        &sD1,
                        &sD2,
                        &sD3,
                        &sD4,
                        &sD5,
                        &sD6,
                        &sD7,
                        &sD8,
                        &sD9,
                        &sD10,
                        &sEndYY,
                        &sEndMM,
                        &sEndDD,
                        &sEndHH,
                        &sEndMI,
                        &sEndSS,
                        &sEndMili );

                sTmDeliveryStart.tm_year = sStartYY - 1900;
                sTmDeliveryStart.tm_mon  = sStartMM - 1;
                sTmDeliveryStart.tm_mday = sStartDD;
                sTmDeliveryStart.tm_hour = sStartHH;
                sTmDeliveryStart.tm_min  = sStartMI;
                sTmDeliveryStart.tm_sec  = sStartSS;

                sTransactionStartTime.tv_sec =
mktime( &sTmDeliveryStart );
                sTransactionStartTime.tv_nsec = (long)sStartMili *
1000000;

                sTmDeliveryEnd.tm_year = sEndYY - 1900;
                sTmDeliveryEnd.tm_mon  = sEndMM - 1;
                sTmDeliveryEnd.tm_mday = sEndDD;
                sTmDeliveryEnd.tm_hour = sEndHH;
                sTmDeliveryEnd.tm_min  = sEndMI;
                sTmDeliveryEnd.tm_sec  = sEndSS;

                sTransactionEndTime.tv_sec =
mktime( &sTmDeliveryEnd );
                sTransactionEndTime.tv_nsec = (long)sEndMili *
1000000;

                if( GetTimespecDiff( &sMeasureStartTime,
                                     &sTransactionStartTime ) >= 0 )
                {
                    if( GetTimespecDiff( &sMeasureEndTime,
                                         &sTransactionEndTime ) <=
0 )
                    {
                        sData[DELIVERY_DEFERRED_IDX].mCount++;

                        if( (sD1 == -1) || (sD2 == -1) || (sD3 == -
1) || (sD4 == -1) || (sD5 == -1) ||
                            (sD6 == -1) || (sD7 == -1) || (sD8 == -
1) || (sD9 == -1) || (sD10 == -1) )
                        {

sData[DELIVERY_DEFERRED_IDX].mFailures++;
                            continue;
                        }
                        else
                        {
                            sData[DELIVERY_DEFERRED_IDX].mSuccess++;
                        }

                        sResponseTime =
GetTimespecDiff( &sTransactionStartTime, &sTransactionEndTime);

                        sData[DELIVERY_DEFERRED_IDX].mRTSum +=
sResponseTime;

                        if( sData[DELIVERY_DEFERRED_IDX].mRTMax <
sResponseTime )
                        {
                            sData[DELIVERY_DEFERRED_IDX].mRTMax =
sResponseTime;
                        }

                        if( sData[DELIVERY_DEFERRED_IDX].mRTMin >
sResponseTime )
                        {
                            sData[DELIVERY_DEFERRED_IDX].mRTMin =
sResponseTime;
                        }

                        sLine = (int)(sResponseTime *
RT_REC_PER_SEC / (double)USEC_PER_SEC );

                        if( sLine >= (RT_MAXREC * RT_REC_PER_SEC) )
                        {
                            sLine = (RT_MAXREC * RT_REC_PER_SEC) -
1;
                        }

                        gRTGraph[DELIVERY_DEFERRED_IDX][sLine]++;
                    }
                }
            }
        }
    }

    for( sWarehouseIdx = 1; sWarehouseIdx <= sWarehouseNum;
sWarehouseIdx++ )
    {
        for( sTerminalIdx = 1; sTerminalIdx <= sTerminalNum;
sTerminalIdx++ )
        {
            printf( "." );

            sprintf( sFileName, "../log/N_%05d_%02d.log",
sWarehouseIdx, sTerminalIdx );

            sLogFile = fopen( sFileName, "r" );
            if( sLogFile == NULL )
            {
                fprintf( stderr, "fail fopen(%s)\n", sFileName );
                exit(1);
            }

            sTransactionStartTime = sTestStartTime;

            while( feof( sLogFile ) == 0 )
            {
                fscanf( sLogFile,

"\n*%d> %lf %lf %lf %d %d %d %d %d %lf %ld %ld ",
                        &sTransactionType,
                        &sMenuTime,
                        &sKeyingTime,
```

```
                        &sResponseTime,
                        &sIsRollbacked,
                        &sCount1,
                        &sCount2,
                        &sACID1,
                        &sACID2,
                        &sThinkTime,
                        &sTransactionEndTimeSec,
                        &sTransactionEndTimeNano );

                sTransactionEndTime.tv_sec  =
sTransactionEndTimeSec;
                sTransactionEndTime.tv_nsec =
sTransactionEndTimeNano;

#ifdef DEBUG
                localtime_r( (time_t *)&sTransactionEndTime.tv_sec,
&sTmTransEnd );
                fprintf( sRTFile[sTransactionType],
                         "          [new
Date(%04d, %02d, %02d, %02d, %02d, %02d, %03ld), %.6lf],\n",
                         1900 + sTmTransEnd.tm_year,
                         sTmTransEnd.tm_mon,
                         sTmTransEnd.tm_mday,
                         sTmTransEnd.tm_hour,
                         sTmTransEnd.tm_min,
                         sTmTransEnd.tm_sec,
                         sTransactionEndTimeNano / 1000,
                         sResponseTime / (double)USEC_PER_SEC );
#endif

                if( (sTransactionType == 0) && (sCount2 != -1) )
                {
                    if( sIsRollbacked == 0 )
                    {
                        sNewOrderCount++;
                    }

                    sLine = (int)(GetTimespecDiff( &sTestStartTime,
&sTransactionEndTime ) /
                                  (double)USEC_PER_SEC /
                                  (double)THROUGHPUT_INTERVAL_SIZE);

                    gThroghtputGraph[sLine]++;

                    if( sACIDTest == 1 )
                    {
                        /* last 300 second */
                        if( sTransactionEndTimeSec >=
sMeasureEndTimeSec - 300 )
                        {
                            sACIDItem.mTransactionEndTimeSec  =
sTransactionEndTimeSec;
                            sACIDItem.mTransactionEndTimeNano =
sTransactionEndTimeNano;
                            sACIDItem.mW_ID                   =
(unsigned short)sWarehouseIdx;
                            sACIDItem.mD_ID                   =
(unsigned char)sACID1;
                            sACIDItem.mRollbacked             =
(char)sIsRollbacked;
                            sACIDItem.mO_ID                   =
sACID2;

                            fwrite( &sACIDItem, sizeof(ACIDItem), 1,
sACIDFile );

                            sACIDItemCount++;
                        }
                    }
                }

                if( GetTimespecDiff( &sMeasureStartTime,
                                     &sTransactionStartTime ) >= 0 )
                {
                    if( GetTimespecDiff( &sMeasureEndTime,
                                         &sTransactionEndTime ) <=
0 )
                    {
                        sData[MENU_IDX].mCount++;

                        if( sCount1 == -1 )
                        {
                            sData[MENU_IDX].mFailures++;
                            goto SKIP_LABEL;
                        }
                        else
                        {
                            sData[MENU_IDX].mSuccess++;
                        }

                        sData[MENU_IDX].mRTSum += sMenuTime;

                        if( sData[MENU_IDX].mRTMax < sMenuTime )
                        {
                            sData[MENU_IDX].mRTMax = sMenuTime;
                        }

                        if( sData[MENU_IDX].mRTMin > sMenuTime )
                        {
                            sData[MENU_IDX].mRTMin = sMenuTime;
                        }

                        sLine = (int)(sMenuTime * RT_REC_PER_SEC /
(double)USEC_PER_SEC );

                        if( sLine >= (RT_MAXREC * RT_REC_PER_SEC) )
                        {
                            sLine = (RT_MAXREC * RT_REC_PER_SEC) -
1;
                        }

                        gRTGraph[MENU_IDX][sLine]++;

                        sTransactionCount++;
                        sData[sTransactionType].mCount++;

                        if( sCount2 == -1 )
                        {
                            sData[sTransactionType].mFailures++;
                            goto SKIP_LABEL;
                        }
                        else
                        {
                            sData[sTransactionType].mSuccess++;
                        }

                        sData[sTransactionType].mKeySum +=
sKeyingTime;

                        if( sData[sTransactionType].mKeyMax <
sKeyingTime )
                        {
                            sData[sTransactionType].mKeyMax =
sKeyingTime;
                        }

                        if( sData[sTransactionType].mKeyMin >
sKeyingTime )
                        {
                            sData[sTransactionType].mKeyMin =
sKeyingTime;
                        }

                        sData[sTransactionType].mRTSum +=
sResponseTime;

                        if( sData[sTransactionType].mRTMax <
sResponseTime )
                        {
                            sData[sTransactionType].mRTMax =
sResponseTime;
                        }

                        if( sData[sTransactionType].mRTMin >
sResponseTime )
                        {
                            sData[sTransactionType].mRTMin =
sResponseTime;
                        }

                        sLine = (int)(sResponseTime *
RT_REC_PER_SEC / (double)USEC_PER_SEC );

                        if( sLine >= (RT_MAXREC * RT_REC_PER_SEC) )
                        {
                            sLine = (RT_MAXREC * RT_REC_PER_SEC) -
1;
                        }

                        gRTGraph[sTransactionType][sLine]++;

                        sData[sTransactionType].mThinkSum +=
sThinkTime;

                        if( sData[sTransactionType].mThinkMax <
sThinkTime )
                        {
                            sData[sTransactionType].mThinkMax =
sThinkTime;
                        }

                        if( sData[sTransactionType].mThinkMin >
sThinkTime )
                        {
                            sData[sTransactionType].mThinkMin =
sThinkTime;
                        }

                        switch( sTransactionType )
```

```
                        {
                            case TRANSACTION_TYPE_NEW_ORDER :
                                sLine = (int)(sThinkTime /
(double)USEC_PER_SEC );

                                if( sLine < THINK_MAXREC )
                                {
                                    gThinkGraph[sLine]++;
                                }

                                if( sIsRollbacked == 1 )
                                {
                                    gRollbackCount++;
                                }
                                else
                                {
                                    gCommitCount++;
                                }

                                gTotalItemsCount += sCount1;
                                gRemoteItemCount += sCount2;
                                break;
                            case TRANSACTION_TYPE_PAYMENT :
                                gRemotePaymentCount   += sCount1;
                                gPaymentByC_LASTCount += sCount2;
                                break;
                            case TRANSACTION_TYPE_ORDER_STATUS :
                                gOrderStatusByC_LASTCount +=
sCount1;
                                break;
                            case TRANSACTION_TYPE_DELIVERY :
                                break;
                            case TRANSACTION_TYPE_STOCK_LEVEL :
                                break;
                        }
                    }
                }

            SKIP_LABEL :

                sTransactionStartTime = sTransactionEndTime;
            }

            fclose( sLogFile );
            sLogFile = NULL;
        }
    }

    if( sDeferred == 0 )
    {
        sLoop = 6;
    }
    else
    {
        sLoop = 7;
    }

    for( i = 0; i < sLoop; i++ )
    {
        sData[i].mKeyAvg   = sData[i].mKeySum /
(double)sData[i].mSuccess;
        sData[i].mRTAvg    = sData[i].mRTSum /
(double)sData[i].mSuccess;
        sData[i].mThinkAvg = sData[i].mThinkSum /
(double)sData[i].mSuccess;

        sTemp = 0;

        for( j = 0; j < (RT_MAXREC * RT_REC_PER_SEC); j++ )
        {
            sTemp += gRTGraph[i][j];

            if( sTemp >= ( sData[i].mSuccess * 90 / 100 ) )
            {
                sData[i].mRT90thLine = j;
                sData[i].mRT90th = (double)(j + 1) /
(double)RT_REC_PER_SEC;
                break;
            }
        }

        sData[i].mKeyAvg   /= (double)USEC_PER_SEC;
        sData[i].mKeyMin   /= (double)USEC_PER_SEC;
        sData[i].mKeyMax   /= (double)USEC_PER_SEC;
        sData[i].mRTAvg    /= (double)USEC_PER_SEC;
        sData[i].mRTMin    /= (double)USEC_PER_SEC;
        sData[i].mRTMax    /= (double)USEC_PER_SEC;
        sData[i].mThinkAvg /= (double)USEC_PER_SEC;
        sData[i].mThinkMin /= (double)USEC_PER_SEC;
        sData[i].mThinkMax /= (double)USEC_PER_SEC;

        if( i < 5 )
        {
            fclose( sRTFile[i] );
            sRTFile[i] = NULL;
        }
    }

    printf( "\n< Result >\n\n");

    printf( "COMPUTED THROUGHPUT: %.3lf tpmC using %d
warehouses\n\n",
            (double)((double)sData[0].mSuccess * 60 /
                     ((double)GetTimespecDiff(&sMeasureStartTime,
&sMeasureEndTime) /
                      (double)USEC_PER_SEC)),
            sWarehouseNum);

    printf( "MENU:\n");
    printf( "  Total count : %ld\n", sData[MENU_IDX].mCount );
    printf( "  Success count : %ld\n", sData[MENU_IDX].mSuccess );
    printf( "  Failure count : %ld\n", sData[MENU_IDX].mFailures );
    printf( "  Response time (min/avg/max/90th): %.6lf / %.6lf
/ %.6lf / %.6lf\n",
            sData[MENU_IDX].mRTMin,
            sData[MENU_IDX].mRTAvg,
            sData[MENU_IDX].mRTMax,
            sData[MENU_IDX].mRT90th );
    printf( "\n" );

    printf( "NEW-ORDER TRANSACTIONS:\n");
    printf( "  Total count : %ld\n", sData[0].mCount );
    printf( "  Success count : %ld\n", sData[0].mSuccess );
    printf( "  Failure count : %ld\n", sData[0].mFailures );
    printf( "  Total committed count : %ld\n", gCommitCount );
    printf( "  Total rolled back count : %ld\n", gRollbackCount );
    printf( "  Total Order-Line count : %ld\n", gTotalItemsCount );
    printf( "  Total Remote Order-Line count : %ld\n",
gRemoteItemCount );
    printf( "  Percentage of Total transactions: %.3lf%%\n",
            (double)((double)sData[0].mCount /
(double)sTransactionCount ) * 100 );
    printf( "  Keying time (min/avg/max): %.3lf / %.3lf / %.3lf\n",
            sData[0].mKeyMin,
            sData[0].mKeyAvg,
            sData[0].mKeyMax );
    printf( "  Response time (min/avg/max/90th): %.6lf / %.6lf
/ %.6lf / %.6lf\n",
            sData[0].mRTMin,
            sData[0].mRTAvg,
            sData[0].mRTMax,
            sData[0].mRT90th );
    printf( "  Think time (min/avg/max): %.3lf / %.3lf / %.3lf\n",
            sData[0].mThinkMin,
            sData[0].mThinkAvg,
            sData[0].mThinkMax );
    printf( "  Percentage of rolled-back transactions: %.3lf%%\n",
            (double)((double)gRollbackCount /
(double)sData[0].mSuccess) * 100 );
    printf( "  Average number of items per order: %.3lf\n",
            (double)((double)gTotalItemsCount /
(double)sData[0].mSuccess) );
    if( sWarehouseNum > 1 )
    {
        printf( "  Percentage of remote items: %.3lf%%\n",
                (double)((double)gRemoteItemCount /
gTotalItemsCount ) * 100 );
    }
    else
    {
        printf( "  Percentage of remote items: --- (One warehouse
only)\n" );
    }
    printf( "\n" );

    printf( "PAYMENT TRANSACTIONS:\n");
    printf( "  Total count : %ld\n", sData[1].mCount );
    printf( "  Success count : %ld\n", sData[1].mSuccess );
    printf( "  Failure count : %ld\n", sData[1].mFailures );
    printf( "  Total Remote count : %ld\n", gRemotePaymentCount );
    printf( "  Total by C_LAST count : %ld\n",
gPaymentByC_LASTCount );
    printf( "  Percentage of Total transactions: %.3lf%%\n",
            (double)((double)sData[1].mCount /
(double)sTransactionCount ) * 100 );
    printf( "  Keying time (min/avg/max): %.3lf / %.3lf / %.3lf\n",
            sData[1].mKeyMin,
            sData[1].mKeyAvg,
            sData[1].mKeyMax );
    printf( "  Response time (min/avg/max/90th): %.6lf / %.6lf
/ %.6lf / %.6lf\n",
            sData[1].mRTMin,
            sData[1].mRTAvg,
            sData[1].mRTMax,
            sData[1].mRT90th );
    printf( "  Think time (min/avg/max): %.3lf / %.3lf / %.3lf\n",
            sData[1].mThinkMin,
            sData[1].mThinkAvg,
            sData[1].mThinkMax );
    if( sWarehouseNum > 1 )
```

```
    {
        printf( "  Percentage or remote transactions: %.3lf%%\n",
                (double)((double)gRemotePaymentCount /
(double)sData[1].mSuccess) * 100);
    }
    else
    {
        printf( "  Percentage or remote transactions: --- (One
warehouse only)\n" );
    }
    printf( "  Percentage of customers selected by customer last
name: %.3lf%%\n",
            (double)((double)gPaymentByC_LASTCount /
(double)sData[1].mSuccess) * 100);
    printf( "\n" );

    printf( "ORDER-STATUS TRANSACTIONS:\n");
    printf( "  Total count : %ld\n", sData[2].mCount );
    printf( "  Success count : %ld\n", sData[2].mSuccess );
    printf( "  Failure count : %ld\n", sData[2].mFailures );
    printf( "  Total by C_LAST count : %ld\n",
gOrderStatusByC_LASTCount );
    printf( "  Percentage of Total transactions: %.3lf%%\n",
            (double)((double)sData[2].mCount /
(double)sTransactionCount ) * 100 );
    printf( "  Keying time (min/avg/max): %.3lf / %.3lf / %.3lf\n",
            sData[2].mKeyMin,
            sData[2].mKeyAvg,
            sData[2].mKeyMax );
    printf( "  Response time (min/avg/max/90th): %.6lf / %.6lf
/ %.6lf / %.6lf\n",
            sData[2].mRTMin,
            sData[2].mRTAvg,
            sData[2].mRTMax,
            sData[2].mRT90th );
    printf( "  Think time (min/avg/max): %.3lf / %.3lf / %.3lf\n",
            sData[2].mThinkMin,
            sData[2].mThinkAvg,
            sData[2].mThinkMax );
    printf( "  Percentage of customers selected by customer last
name: %.3lf%%\n",
            (double)((double)gOrderStatusByC_LASTCount /
(double)sData[2].mSuccess) * 100 );
    printf( "\n" );

    printf( "DELIVERY TRANSACTIONS (Interactive):\n");
    printf( "  Total count : %ld\n", sData[3].mCount );
    printf( "  Success count : %ld\n", sData[3].mSuccess );
    printf( "  Failure count : %ld\n", sData[3].mFailures );
    printf( "  Percentage of Total transactions: %.3lf%%\n",
            (double)((double)sData[3].mCount /
(double)sTransactionCount ) * 100 );
    printf( "  Keying time (min/avg/max): %.3lf / %.3lf / %.3lf\n",
            sData[3].mKeyMin,
            sData[3].mKeyAvg,
            sData[3].mKeyMax );
    printf( "  Response time (min/avg/max/90th): %.6lf / %.6lf
/ %.6lf / %.6lf\n",
            sData[3].mRTMin,
            sData[3].mRTAvg,
            sData[3].mRTMax,
            sData[3].mRT90th );
    printf( "  Think time (min/avg/max): %.3lf / %.3lf / %.3lf\n",
            sData[3].mThinkMin,
            sData[3].mThinkAvg,
            sData[3].mThinkMax );
    printf( "\n" );

    if( sDeferred == 1 )
    {
        printf( "DELIVERY TRANSACTIONS (Deferred):\n");
        printf( "  Total count : %ld\n",
sData[DELIVERY_DEFERRED_IDX].mCount );
        printf( "  Success count : %ld\n",
sData[DELIVERY_DEFERRED_IDX].mSuccess );
        printf( "  Total skipped count : %ld\n",
sData[DELIVERY_DEFERRED_IDX].mFailures );
        printf( "  Percentage of Deliveries skipped: %.3lf%%\n",

(double)((double)sData[DELIVERY_DEFERRED_IDX].mFailures /
(double)sData[DELIVERY_DEFERRED_IDX].mCount ) * 100 );
        printf( "  Response time (min/avg/max/90th): %.6lf / %.6lf
/ %.6lf / %.6lf\n",
                sData[DELIVERY_DEFERRED_IDX].mRTMin,
                sData[DELIVERY_DEFERRED_IDX].mRTAvg,
                sData[DELIVERY_DEFERRED_IDX].mRTMax,
                sData[DELIVERY_DEFERRED_IDX].mRT90th );
        printf( "\n" );
    }

    printf( "STOCK-LEVEL TRANSACTIONS:\n");
    printf( "  Total count : %ld\n", sData[4].mCount );
    printf( "  Success count : %ld\n", sData[4].mSuccess );
    printf( "  Failure count : %ld\n", sData[4].mFailures );
    printf( "  Percentage of Total transactions: %.3lf%%\n",
            (double)((double)sData[4].mCount /
(double)sTransactionCount ) * 100 );
    printf( "  Keying time (min/avg/max): %.3lf / %.3lf / %.3lf\n",
            sData[4].mKeyMin,
            sData[4].mKeyAvg,
            sData[4].mKeyMax );
    printf( "  Response time (min/avg/max/90th): %.6lf / %.6lf
/ %.6lf / %.6lf\n",
            sData[4].mRTMin,
            sData[4].mRTAvg,
            sData[4].mRTMax,
            sData[4].mRT90th );
    printf( "  Think time (min/avg/max): %.3lf / %.3lf / %.3lf\n",
            sData[4].mThinkMin,
            sData[4].mThinkAvg,
            sData[4].mThinkMax );

    printf( "\nTest Duration\n");
    printf( "    Test
Start                                : %04d-%02d-
%02d %02d:%02d:%02d\n",
            1900 + sTmTestStart.tm_year, 1 + sTmTestStart.tm_mon,
sTmTestStart.tm_mday,
            sTmTestStart.tm_hour, sTmTestStart.tm_min,
sTmTestStart.tm_sec );
    printf( "    Test
End                                  : %04d-%02d-
%02d %02d:%02d:%02d\n",
            1900 + sTmTestEnd.tm_year, 1 + sTmTestEnd.tm_mon,
sTmTestEnd.tm_mday,
            sTmTestEnd.tm_hour, sTmTestEnd.tm_min,
sTmTestEnd.tm_sec );
    printf( "    Ramp-up Time                                : %d
seconds\n",
            (int)(GetTimespecDiff( &sTestStartTime,
&sMeasureStartTime ) /
                  (double)USEC_PER_SEC + 0.5) );
    printf( "    Measurement Interval                        : %d
seconds\n",
            (int)(GetTimespecDiff( &sMeasureStartTime,
&sMeasureEndTime ) /
                  (double)USEC_PER_SEC + 0.5 ) );
    printf( "    Ramp-down Time                              : %d
seconds\n",
            (int)(GetTimespecDiff( &sMeasureEndTime, &sTestEndTime )
/
                  (double)USEC_PER_SEC + 0.5) );
    printf( "    Number of transaction (all types)\n"
            "    completed in Measurement
Interval           : %ld\n",
            sData[0].mSuccess + sData[1].mSuccess +
sData[2].mSuccess +
            sData[3].mSuccess + sData[4].mSuccess );

    /* ACID */
    if( sACIDTest == 1 )
    {
        printf( "    Number of committed New-Order
transactions  : %ld\n",
                sNewOrderCount );

        rewind( sACIDFile );

        sACIDItemArray = (ACIDItem*)malloc( sizeof(ACIDItem) *
sACIDItemCount );
        fread( sACIDItemArray, sizeof(ACIDItem), sACIDItemCount,
sACIDFile );

        rewind( sACIDFile );

        qsort( sACIDItemArray, sACIDItemCount, sizeof(ACIDItem),
sortACIDItem );
        fwrite( sACIDItemArray, sizeof(ACIDItem), sACIDItemCount,
sACIDFile );

        fclose( sACIDFile );
        sACIDFile = NULL;
    }

    printf( "\n< Graph >\n\n");

    /* Response Times Frequency Distribution for All Transactions
*/
    for( i = 0; i < 7; i++ )
    {
        if( i == 5 )
        {
            sprintf( sFileName, "../graph/RT_menu.html" );
        }
        else if( i == 3 )
        {
            sprintf( sFileName, "../graph/RT_3_i.html" );
        }
        else if( i == 6 )
        {
```

```
            sprintf( sFileName, "../graph/RT_3_d.html" );
        }
        else
        {
            sprintf( sFileName, "../graph/RT_%d.html", i );
        }

        sGraphFile = fopen( sFileName, "w" );

        fprintf( sGraphFile, REPORT_HTML_PART_1 );
        fprintf( sGraphFile, REPORT_HTML_PART_1_RT );
        fprintf( sGraphFile, "          ['0.000', 0, null, null,
null]" );

        sPrintMax = 0;

        sTemp = 0;

        for( j = 0; j < RT_MAXREC * RT_REC_PER_SEC; j++ )
        {
            sTemp += gRTGraph[i][j];

            if( ((j + 1) % 1000) == 0 )
            {
                if( (j / 1000) > (sData[i].mRT90thLine * 4) / 1000 )
                {
                    break;
                }

                fprintf( sGraphFile, ",\n          ['%.3lf', %ld, ",
                         (double)(j + 1) / (double)RT_REC_PER_SEC,
                         sTemp );

                sTemp = 0;

                if( (int)((double)(j + 1) / (double)RT_REC_PER_SEC
* 1000) == ceil(sData[i].mRTAvg * 1000) )
                {
                    fprintf( sGraphFile, "'Avg = %.6lf sec.', ",
sData[i].mRTAvg );
                }
                else
                {
                    fprintf( sGraphFile, "null, " );
                }

                if( (int)((double)(j + 1) / (double)RT_REC_PER_SEC
* 1000) == ceil(sData[i].mRT90th * 1000) )
                {
                    fprintf( sGraphFile, "'90th = %.6lf sec.', ",
sData[i].mRT90th );
                }
                else
                {
                    fprintf( sGraphFile, "null, " );
                }

                if( (int)((double)(j + 1) / (double)RT_REC_PER_SEC
* 1000) == ceil(sData[i].mRTMax * 1000) )
                {
                    sPrintMax = 1;
                    fprintf( sGraphFile, "'Max = %.6lf sec.']",
sData[i].mRTMax );
                    break;
                }
                else
                {
                    if( (sPrintMax == 0) && ((j + 1) >
sData[i].mRT90thLine * 4) )
                    {
                        fprintf( sGraphFile, "'Max = %.6lf sec.']",
sData[i].mRTMax );
                    }
                    else
                    {
                        fprintf( sGraphFile, "null]" );
                    }
                }
            }
        }

        fprintf( sGraphFile, REPORT_HTML_PART_2, gHAixs[0], "",
gVAixs[0] );
        fprintf( sGraphFile, gTitle[i] );
        fprintf( sGraphFile, REPORT_HTML_PART_3 );

        fclose( sGraphFile );
        sGraphFile = NULL;

        printf( "  %s created.\n", sFileName );
    }

    /* Think Times distribution for New Order Transactions */
    sprintf( sFileName, "../graph/ThinkTimes.html" );

    sGraphFile = fopen( sFileName, "w" );

    fprintf( sGraphFile, REPORT_HTML_PART_1 );
    fprintf( sGraphFile, REPORT_HTML_PART_1_THINK );

    for( i = 0; i < THINK_MAXREC; i++ )
    {
        if( i == 0 )
        {
            fprintf( sGraphFile,
                     "          ['%d', %d, ",
                     (i + 1),
                     gThinkGraph[i] );
        }
        else
        {
            fprintf( sGraphFile,
                     ",\n          ['%d', %d, ",
                     (i + 1),
                     gThinkGraph[i] );
        }

        if( (i + 1) == ceil(sData[0].mThinkAvg) )
        {
            fprintf( sGraphFile, "'Avg = %.6lf sec.', ",
sData[0].mThinkAvg );
        }
        else
        {
            fprintf( sGraphFile, "null, " );
        }

        if( (i + 1) == ceil(sData[0].mThinkMax) )
        {
            fprintf( sGraphFile, "'Max = %.6lf sec.']",
sData[0].mThinkMax );
        }
        else
        {
            if( (i + 1) == THINK_MAXREC )
            {
                fprintf( sGraphFile, "'Max = %.6lf sec.']",
sData[0].mThinkMax );
            }
            else
            {
                fprintf( sGraphFile, "null]" );
            }
        }
    }

    fprintf( sGraphFile, REPORT_HTML_PART_2, gHAixs[1], "",
gVAixs[1] );
    fprintf( sGraphFile, "NewOrder Think Time Distribution" );
    fprintf( sGraphFile, REPORT_HTML_PART_3 );

    printf( "  %s created.\n", sFileName );

    fclose( sGraphFile );
    sGraphFile = NULL;

    /* Throughput versus Time */
    sprintf( sFileName, "../graph/Throughput.html" );

    sGraphFile = fopen( sFileName, "w" );

    fprintf( sGraphFile, REPORT_HTML_PART_1 );
    fprintf( sGraphFile, REPORT_HTML_PART_1_THROUGHPUT );
    fprintf( sGraphFile, "          ['0', 0]" );

    for( i = 0; i < sTestIntervalCount; i++ )
    {
        fprintf( sGraphFile,
                 ",\n          ['%d', %d]",
                 (i + 1) * THROUGHPUT_INTERVAL_SIZE,
                 gThroghtputGraph[i] * 60 /
THROUGHPUT_INTERVAL_SIZE );
    }

    sprintf( sTestStartTimeString,
             " [%04d-%02d-%02d %02d:%02d:%02d]",
             1900 + sTmTestStart.tm_year,
             1 + sTmTestStart.tm_mon,
             sTmTestStart.tm_mday,
             sTmTestStart.tm_hour,
             sTmTestStart.tm_min,
             sTmTestStart.tm_sec );

    fprintf( sGraphFile, REPORT_HTML_PART_2, gHAixs[2],
sTestStartTimeString, gVAixs[2] );
    fprintf( sGraphFile, "New Order Throughput vs. Elapsed Time" );
    fprintf( sGraphFile, REPORT_HTML_PART_3 );

    printf( "  %s created.\n", sFileName );
```

```
    fclose( sGraphFile );
    sGraphFile = NULL;

    free( gThroghtputGraph );
    gThroghtputGraph = NULL;

    return 0;
}
```

stop.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#include <tpc.h>
#include <support.h>

int main( int aArgc, char * aArgv[] )
{
    int               sShmId = -1;
    key_t             sKeyVal = 1234;
    TransactionInfo * sTransactionInfo = NULL;

    sShmId = shmget( sKeyVal,
                     sizeof(TransactionInfo),
                     IPC_CREAT | 0666 );

    if( sShmId == -1 )
    {
        fprintf( stderr, "shmget error\n" );
        exit(1);
    }

    sTransactionInfo = (TransactionInfo*)shmat( sShmId, NULL, 0 );

    if( sTransactionInfo == (void*)-1 )
    {
        fprintf( stderr, "shmat error\n" );
        exit(1);
    }

    sTransactionInfo->mStop = 1;

    (void)shmdt( (const void*)sTransactionInfo );
    sTransactionInfo = NULL;
    sShmId = -1;

    printf("*******************************\n");
    printf("*** set shutdown flag => ok ***\n");
    printf("*******************************\n");

    return 0;
}
```

support.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include <assert.h>

#include <tpc.h>
#include <support.h>

unsigned int C_255  = 1;
unsigned int C_1023 = 2;
unsigned int C_8191 = 3;

static time_t cache_sec = 0;
static struct tm cache_datetime;

char * gName[] =
{
    "BAR", "OUGHT", "ABLE", "PRI", "PRES",
    "ESE", "ANTI", "CALLY", "ATION", "EING"
};

int RandomNumber( unsigned int * aSeed, int aMin, int aMax )
{
    return aMin + (rand_r(aSeed) % ((aMax - aMin) + 1));
}

void SetCValue( unsigned int aC_255, unsigned int aC_1023, unsigned
int aC_8191)
{
    C_255  = aC_255;
    C_1023 = aC_1023;
    C_8191 = aC_8191;
}

int NURand( unsigned int * aSeed, unsigned A, unsigned x, unsigned
y)
{
    unsigned int C = 0;

    switch( A )
    {
        case 255:  C = C_255;  break;
        case 1023: C = C_1023; break;
        case 8191: C = C_8191; break;
        default:
            fprintf( stderr,
                     "NURand: unexpected value (%d) of A used\n",
                     A );
            abort();
    }

    return (int)
        (((RandomNumber(aSeed, 0, A) | RandomNumber(aSeed, x, y)) +
C) % (y-x+1)) + x;
}

void Lastname( int aNum, char * aName )
{
    strcpy( aName, gName[aNum / 100] );
    strcat( aName, gName[(aNum / 10) % 10] );
    strcat( aName, gName[aNum % 10] );

    return;
}

/*
 * turn system time into database format
 * the format argument should be a strftime() format string that
produces
 *   a datetime string acceptable to the database
 */
void gettimestamp (char str[], size_t len)
{
    time_t t;
    struct tm datetime;

    t = time(NULL);

    if( t != cache_sec )
    {
        localtime_r(&t, &datetime);
        cache_sec = t;
        cache_datetime = datetime;
    }
    else
    {
        datetime = cache_datetime;
    }

    snprintf( str,
              len + 1,
              "%04d-%02d-%02d %02d:%02d:%02d",
              datetime.tm_year+1900,
              datetime.tm_mon+1,
              datetime.tm_mday,
              datetime.tm_hour,
              datetime.tm_min,
              datetime.tm_sec );
}

double GetTimespecDiff( struct timespec * aTimestamp1,
                        struct timespec * aTimestamp2 )
{
    return ((double)aTimestamp2->tv_sec * 1000000.0) +
((double)aTimestamp2->tv_nsec/1000.0) -
           ((double)aTimestamp1->tv_sec * 1000000.0) -
((double)aTimestamp1->tv_nsec/1000.0);
}
```

spt_proc.h

```
#define SQL_THROW( aLable )                     \
    goto aLable;

#define SQL_TRY( aExpression )                  \
    do                                          \
    {                                           \
        if( !(SQL_SUCCEEDED( aExpression ) ) )  \
        {                                       \
```

```
            goto SQL_FINISH_LABEL;              \
        }                                       \
    } while( 0 )

#define SQL_CATCH( aLabel )                     \
    goto SQL_FINISH_LABEL;                      \
aLabel:

#define SQL_RAMP( aLabel ) aLabel:

#define SQL_FINISH                              \
    goto SQL_FINISH_LABEL;                      \
SQL_FINISH_LABEL:
```

support.h

```
#ifndef _SUPPORT_H_
#define _SUPPORT_H_ 1

#define HTTP_SEND_BUFF_SIZE  (1024 * 32)
#define HTTP_RECV_BUFF_SIZE  (1024 * 1024)

typedef struct HttpContext
{
    int    mSocket;
    int    mPort;
    char * mReqPage;
    char * mHostIp;
    char   mGetMethodBuf[4096];
    char   mSendBuf[HTTP_SEND_BUFF_SIZE];
    char   mRecvBuf[HTTP_RECV_BUFF_SIZE];
} HttpContext;

int RandomNumber( unsigned int * aSeed, int aMin, int aMax );
void SetCValue( unsigned int aC_255, unsigned int aC_1023, unsigned
int aC_8192);
int NURand( unsigned int * aSeed, unsigned A, unsigned x, unsigned
y);

void Lastname( int aNum, char * aName );

void gettimestamp (char str[], size_t len);

double GetTimespecDiff( struct timespec * aTimestamp1,
                        struct timespec * aTimestamp2 );

#endif
```

tpc.h

```
#ifndef _TPC_H_
#define _TPC_H_ 1

#ifdef __cplusplus
#define BEGIN_CPP_DECLS     extern "C" {
#define END_CPP_DECLS       }
#else
#define BEGIN_CPP_DECLS
#define END_CPP_DECLS
#endif

BEGIN_CPP_DECLS

#define MAXITEMS      100000
#define CUST_PER_DIST 3000
#define DIST_PER_WARE 10
#define ORD_PER_DIST  3000

#define MAX_NUM_ITEMS 15
#define MAX_ITEM_LEN  24

#define KEYING_TIME_NEW_ORDER    18
#define KEYING_TIME_PAYMENT       3
#define KEYING_TIME_ORDER_STATUS  2
#define KEYING_TIME_DELIVERY      2
#define KEYING_TIME_STOCK_LEVEL   2

#define THINK_TIME_NEW_ORDER    (12.01)
#define THINK_TIME_PAYMENT      (12.01)
#define THINK_TIME_ORDER_STATUS (10.01)
#define THINK_TIME_DELIVERY      (5.01)
#define THINK_TIME_STOCK_LEVEL   (5.01)

#define MAX_RETRY_COUNT    ( 10 )

typedef struct TransactionInfo
{
    unsigned int mAtomic;
    unsigned int mTotal;
    int          mStop;
    int          mAlign;
    unsigned int mCount[5];
    unsigned int mFailCount[5];
    int          mWeight[5];
} TransactionInfo;

typedef enum
{
    TRANSACTION_TYPE_NEW_ORDER    = 0,
    TRANSACTION_TYPE_PAYMENT      = 1,
    TRANSACTION_TYPE_ORDER_STATUS = 2,
    TRANSACTION_TYPE_DELIVERY     = 3,
    TRANSACTION_TYPE_STOCK_LEVEL  = 4
} TransactionType;

typedef struct ThreadArg
{
    unsigned int      mTransSeed;
    unsigned int      mSeed;
    int               mW_ID;
    int               mTerminalID;
    char            * mWebIP;
    char            * mWebPage;
    int               mWebPort;
} ThreadArg;

/* protocol */
typedef struct ProtocolInfo
{
    int   mWarehouse;
    int   mRampup;
    int   mRampdown;
    int   mMeasure;
    int   mPrintInterval;
    int   mACIDTest;
    pid_t mPID;
} ProtocolInfo;

typedef struct ProtocolTpmC
{
    int   mIdx;
    float mTpmC;
} ProtocolTpmC;

/* Request */
typedef struct NewOrderItemReq
{
    int mOL_I_ID;
    int mOL_SUPPLY_W_ID;
    int mOL_QUANTITY;
} NewOrderItemReq;

typedef struct NewOrderReq
{
    int             mW_ID;
    int             mD_ID;
    int             mC_ID;
    int             mOL_CNT;
    int             mRemoteItemCount;
    int             mInvalidItem;
    NewOrderItemReq mItem[15];
} NewOrderReq;

typedef struct PaymentReq
{
    int    mW_ID;
    int    mD_ID;
    int    mC_ID;
    int    mC_W_ID;
    int    mC_D_ID;
    char   mC_LAST[16];
    double mH_AMOUNT;
} PaymentReq;

typedef struct OrderStatusReq
{
    int  mW_ID;
    int  mD_ID;
    int  mC_ID;
    char mC_LAST[16];
} OrderStatusReq;

typedef struct DeliveryReq
{
    int mW_ID;
    int mO_CARRIER_ID;
} DeliveryReq;

typedef struct StockLevelReq
{
    int mW_ID;
    int mD_ID;
    int mThreshold;
} StockLevelReq;

typedef struct TpccArg
```

```
{
    union
    {
        NewOrderReq    mNewOrder;
        PaymentReq     mPayment;
        OrderStatusReq mOrderStatus;
        DeliveryReq    mDelivery;
        StockLevelReq  mStockLevel;
    };
} TpccArg;

typedef struct ACIDItem
{
    long           mTransactionEndTimeSec;
    long           mTransactionEndTimeNano;
    unsigned short mW_ID;
    unsigned char  mD_ID;
    char           mRollbacked;
    int            mO_ID;
} ACIDItem;

END_CPP_DECLS

#endif
```

Common.java

```
package com.sunje.tpcc;

import java.io.PrintWriter;
import java.util.concurrent.atomic.AtomicInteger;

public class Common {
    static int MAX_NUM_ITEMS = 15;
    static int NEW_ORDER_STMT_COUNT = 9;
    private static AtomicInteger countOk = new AtomicInteger(0);
    private static AtomicInteger countFail = new AtomicInteger(0);
    private static int DEBUG = 0;

    static String     PAGE_HEADER = "<html><head><title>GOLDILOCKS-
tpcc</title></head><body>";
    static String     PAGE_FOOTER = "</body></html>";


    static public class TimerThread extends Thread {
        private boolean mStop = false;

        public void run() {
            System.out.println("test thread run..");
            while (mStop == false) {
                try {
                    sleep(1000 * 1);

                    System.out.println("test thread..");

                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
    }

    static TimerThread timer = null;

    static void trSuccess() {
        countOk.incrementAndGet();
    }

    static void trFail() {
        countFail.incrementAndGet();
    }

    static void startTimerThread() {
        if (timer == null) {
            timer = new TimerThread();
            timer.start();
        }
    }

    static void stopTimerThread() {
        if (timer != null) {
            timer.mStop = true;
        }
    }

    static int getTrOk() {
        return countOk.get();
    }

    static int getTrFail() {
        return countFail.get();
    }

    static void println(String str) {
        if (DEBUG == 1) {
            System.out.println(str);
        }
    }

    static void printMenuBar(PrintWriter writer)
    {
        writer.println( "<br><br><a
href=NewOrderInput.html>NewOrder</a>  "
                + "<a href=PaymentInput.html>Payment</a>  "
                + "<a href=OrderStatusInput.html>OrderStatus</a>
 "
                + "<a href=DeliveryInput.html>Delivery</a>  "
                + "<a href=StockLevelInput.html>StockLevel</a>
 "
                + "<a href=MainMenu.html>Main Menu</a>" );

    }
}
```

Delivery.java

```
package com.sunje.tpcc;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.BlockingQueue;
import java.sql.CallableStatement;
import java.sql.Connection;

import javax.naming.InitialContext;
import javax.sql.DataSource;

@SuppressWarnings("serial")
@WebServlet("/Delivery")
public class Delivery extends HttpServlet
{
    protected class ResultFile
    {
        private String mLogFilePath;
        private String mSeparator;

        private FileWriter mWriter;
        private StringBuffer mFileName;

        public ResultFile(String aDate)
        {
            mLogFilePath =
System.getProperty( "TPCC_DELIVERY_RESULT_DIR",

System.getProperty("user.home") );

            mSeparator = System.getProperty( "file.separator" );

            mFileName = new StringBuffer();

            mFileName.append( mLogFilePath );
            mFileName.append( mSeparator );
            mFileName.append( "DELIVERY_RESULT_" );
            mFileName.append( aDate );
            mFileName.append( ".log" );
        }

        public void record( String aQueuedTime,
                            int    aW_ID,
                            int    aO_CARRIER_ID,
                            String aResult,
                            String aCompletedTime )
        {
            StringBuffer sMsg = new StringBuffer();

            sMsg.append( aQueuedTime );
            sMsg.append( ' ' );
            sMsg.append( aW_ID );
            sMsg.append( ' ' );
            sMsg.append( aO_CARRIER_ID );
            sMsg.append( ' ' );
            sMsg.append( aResult );
            sMsg.append( ' ' );
            sMsg.append( aCompletedTime );
```

```
            try
            {
                mWriter = new FileWriter(mFileName.toString(),
true);
                mWriter.write( sMsg.toString() );
                mWriter.write( "\n" );
            }
            catch( IOException e )
            {
                Common.println( "Exception Encountered : " + e );
            }
            finally
            {
                try
                {
                    mWriter.close();
                }
                catch( Exception e )
                {
                    Common.println( "Exception Encountered : " +
e );
                }
            }
        }
    }

    class DeliveryReq
    {
          Date mQueuedTime;
        int  mW_ID;
        int  mO_CARRIER_ID;
    };

    class DeliveryRes
    {
        String mResult;
    };

    protected BlockingQueue<DeliveryReq> mQueue;

    class BatchTransaction implements Runnable
    {
          private SimpleDateFormat mFormatter;
          private ResultFile mResultFile;
          private BlockingQueue<DeliveryReq> mQueue;

         public BatchTransaction(BlockingQueue<DeliveryReq> aQueue)
          {
            Calendar sCal = Calendar.getInstance();

            mQueue = aQueue;

            mResultFile = new ResultFile((new
SimpleDateFormat("yyyyMMdd_HHmm")).format(sCal.getTime()));
            mFormatter = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS");
        }

       public boolean DeliveryTransaction(DeliveryReq aDeliveryReq,
DeliveryRes aDeliveryRes)
        {
            DataSource ds = null;
            Connection conn = null;
            InitialContext ctx;
            int sIdx = 1;
            int sOutIdx;
            boolean sReturn;
            int sSuccess;

            try
            {
                ctx = new InitialContext();
                ds = (DataSource)
ctx.lookup("java:comp/env/jdbc/goldilocks");
                conn = ds.getConnection();


conn.setTransactionIsolation(Connection.TRANSACTION_SERIALIZABLE);

                CallableStatement stmt;

                stmt = conn.prepareCall("{CALL Delivery(?,?,?,?)}");

                stmt.setInt(sIdx++, aDeliveryReq.mW_ID);
                stmt.setInt(sIdx++, aDeliveryReq.mO_CARRIER_ID);

                sOutIdx = sIdx;
                stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);

                stmt.registerOutParameter(sIdx++,
java.sql.Types.INTEGER);

                // transaction..
                stmt.execute();

                aDeliveryRes.mResult = stmt.getString(sOutIdx++);
                sSuccess = stmt.getInt(sOutIdx++);

                stmt.close();
                conn.close();
                ctx.close();

                if (sSuccess == 1)
                {
                    sReturn = true;
                    Common.trSuccess();
                }
                else
                {
                    sReturn = false;
                    Common.trFail();
                }
            }
            catch (Exception e)
            {
                sReturn = false;
                System.out.println("Exception Encountered : " + e);
                Common.trFail();
            }
            finally
            {
                if (conn != null)
                {
                    conn = null;
                }
            }

            return sReturn;
        }

          @Override
          public void run()
          {
            boolean sReturn;
            DeliveryReq sDeliveryReq;
            DeliveryRes sDeliveryRes = new DeliveryRes();
            Calendar sCal;

            while(true)
            {
                try
                {
                    sDeliveryReq = mQueue.take();

                    sReturn = DeliveryTransaction(sDeliveryReq,
sDeliveryRes);

                    if(sReturn == true)
                    {
                        /**
                         * Upon completion of the business
transaction, the following information must have been recorded into
a result file:
                         * 1. The time at which the business
transaction was queued.
                         * 2. The warehouse number (W_ID) and the
carried number (O_CARRIER_ID) associated with the business
transaction.
                         * 3. The district number (D_ID) and the
order number (O_ID) of each order delivered by the business
transaction.
                         * 4. The time at which the business
transaction completed.
                         */

                        sCal = Calendar.getInstance();


mResultFile.record( mFormatter.format(sDeliveryReq.mQueuedTime),
                                                sDeliveryReq.mW_ID,

sDeliveryReq.mO_CARRIER_ID,
                                                sDeliveryRes.mResult,

mFormatter.format(sCal.getTime()) );
                    }
                }
                catch(InterruptedException e)
                {
                    Common.println( "Exception Encountered : " +
e );
                }
            }
          }
    }

    public Delivery()
    {
```

```
        try
        {
            mQueue = new ArrayBlockingQueue<DeliveryReq>(2000);

            Thread sBatchTransaction = new Thread(new
BatchTransaction(mQueue));
            sBatchTransaction.start();
        }
        catch (Exception e)
        {
            Common.println( "Exception Encountered : " + e );
        }
    }

    @Override
    protected void doGet(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException
    {
        resp.setContentType( "text/html" );
        PrintWriter writer = resp.getWriter();
        writer.println( Common.PAGE_HEADER );

        if( printResult( writer, req ) == false )
        {
            resp.sendError( HttpServletResponse.SC_BAD_REQUEST );
        }

        Common.printMenuBar( writer );

        writer.println( Common.PAGE_FOOTER );
        writer.close();
    }

    protected boolean printResult(PrintWriter writer,
HttpServletRequest req)
    {
          boolean sResult;
        DeliveryReq sDeliveryReq = new DeliveryReq();
        Calendar sCal = Calendar.getInstance();

        sDeliveryReq.mW_ID =
Integer.parseInt( req.getParameter("W_ID" ) );
        sDeliveryReq.mO_CARRIER_ID =
Integer.parseInt( req.getParameter("O_CARRIER_ID" ) );
        sDeliveryReq.mQueuedTime = sCal.getTime();

        try
        {
            mQueue.put(sDeliveryReq);

            sResult = true;

            writer.println( "<pre>" );
            writer.format( "%34cDelivery\n",32 );
            writer.format( "Warehouse: %4d\n\n",
sDeliveryReq.mW_ID );
            writer.format( "Carrier Number: %2d\n\n",
sDeliveryReq.mO_CARRIER_ID );
            writer.format( "Execution Status: Delivery has been
queued \n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n" );
            writer.println( "</pre>" );
        }
        catch(InterruptedException e)
        {
            sResult = false;
            Common.println( "Exception Encountered : " + e );

        }

        return sResult;
    }
}
```

NewOrder.java

```
package com.sunje.tpcc;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.CallableStatement;
import java.sql.Connection;

import javax.naming.InitialContext;
import javax.sql.DataSource;

@SuppressWarnings("serial")
@WebServlet("/NewOrder")
public class NewOrder extends HttpServlet
{
    class NewOrderItemReq
    {
        int mOL_I_ID;
        int mOL_SUPPLY_W_ID;
        int mOL_QUANTITY;
    };

    class NewOrderReq
    {
        int    mW_ID;
        int    mD_ID;
        int    mC_ID;
        int    mOL_CNT;
        int    mALL_LOCAL;
        String mItemString;
    };

    class NewOrderRes
    {
        String  mC_LAST;
        String  mC_CREDIT;
        float   mC_DISCOUNT;
        int     mO_ID;
        String  mO_ENTRY_D;
        float   mW_TAX;
        float   mD_TAX;
        boolean mIsRollbacked;
        float   mTotalAmount;
        String  mItemData;
    };

    @Override
    protected void doGet(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException
    {
        StringBuilder sMessage = new StringBuilder();

        resp.setContentType( "text/html" );
        PrintWriter writer = resp.getWriter();
        writer.println( Common.PAGE_HEADER );

        if( printResult( writer, req, sMessage ) == false )
        {
            resp.sendError( HttpServletResponse.SC_BAD_REQUEST,
sMessage.toString() );
        }

        Common.printMenuBar( writer );

        writer.println( Common.PAGE_FOOTER );
        writer.close();
    }

    boolean NewOrderTransaction( NewOrderReq   aNewOrderReq,
                                 NewOrderRes   aNewOrderRes,
                                 StringBuilder aMessage )
    {
        DataSource ds = null;
        Connection conn = null;
        InitialContext ctx;
        int sIdx = 1;
        int sOutIdx;
        boolean sReturn;
        int sSuccess;

        try
        {
            ctx = new InitialContext();
            ds = (DataSource)
ctx.lookup( "java:comp/env/jdbc/goldilocks" );
            conn = ds.getConnection();


conn.setTransactionIsolation( Connection.TRANSACTION_SERIALIZABLE );

            CallableStatement stmt;

            stmt = conn.prepareCall( "{CALL
NewOrder(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)}" );

            stmt.setInt(sIdx++, aNewOrderReq.mW_ID);
            stmt.setInt(sIdx++, aNewOrderReq.mD_ID);
            stmt.setInt(sIdx++, aNewOrderReq.mC_ID);
            stmt.setInt(sIdx++, aNewOrderReq.mOL_CNT);
            stmt.setInt(sIdx++, aNewOrderReq.mALL_LOCAL);
            stmt.setString(sIdx++, aNewOrderReq.mItemString);

            sOutIdx = sIdx;
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,  java.sql.Types.FLOAT);
```

```
            stmt.registerOutParameter(sIdx++,  java.sql.Types.FLOAT);
            stmt.registerOutParameter(sIdx++,  java.sql.Types.FLOAT);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.INTEGER);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,  java.sql.Types.FLOAT);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.INTEGER);

            stmt.registerOutParameter(sIdx++,
java.sql.Types.INTEGER);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);

            // transaction..
            stmt.execute();

            aNewOrderRes.mC_LAST      = stmt.getString(sOutIdx++);
            aNewOrderRes.mC_CREDIT    = stmt.getString(sOutIdx++);
            aNewOrderRes.mC_DISCOUNT  = stmt.getFloat(sOutIdx++);
            aNewOrderRes.mW_TAX       = stmt.getFloat(sOutIdx++);
            aNewOrderRes.mD_TAX       = stmt.getFloat(sOutIdx++);
            aNewOrderRes.mO_ID        = stmt.getInt(sOutIdx++);
            aNewOrderRes.mO_ENTRY_D   = stmt.getString(sOutIdx++);
            aNewOrderRes.mTotalAmount = stmt.getFloat(sOutIdx++);
            aNewOrderRes.mItemData    = stmt.getString(sOutIdx++);

            if( stmt.getInt(sOutIdx++) == 0 )
            {
                aNewOrderRes.mIsRollbacked = false;
            }
            else
            {
                aNewOrderRes.mIsRollbacked = true;
            }

            sSuccess = stmt.getInt(sOutIdx++);

            if( sSuccess != 1 )
            {
                aMessage.append(stmt.getString(sOutIdx++));
            }

            stmt.close();
            conn.close();
            ctx.close();

            if( sSuccess == 1 )
            {
                    sReturn = true;
                    Common.trSuccess();
            }
            else
            {
                    sReturn = false;
                    Common.trFail();
            }
        }
        catch( Exception e )
        {
            sReturn = false;
            System.out.println( "Exception Encountered : " + e );
            Common.trFail();
        }
        finally
        {
            if( conn != null )
            {
                conn = null;
            }
        }

        return sReturn;
    }


    protected boolean printResult(PrintWriter writer,
HttpServletRequest req, StringBuilder aMessage)
    {
        int i;
        NewOrderReq sNewOrderReq = new NewOrderReq();
        NewOrderRes sNewOrderRes = new NewOrderRes();
        NewOrderItemReq sReqItem[] = new
NewOrderItemReq[Common.MAX_NUM_ITEMS];
        int sIndex = 0;
        int sEndPos = 0;
        int sStock;
        float sPrice;
        float sAmount;
        boolean sReturn;

        sNewOrderReq.mW_ID =
Integer.parseInt( req.getParameter( "W_ID" ) );
        sNewOrderReq.mD_ID =
Integer.parseInt( req.getParameter( "D_ID" ) );
        sNewOrderReq.mC_ID =
Integer.parseInt( req.getParameter( "C_ID" ) );
        sNewOrderReq.mOL_CNT = 0;
        sNewOrderReq.mALL_LOCAL = 1;
        sNewOrderReq.mItemString = "";

        for( i = 0; i < Common.MAX_NUM_ITEMS; i++ )
        {
            if( (req.getParameter( "OL_SUPPLY_W_ID_" + i ) == null)
||
                (req.getParameter( "OL_I_ID_" + i ) == null)
||
                (req.getParameter( "OL_QUANTITY_" + i ) == null) )
            {
                continue;
            }

            if( (req.getParameter( "OL_SUPPLY_W_ID_" + i ).isEmpty()
== true) ||
                (req.getParameter( "OL_I_ID_" + i ).isEmpty() ==
true)        ||
                (req.getParameter( "OL_QUANTITY_" + i ).isEmpty()
== true) )
            {
                continue;
            }

            sIndex = sNewOrderReq.mOL_CNT;

            sReqItem[sIndex] = new NewOrderItemReq();

            sReqItem[sIndex].mOL_SUPPLY_W_ID =
Integer.parseInt( req.getParameter( "OL_SUPPLY_W_ID_" + i ) );
            sReqItem[sIndex].mOL_I_ID        =
Integer.parseInt( req.getParameter( "OL_I_ID_" + i ) );
            sReqItem[sIndex].mOL_QUANTITY    =
Integer.parseInt( req.getParameter( "OL_QUANTITY_" + i ) );

            sNewOrderReq.mItemString += sReqItem[sIndex].mOL_I_ID +
"|"
                                        +
sReqItem[sIndex].mOL_SUPPLY_W_ID + "|"
                                          +
sReqItem[sIndex].mOL_QUANTITY    + "|";

            sNewOrderReq.mOL_CNT++;

            if( sNewOrderReq.mW_ID !=
sReqItem[sIndex].mOL_SUPPLY_W_ID )
            {
                sNewOrderReq.mALL_LOCAL = 0;
            }
        }

        sReturn = NewOrderTransaction( sNewOrderReq, sNewOrderRes,
aMessage );

        if( sReturn == true )
        {
            writer.println( "<pre>" );
            writer.format( "%35cNew Order\n", 32 );

            if( sNewOrderRes.mIsRollbacked == false )
            {
                writer.format( "Warehouse: %4d
District: %2d%24cDate: %s\n",
                               sNewOrderReq.mW_ID,
                               sNewOrderReq.mD_ID, 32,
                               sNewOrderRes.mO_ENTRY_D );
                writer.format( "Customer:  %4d   Name: %16s
Credit: %s   %%Disc: %5.2f\n",
                               sNewOrderReq.mC_ID,
                               sNewOrderRes.mC_LAST,
                               sNewOrderRes.mC_CREDIT,
                               sNewOrderRes.mC_DISCOUNT );
                writer.format( "Order Number: %8d  Number of
Lines: %2d%8cW_tax: %5.2f   D_tax: %5.2f\n\n",
                               sNewOrderRes.mO_ID,
                               sNewOrderReq.mOL_CNT, 32,
                               sNewOrderRes.mW_TAX,
                               sNewOrderRes.mD_TAX );
                writer.format( " Supp_W  Item_Id  Item Name%17cQty
Stock  B/G   Price    Amount\n",
                               32 );

                sIndex = 0;
                sEndPos = 0;

                for( i = 0; i < sNewOrderReq.mOL_CNT; i++ )
                {
```

```
                    writer.format( "  %4d",
sReqItem[i].mOL_SUPPLY_W_ID );
                    writer.format( "   %6d", sReqItem[i].mOL_I_ID );

                    if( sNewOrderRes.mItemData != null &&
sNewOrderRes.mItemData.isEmpty() == false )
                    {
                        sEndPos =
sNewOrderRes.mItemData.indexOf( '|', sIndex );
                        writer.format( "   %-24s",
sNewOrderRes.mItemData.substring(sIndex, sEndPos) );
                        sIndex = sEndPos + 1;

                        writer.format( "  %2d",
sReqItem[i].mOL_QUANTITY );

                        sEndPos =
sNewOrderRes.mItemData.indexOf( '|', sIndex );
                        sStock =
Integer.parseInt(sNewOrderRes.mItemData.substring(sIndex,  sEndPos));
                        writer.format( "    %3d", sStock);
                        sIndex = sEndPos + 1;

                        sEndPos =
sNewOrderRes.mItemData.indexOf( '|', sIndex );
                        writer.format( "    %s",
sNewOrderRes.mItemData.substring(sIndex, sEndPos) );
                        sIndex = sEndPos + 1;

                        sEndPos =
sNewOrderRes.mItemData.indexOf( '|', sIndex );
                        sPrice =
Float.parseFloat(sNewOrderRes.mItemData.substring(sIndex,  sEndPos));
                        writer.format( "    $%6.2f", sPrice );
                        sIndex = sEndPos + 1;

                        sEndPos =
sNewOrderRes.mItemData.indexOf( '|', sIndex );
                        sAmount =
Float.parseFloat(sNewOrderRes.mItemData.substring(sIndex,  sEndPos));
                        writer.format( "  $%7.2f\n", sAmount );
                        sIndex = sEndPos + 1;
                    }
                    else
                    {
                        writer.format( "\n\n\nitem data null\n" );
                    }
                }

                for( ; i < Common.MAX_NUM_ITEMS; i++ )
                {
                    writer.println( "" );
                }

                writer.format( "Execution Status: TRANSACTION
COMMITED    %20cTotal:  $%8.2f\n",
                               32,
                               sNewOrderRes.mTotalAmount );
            }
            else
            {
                writer.format( "Warehouse: %4d
District: %2d%24cDate: YYYY-MM-DD hh:mm:ss\n",
                               sNewOrderReq.mW_ID,
                               sNewOrderReq.mD_ID,
                               32 );
                writer.format( "Customer:  %4d   Name: %16s
Credit: %s   %%Disc: --.--\n",
                               sNewOrderReq.mC_ID,
                               sNewOrderRes.mC_LAST,
                               sNewOrderRes.mC_CREDIT );
                writer.format( "Order Number: %8d  Number of Lines:
--%8cW_tax: --.--   D_tax: --.--\n\n",
                               sNewOrderRes.mO_ID,
                               32 );
                writer.format( " Supp_W  Item_Id  Item Name%17cQty
Stock  B/G   Price    Amount\n",
                               32);

                for (i = 0; i < sNewOrderReq.mOL_CNT; i++)
                {
                    writer.format( "  %4d   %6d   -----------------
-------  %2d    ---    -    $---.--  $----.--\n",
                                   sReqItem[i].mOL_SUPPLY_W_ID,
                                   sReqItem[i].mOL_I_ID,
                                   sReqItem[i].mOL_QUANTITY );
                }

                for( ; i < Common.MAX_NUM_ITEMS; i++ )
                {
                    writer.println( "" );
                }

                writer.format( "Execution Status: ITEM NUMBER IS NO
VALID %20cTotal:  $-----.--\n",
                               32);
            }
            writer.println( "</pre>" );
        }

        return sReturn;
    }
}
```

OrderStatus.java

```
package com.sunje.tpcc;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.CallableStatement;
import java.sql.Connection;

import javax.naming.InitialContext;
import javax.sql.DataSource;

@SuppressWarnings("serial")
@WebServlet("/OrderStatus")
public class OrderStatus extends HttpServlet
{
    class OrderStatusReq
    {
        int    mW_ID;
        int    mD_ID;
        int    mC_ID;
        String mC_LAST;
    };

    class OrderStatusRes
    {
        int    mC_ID;
        String mC_FIRST;
        String mC_MIDDLE;
        String mC_LAST;
        float  mC_BALANCE;
        int    mO_ID;
        String mO_ENTRY_D;
        int    mO_CARRIER_ID;
        int    mO_OL_COUNT;
        String mItemData;
    };

    @Override
    protected void doGet(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException
    {
        StringBuilder sMessage = new StringBuilder();

        resp.setContentType( "text/html" );
        PrintWriter writer = resp.getWriter();
        writer.println( Common.PAGE_HEADER );

        if( printResult( writer, req, sMessage ) == false )
        {
            resp.sendError( HttpServletResponse.SC_BAD_REQUEST,
sMessage.toString() );
        }

        Common.printMenuBar( writer );

        writer.println( Common.PAGE_FOOTER );
        writer.close();
    }

    boolean OrderStatusTransaction( OrderStatusReq aOrderStatusReq,
                                    OrderStatusRes aOrderStatusRes,
                                    StringBuilder  aMessage )
    {
        DataSource ds = null;
        Connection conn = null;
        InitialContext ctx;
        int sIdx = 1;
        int sOutIdx;
        boolean sReturn;
        int sSuccess;

        try
        {
            ctx = new InitialContext();
            ds = (DataSource)
ctx.lookup( "java:comp/env/jdbc/goldilocks" );
            conn = ds.getConnection();
```

```
conn.setTransactionIsolation( Connection.TRANSACTION_SERIALIZABLE );

            CallableStatement stmt;

            stmt = conn.prepareCall( "{CALL
OrderStatus(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)}" );

            stmt.setInt(sIdx++, aOrderStatusReq.mW_ID);
            stmt.setInt(sIdx++, aOrderStatusReq.mD_ID);
            stmt.setInt(sIdx++, aOrderStatusReq.mC_ID);
            stmt.setString(sIdx++, aOrderStatusReq.mC_LAST);

            sOutIdx = sIdx;
            stmt.registerOutParameter(sIdx++,
java.sql.Types.INTEGER);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.INTEGER);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,  java.sql.Types.FLOAT);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.INTEGER);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.INTEGER);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);

            stmt.registerOutParameter(sIdx++,
java.sql.Types.INTEGER);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);

            // transaction..
            stmt.execute();

            aOrderStatusRes.mO_ID         = stmt.getInt(sOutIdx++);
            aOrderStatusRes.mC_ID         = stmt.getInt(sOutIdx++);
            aOrderStatusRes.mC_FIRST      =
stmt.getString(sOutIdx++);
            aOrderStatusRes.mC_MIDDLE     =
stmt.getString(sOutIdx++);
            aOrderStatusRes.mC_LAST       =
stmt.getString(sOutIdx++);
            aOrderStatusRes.mO_ENTRY_D    =
stmt.getString(sOutIdx++);
            aOrderStatusRes.mC_BALANCE    =
stmt.getFloat(sOutIdx++);
            aOrderStatusRes.mO_CARRIER_ID = stmt.getInt(sOutIdx++);
            aOrderStatusRes.mO_OL_COUNT   = stmt.getInt(sOutIdx++);
            aOrderStatusRes.mItemData     =
stmt.getString(sOutIdx++);
            sSuccess                      = stmt.getInt(sOutIdx++);

            if( sSuccess != 1 )
            {
                aMessage.append(stmt.getString(sOutIdx++));
            }

            stmt.close();
            conn.close();
            ctx.close();

            if( sSuccess == 1 )
            {
                    sReturn = true;
                    Common.trSuccess();
            }
            else
            {
                    sReturn = false;
                    Common.trFail();
            }
        }
        catch( Exception e )
        {
            sReturn = false;
            System.out.println( "Exception Encountered : " + e );
            Common.trFail();
        }
        finally
        {
            if( conn != null )
            {
                conn = null;
            }
        }
        return sReturn;
    }

    protected boolean printResult(PrintWriter writer,
HttpServletRequest req, StringBuilder aMessage)
    {
        OrderStatusReq sOrderStatusReq = new OrderStatusReq();
        OrderStatusRes sOrderStatusRes = new OrderStatusRes();

        int sSUPPLY_W_ID;
        int sI_ID;
        int sQUANTITY;
        float sAMOUNT;

        int sIndex = 0;
        int sEndPos = 0;

        int i;

        boolean sReturn;

        sOrderStatusReq.mW_ID =
Integer.parseInt( req.getParameter( "W_ID" ) );
        sOrderStatusReq.mD_ID =
Integer.parseInt( req.getParameter( "D_ID" ) );
        sOrderStatusReq.mC_ID = 0;

        if( req.getParameter( "C_ID" ) != null &&
req.getParameter( "C_ID").isEmpty() == false )
        {
            sOrderStatusReq.mC_ID =
Integer.parseInt( req.getParameter( "C_ID" ) );
        }

        sOrderStatusReq.mC_LAST = req.getParameter( "C_LAST" );

        sReturn = OrderStatusTransaction( sOrderStatusReq,
sOrderStatusRes, aMessage );

        if( sReturn == true )
        {
            writer.println( "<pre>" );
            writer.format( "%31cOrder-Status\n", 32 );
            writer.format( "Warehouse: %4d   District: %2d\n",
                           sOrderStatusReq.mW_ID,
                           sOrderStatusReq.mD_ID );
            writer.format( "Customer: %4d   Name: %16s %2s %-16s\n",
                           sOrderStatusRes.mC_ID,
                           sOrderStatusRes.mC_FIRST,
                           sOrderStatusRes.mC_MIDDLE,
                           sOrderStatusRes.mC_LAST );
            writer.format( "Cust-Balance: $%+9.2f\n\n",
                           sOrderStatusRes.mC_BALANCE );
            writer.format( "Order-Number: %8d   Entry-Date: %.19s
Carrier-Number:",
                           sOrderStatusRes.mO_ID,
                           sOrderStatusRes.mO_ENTRY_D );

            if( sOrderStatusRes.mO_CARRIER_ID == 0 )
            {
                writer.format( " --\n" );
            }
            else
            {
                writer.format( " %2d\n",
sOrderStatusRes.mO_CARRIER_ID );
            }

            writer.format( "Supply_W     Item_Id    Qty     Amount
Delivery_Date\n" );

            sIndex = 0;
            sEndPos = 0;

            for( i = 0; i < sOrderStatusRes.mO_OL_COUNT; i++ )
            {
                sEndPos = sOrderStatusRes.mItemData.indexOf( '|',
sIndex );
                sSUPPLY_W_ID =
Integer.parseInt(sOrderStatusRes.mItemData.substring(sIndex,
sEndPos));
                writer.format( "  %4d", sSUPPLY_W_ID );
                sIndex = sEndPos + 1;

                sEndPos = sOrderStatusRes.mItemData.indexOf( '|',
sIndex );
                sI_ID =
Integer.parseInt(sOrderStatusRes.mItemData.substring(sIndex,
sEndPos));
                writer.format( "       %6d", sI_ID );
                sIndex = sEndPos + 1;

                sEndPos = sOrderStatusRes.mItemData.indexOf( '|',
sIndex );
```

```
                sQUANTITY =
Integer.parseInt(sOrderStatusRes.mItemData.substring(sIndex,
sEndPos));
                writer.format( "     %2d", sQUANTITY );
                sIndex = sEndPos + 1;

                sEndPos = sOrderStatusRes.mItemData.indexOf( '|',
sIndex );
                sAMOUNT =
Float.parseFloat(sOrderStatusRes.mItemData.substring(sIndex,
sEndPos));
                writer.format( "     $%8.2f", sAMOUNT );
                sIndex = sEndPos + 1;

                sEndPos = sOrderStatusRes.mItemData.indexOf( '|',
sIndex );
                if( sIndex != sEndPos )
                {
                    writer.format( "       %.10s\n",
sOrderStatusRes.mItemData.substring(sIndex, sEndPos) );
                }
                else
                {
                    writer.format( "       YYYY-MM-DD\n" );
                }
                sIndex = sEndPos + 1;
            }

            for( ; i < Common.MAX_NUM_ITEMS; i++ )
            {
                writer.println( "" );
            }

            writer.println( "</pre>" );
        }

        return sReturn;
    }
}
```

Payment.java

```
package com.sunje.tpcc;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import java.sql.CallableStatement;
import java.sql.Connection;

import javax.naming.InitialContext;
import javax.sql.DataSource;

@SuppressWarnings("serial")
@WebServlet("/Payment")
public class Payment extends HttpServlet
{

    class PaymentReq
    {
        int    mW_ID;
        int    mD_ID;
        int    mC_ID;
        int    mC_W_ID;
        int    mC_D_ID;
        String mC_LAST;
        float  mH_AMOUNT;
    };

    class PaymentRes
    {
        String  mW_STREET_1;
        String  mW_STREET_2;
        String  mW_CITY;
        String  mW_STATE;
        String  mW_ZIP;
        String  mH_DATE;
        String  mD_STREET_1;
        String  mD_STREET_2;
        String  mD_CITY;
        String  mD_STATE;
        String  mD_ZIP;
        int     mC_ID;
        String  mC_FIRST;
        String  mC_MIDDLE;
        String  mC_LAST;
        String  mC_STREET_1;
        String  mC_STREET_2;
        String  mC_CITY;
        String  mC_STATE;
        String  mC_ZIP;
        String  mC_PHONE;
        String  mC_CREDIT;
        String  mC_SINCE;
        float   mC_CREDIT_LIM;
        float   mC_DISCOUNT;
        float   mC_BALANCE;
        String  mC_DATA;
    };

    @Override
    protected void doGet(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException
    {
        StringBuilder sMessage = new StringBuilder();

        resp.setContentType( "text/html" );
        PrintWriter writer = resp.getWriter();
        writer.println( Common.PAGE_HEADER );

        if( printResult( writer, req, sMessage ) == false )
        {
            resp.sendError( HttpServletResponse.SC_BAD_REQUEST,
sMessage.toString() );
        }

        Common.printMenuBar( writer );

        writer.println( Common.PAGE_FOOTER );
        writer.close();
    }

    boolean PaymentTransaction( PaymentReq    aPaymentReq,
                                PaymentRes    aPaymentRes,
                                StringBuilder aMessage )
    {
        DataSource ds = null;
        Connection conn = null;
        InitialContext ctx;
        boolean sReturn;
        int sSuccess;

        try
        {
            ctx = new InitialContext();
            ds = (DataSource)
ctx.lookup( "java:comp/env/jdbc/goldilocks" );
            conn = ds.getConnection();


conn.setTransactionIsolation( Connection.TRANSACTION_SERIALIZABLE );

            int sIdx = 1;
            int sOutIdx = 1;

            CallableStatement stmt;

            stmt = conn.prepareCall( "{CALL
Payment(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?)}" );

            stmt.setInt(sIdx++, aPaymentReq.mW_ID);
            stmt.setInt(sIdx++, aPaymentReq.mD_ID);
            stmt.setInt(sIdx++, aPaymentReq.mC_ID);
            stmt.setInt(sIdx++, aPaymentReq.mC_W_ID);
            stmt.setInt(sIdx++, aPaymentReq.mC_D_ID);
            stmt.setString(sIdx++, aPaymentReq.mC_LAST);
            stmt.setFloat(sIdx++, aPaymentReq.mH_AMOUNT);

            sOutIdx = sIdx;
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);

            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);

            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
```

```
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);

            stmt.registerOutParameter(sIdx++,
java.sql.Types.INTEGER);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);

            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,  java.sql.Types.FLOAT);
            stmt.registerOutParameter(sIdx++,  java.sql.Types.FLOAT);
            stmt.registerOutParameter(sIdx++,  java.sql.Types.FLOAT);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);

            stmt.registerOutParameter(sIdx++,
java.sql.Types.INTEGER);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);

            // transaction..
            stmt.execute();

            aPaymentRes.mW_STREET_1   = stmt.getString(sOutIdx++);
            aPaymentRes.mW_STREET_2   = stmt.getString(sOutIdx++);
            aPaymentRes.mW_CITY       = stmt.getString(sOutIdx++);
            aPaymentRes.mW_STATE      = stmt.getString(sOutIdx++);
            aPaymentRes.mW_ZIP        = stmt.getString(sOutIdx++);
            aPaymentRes.mH_DATE       = stmt.getString(sOutIdx++);
            aPaymentRes.mD_STREET_1   = stmt.getString(sOutIdx++);
            aPaymentRes.mD_STREET_2   = stmt.getString(sOutIdx++);
            aPaymentRes.mD_CITY       = stmt.getString(sOutIdx++);
            aPaymentRes.mD_STATE      = stmt.getString(sOutIdx++);
            aPaymentRes.mD_ZIP        = stmt.getString(sOutIdx++);
            aPaymentRes.mC_ID         = stmt.getInt(sOutIdx++);
            aPaymentRes.mC_FIRST      = stmt.getString(sOutIdx++);
            aPaymentRes.mC_MIDDLE     = stmt.getString(sOutIdx++);
            aPaymentRes.mC_LAST       = stmt.getString(sOutIdx++);
            aPaymentRes.mC_STREET_1   = stmt.getString(sOutIdx++);
            aPaymentRes.mC_STREET_2   = stmt.getString(sOutIdx++);
            aPaymentRes.mC_CITY       = stmt.getString(sOutIdx++);
            aPaymentRes.mC_STATE      = stmt.getString(sOutIdx++);
            aPaymentRes.mC_ZIP        = stmt.getString(sOutIdx++);
            aPaymentRes.mC_PHONE      = stmt.getString(sOutIdx++);
            aPaymentRes.mC_CREDIT     = stmt.getString(sOutIdx++);
            aPaymentRes.mC_CREDIT_LIM = stmt.getFloat(sOutIdx++);
            aPaymentRes.mC_DISCOUNT   = stmt.getFloat(sOutIdx++);
            aPaymentRes.mC_BALANCE    = stmt.getFloat(sOutIdx++);
            aPaymentRes.mC_SINCE      = stmt.getString(sOutIdx++);
            aPaymentRes.mC_DATA       = stmt.getString(sOutIdx++);
            sSuccess                  = stmt.getInt(sOutIdx++);

            if( sSuccess != 1 )
            {
                aMessage.append(stmt.getString(sOutIdx++));
            }

            stmt.close();
            conn.close();
            ctx.close();

            if( sSuccess == 1 )
            {
                    sReturn = true;
                    Common.trSuccess();
            }
            else
            {
                    sReturn = false;
                    Common.trFail();
            }
        }
        catch( Exception e )
        {
            sReturn = false;
            System.out.println( "Exception Encountered : " + e );
            Common.trFail();
        }
        finally
        {
            if( conn != null )
            {
                conn = null;
            }
        }

        return sReturn;
    }

    protected boolean printResult(PrintWriter writer,
HttpServletRequest req, StringBuilder aMessage)
    {
        int i;
        int j;
        boolean sReturn;

        PaymentReq sPaymentReq = new PaymentReq();
        PaymentRes sPaymentRes = new PaymentRes();

        sPaymentReq.mW_ID =
Integer.parseInt( req.getParameter( "W_ID" ) );
        sPaymentReq.mD_ID =
Integer.parseInt( req.getParameter( "D_ID" ) );
        sPaymentReq.mC_ID = 0;

        if( req.getParameter( "C_ID" ) != null &&
req.getParameter( "C_ID" ).isEmpty() == false )
        {
            sPaymentReq.mC_ID =
Integer.parseInt( req.getParameter( "C_ID" ) );
        }

        sPaymentReq.mC_W_ID =
Integer.parseInt( req.getParameter( "C_W_ID" ) );
        sPaymentReq.mC_D_ID =
Integer.parseInt( req.getParameter( "C_D_ID" ) );
        sPaymentReq.mC_LAST = req.getParameter( "C_LAST" );
        sPaymentReq.mH_AMOUNT =
Float.parseFloat( req.getParameter( "H_AMOUNT" ) );

        sReturn = PaymentTransaction( sPaymentReq, sPaymentRes,
aMessage );

        if( sReturn == true )
        {
            writer.println( "<pre>" );
            writer.format( "%34cPayment\n", 32 );
            writer.format( "Date: %s\n\n", sPaymentRes.mH_DATE );
            writer.format( "Warehouse: %4d%26cDistrict: %2d\n",
                           sPaymentReq.mW_ID, 32,
                           sPaymentReq.mD_ID );
            writer.format( "%-20s%21c%-20s\n",
                           sPaymentRes.mW_STREET_1, 32,
                           sPaymentRes.mD_STREET_1 );
            writer.format( "%-20s%21c%-20s\n",
                           sPaymentRes.mW_STREET_2, 32,
                           sPaymentRes.mD_STREET_2 );
            writer.format( "%-20s %2s %.5s-%4s%7c%-20s %2s %.5s-
%4s\n\n",
                           sPaymentRes.mW_CITY,
                           sPaymentRes.mW_STATE,
                           sPaymentRes.mW_ZIP,
                           sPaymentRes.mW_ZIP.substring(5), 32,
                           sPaymentRes.mD_CITY,
                           sPaymentRes.mD_STATE,
                           sPaymentRes.mD_ZIP,
                           sPaymentRes.mD_ZIP.substring(5) );
            writer.format( "Customer: %4d  Cust-Warehouse: %4d
Cust-District: %2d\n",
                           sPaymentRes.mC_ID,
                           sPaymentReq.mC_W_ID,
                           sPaymentReq.mC_D_ID );
            writer.format( "Name:   %16s %2s %-
16s%5cSince:  %.10s\n",
                           sPaymentRes.mC_FIRST,
                           sPaymentRes.mC_MIDDLE,
                           sPaymentRes.mC_LAST, 32,
                           sPaymentRes.mC_SINCE );
            writer.format( "        %-20s%21cCredit: %2s\n",
                           sPaymentRes.mC_STREET_1, 32,
                           sPaymentRes.mC_CREDIT );
            writer.format( "        %-20s%21c%%Disc:  %5.2f\n",
                           sPaymentRes.mC_STREET_2, 32,
                           sPaymentRes.mC_DISCOUNT );
            writer.format( "        %-20s %s %.5s-
%.4s%7cPhone:  %.6s-%.3s-%.3s-%.4s\n\n",
                           sPaymentRes.mC_CITY,
                           sPaymentRes.mC_STATE,
                           sPaymentRes.mC_ZIP,
                           sPaymentRes.mC_ZIP.substring(5), 32,
                           sPaymentRes.mC_PHONE,
```

```
                            sPaymentRes.mC_PHONE.substring(6),
                            sPaymentRes.mC_PHONE.substring(9),
                            sPaymentRes.mC_PHONE.substring(12) );
            writer.format( "Amount Paid:%10c$%7.2f       New-Cust-
Balance: $%+14.2f\n",
                            32, sPaymentReq.mH_AMOUNT,
                            sPaymentRes.mC_BALANCE );
            writer.format( "Credit Limit:   $%13.2f\n\n",
                            sPaymentRes.mC_CREDIT_LIM );
            writer.format( "Cust-Data: " );

            i = 0;
            if( sPaymentRes.mC_CREDIT.equals("BC") == true )
            {
                writer.format( "%s\n",
sPaymentRes.mC_DATA.substring(0, 50) );
                i++;
                int sChunks = sPaymentRes.mC_DATA.length() / 50;
                for( j = 1; j < sChunks; j++ )
                {
                    writer.format( "           %s\n",
sPaymentRes.mC_DATA.substring(j * 50, (j+1) * 50) );
                    i++;
                }
            }

            for( ; i < 6; i++ )
            {
                writer.println( "" );
            }

            writer.println( "</pre>" );
        }

        return sReturn;
    }
}
```

StockLevel.java

```
package com.sunje.tpcc;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.CallableStatement;
import java.sql.Connection;

import javax.naming.InitialContext;
import javax.sql.DataSource;

@SuppressWarnings("serial")
@WebServlet("/StockLevel")
public class StockLevel extends HttpServlet
{
    class StockLevelReq
    {
        int mW_ID;
        int mD_ID;
        int mThreshold;
    };

    class StockLevelRes
    {
        int mO_ID;
        int mStock_Count;
    };

    @Override
    protected void doGet(HttpServletRequest req,
HttpServletResponse resp) throws ServletException, IOException
    {
        StringBuilder sMessage = new StringBuilder();

        resp.setContentType( "text/html" );
        PrintWriter writer = resp.getWriter();
        writer.println( Common.PAGE_HEADER );

        if( printResult( writer, req, sMessage ) == false )
        {
            resp.sendError( HttpServletResponse.SC_BAD_REQUEST,
sMessage.toString() );
        }

        Common.printMenuBar( writer );

        writer.println( Common.PAGE_FOOTER );
        writer.close();
    }

    boolean StockLevelTransaction( StockLevelReq aStockLevelReq,
                                   StockLevelRes aStockLevelRes,
                                   StringBuilder aMessage )
    {
        DataSource ds = null;
        Connection conn = null;
        InitialContext ctx;
        int sIdx = 1;
        int sOutIdx = 1;
        boolean sReturn;
        int sSuccess;

        try
        {
            ctx = new InitialContext();
            ds = (DataSource)
ctx.lookup( "java:comp/env/jdbc/goldilocks" );
            conn = ds.getConnection();


conn.setTransactionIsolation( Connection.TRANSACTION_SERIALIZABLE );

            CallableStatement stmt;

            stmt = conn.prepareCall( "{CALL
StockLevel(?,?,?,?,?,?,?)}" );

            stmt.setInt(sIdx++, aStockLevelReq.mW_ID);
            stmt.setInt(sIdx++, aStockLevelReq.mD_ID);
            stmt.setInt(sIdx++, aStockLevelReq.mThreshold);

            sOutIdx = sIdx;
            stmt.registerOutParameter(sIdx++,
java.sql.Types.INTEGER);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.INTEGER);

            stmt.registerOutParameter(sIdx++,
java.sql.Types.INTEGER);
            stmt.registerOutParameter(sIdx++,
java.sql.Types.VARCHAR);

            // transaction..
            stmt.execute();

            aStockLevelRes.mO_ID        = stmt.getInt(sOutIdx++);
            aStockLevelRes.mStock_Count = stmt.getInt(sOutIdx++);
            sSuccess                    = stmt.getInt(sOutIdx++);

            if( sSuccess != 1 )
            {
                aMessage.append(stmt.getString(sOutIdx++));
            }

            stmt.close();
            conn.close();
            ctx.close();

            if( sSuccess == 1 )
            {
                    sReturn = true;
                    Common.trSuccess();
            }
            else
            {
                    sReturn = false;
                    Common.trFail();
            }
        }
        catch( Exception e )
        {
            sReturn = false;
            System.out.println( "Exception Encountered : " + e );
            Common.trFail();
        }
        finally
        {
            if( conn != null )
            {
                conn = null;
            }
        }

        return sReturn;
    }

    protected boolean printResult(PrintWriter writer,
HttpServletRequest req, StringBuilder aMessage)
    {
        StockLevelReq sStockLevelReq = new StockLevelReq();
        StockLevelRes sStockLevelRes = new StockLevelRes();
        boolean sReturn;
```

```
        sStockLevelReq.mW_ID =
Integer.parseInt( req.getParameter("W_ID" ) );
        sStockLevelReq.mD_ID =
Integer.parseInt( req.getParameter("D_ID" ) );
        sStockLevelReq.mThreshold =
Integer.parseInt( req.getParameter("THRESHOLD" ) );

        sReturn = StockLevelTransaction( sStockLevelReq,
sStockLevelRes, aMessage );

        if( sReturn == true )
        {
            writer.println( "<pre>" );
            writer.format( "%31cStock-Level\n", 32 );
            writer.format( "Warehouse: %4d    District: %2d\n\n",
                           sStockLevelReq.mW_ID,
                           sStockLevelReq.mD_ID );
            writer.format( "Stock Level Threshold: %2d\n\n",
                           sStockLevelReq.mThreshold );
            writer.format( "low
    stock: %3d\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n",
                           sStockLevelRes.mStock_Count );
            writer.println( "</pre>" );
          }

          return sReturn;
      }
    }
```

Appendix B: Database Design

analyze_system.sql

```
ANALYZE SYSTEM COMPUTE STATISTICS;
COMMIT;
```

analyze_table.sql

```
\set linesize 400

--##############################################
--# WAREHOUSE
--##############################################

--# result: 1 row
SELECT
       TABLE_NAME
     , NUM_ROWS
  FROM
       DICTIONARY_SCHEMA.USER_TABLES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'WAREHOUSE'
;


--# result: n rows
SELECT
       TABLE_NAME
     , COLUMN_NAME
     , NUM_DISTINCT
     , NUM_NULLS
     , LOW_VALUE
     , HIGH_VALUE
  FROM
       DICTIONARY_SCHEMA.USER_TAB_COLUMNS
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'WAREHOUSE'
;


--# result: n rows
SELECT
       TABLE_NAME
     , INDEX_NAME
     , DISTINCT_KEYS
  FROM
       DICTIONARY_SCHEMA.USER_INDEXES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'WAREHOUSE'
;



--##############################################
--# DISTRICT
--##############################################

--# result: 1 row
SELECT
       TABLE_NAME
     , NUM_ROWS
  FROM
       DICTIONARY_SCHEMA.USER_TABLES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'DISTRICT'
;


--# result: n rows
SELECT
       TABLE_NAME
     , COLUMN_NAME
     , NUM_DISTINCT
     , NUM_NULLS
     , LOW_VALUE
     , HIGH_VALUE
  FROM
       DICTIONARY_SCHEMA.USER_TAB_COLUMNS
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'DISTRICT'
;


--# result: n rows
SELECT
       TABLE_NAME
     , INDEX_NAME
     , DISTINCT_KEYS
  FROM
       DICTIONARY_SCHEMA.USER_INDEXES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'DISTRICT'
;


--##############################################
--# CUSTOMER
--##############################################

--# result: 1 row
SELECT
       TABLE_NAME
     , NUM_ROWS
  FROM
       DICTIONARY_SCHEMA.USER_TABLES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'CUSTOMER'
;


--# result: n rows
SELECT
       TABLE_NAME
     , COLUMN_NAME
     , NUM_DISTINCT
     , NUM_NULLS
     , LOW_VALUE
     , HIGH_VALUE
  FROM
       DICTIONARY_SCHEMA.USER_TAB_COLUMNS
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'CUSTOMER'
;


--# result: n rows
SELECT
       TABLE_NAME
     , INDEX_NAME
     , DISTINCT_KEYS
  FROM
       DICTIONARY_SCHEMA.USER_INDEXES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'CUSTOMER'
;


--##############################################
--# HISTORY
--##############################################

--# result: 1 row
SELECT
       TABLE_NAME
     , NUM_ROWS
  FROM
       DICTIONARY_SCHEMA.USER_TABLES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'HISTORY'
;


--# result: n rows
SELECT
       TABLE_NAME
     , COLUMN_NAME
     , NUM_DISTINCT
     , NUM_NULLS
     , LOW_VALUE
     , HIGH_VALUE
  FROM
       DICTIONARY_SCHEMA.USER_TAB_COLUMNS
```

```
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'HISTORY'
;


--# result: n rows
SELECT
       TABLE_NAME
     , INDEX_NAME
     , DISTINCT_KEYS
  FROM
       DICTIONARY_SCHEMA.USER_INDEXES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'HISTORY'
;


--##############################################
--# NEW_ORDER
--##############################################

--# result: 1 row
SELECT
       TABLE_NAME
     , NUM_ROWS
  FROM
       DICTIONARY_SCHEMA.USER_TABLES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'NEW_ORDER'
;


--# result: n rows
SELECT
       TABLE_NAME
     , COLUMN_NAME
     , NUM_DISTINCT
     , NUM_NULLS
     , LOW_VALUE
     , HIGH_VALUE
  FROM
       DICTIONARY_SCHEMA.USER_TAB_COLUMNS
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'NEW_ORDER'
;


--# result: n rows
SELECT
       TABLE_NAME
     , INDEX_NAME
     , DISTINCT_KEYS
  FROM
       DICTIONARY_SCHEMA.USER_INDEXES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'NEW_ORDER'
;


--##############################################
--# ORDERS
--##############################################

--# result: 1 row
SELECT
       TABLE_NAME
     , NUM_ROWS
  FROM
       DICTIONARY_SCHEMA.USER_TABLES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'ORDERS'
;


--# result: n rows
SELECT
       TABLE_NAME
     , COLUMN_NAME
     , NUM_DISTINCT
     , NUM_NULLS
     , LOW_VALUE
     , HIGH_VALUE
  FROM
       DICTIONARY_SCHEMA.USER_TAB_COLUMNS
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'ORDERS'
;


--# result: n rows
SELECT
       TABLE_NAME
     , INDEX_NAME
     , DISTINCT_KEYS
  FROM
       DICTIONARY_SCHEMA.USER_INDEXES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'ORDERS'
;


--##############################################
--# ORDER_LINE
--##############################################

--# result: 1 row
SELECT
       TABLE_NAME
     , NUM_ROWS
  FROM
       DICTIONARY_SCHEMA.USER_TABLES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'ORDER_LINE'
;


--# result: n rows
SELECT
       TABLE_NAME
     , COLUMN_NAME
     , NUM_DISTINCT
     , NUM_NULLS
     , LOW_VALUE
     , HIGH_VALUE
  FROM
       DICTIONARY_SCHEMA.USER_TAB_COLUMNS
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'ORDER_LINE'
;


--# result: n rows
SELECT
       TABLE_NAME
     , INDEX_NAME
     , DISTINCT_KEYS
  FROM
       DICTIONARY_SCHEMA.USER_INDEXES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'ORDER_LINE'
;


--##############################################
--# ITEM
--##############################################

--# result: 1 row
SELECT
       TABLE_NAME
     , NUM_ROWS
  FROM
       DICTIONARY_SCHEMA.USER_TABLES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'ITEM'
;


--# result: n rows
SELECT
       TABLE_NAME
     , COLUMN_NAME
     , NUM_DISTINCT
     , NUM_NULLS
     , LOW_VALUE
     , HIGH_VALUE
  FROM
       DICTIONARY_SCHEMA.USER_TAB_COLUMNS
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'ITEM'
;


--# result: n rows
SELECT
       TABLE_NAME
     , INDEX_NAME
```

```
     , DISTINCT_KEYS
  FROM
       DICTIONARY_SCHEMA.USER_INDEXES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'ITEM'
;


--#############################################
--# STOCK
--#############################################

--# result: 1 row
SELECT
       TABLE_NAME
     , NUM_ROWS
  FROM
       DICTIONARY_SCHEMA.USER_TABLES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'STOCK'
;


--# result: n rows
SELECT
       TABLE_NAME
     , COLUMN_NAME
     , NUM_DISTINCT
     , NUM_NULLS
     , LOW_VALUE
     , HIGH_VALUE
  FROM
       DICTIONARY_SCHEMA.USER_TAB_COLUMNS
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'STOCK'
;


--# result: n rows
SELECT
       TABLE_NAME
     , INDEX_NAME
     , DISTINCT_KEYS
  FROM
       DICTIONARY_SCHEMA.USER_INDEXES
 WHERE
       TABLE_SCHEMA = 'PUBLIC'
   AND TABLE_NAME   = 'STOCK'
;
```

analyze table district.sql

```
ANALYZE TABLE customer COMPUTE STATISTICS FOR COLUMNS c_id, c_d_id,
c_w_id, c_last;
ANALYZE TABLE customer COMPUTE STATISTICS FOR ALL INDEXES;
COMMIT;
```

analyze table item.sql

```
ANALYZE TABLE item COMPUTE STATISTICS FOR COLUMNS i_id;
ANALYZE TABLE item COMPUTE STATISTICS FOR ALL INDEXES;
COMMIT;
```

analyze table new order.sql

```
ANALYZE TABLE new_order COMPUTE STATISTICS FOR COLUMNS no_o_id,
no_d_id, no_w_id;
ANALYZE TABLE new_order COMPUTE STATISTICS FOR ALL INDEXES;
COMMIT;
```

analyze table order line.sql

```
ANALYZE TABLE order_line COMPUTE STATISTICS FOR COLUMNS ol_o_id,
ol_d_id, ol_w_id, ol_i_id;
ANALYZE TABLE order_line COMPUTE STATISTICS FOR ALL INDEXES;
COMMIT;
```

analyze table orders.sql

```
ANALYZE TABLE orders COMPUTE STATISTICS FOR COLUMNS o_id, o_d_id,
o_w_id, o_c_id;
ANALYZE TABLE orders COMPUTE STATISTICS FOR ALL INDEXES;
COMMIT;
```

analyze table stock.sql

```
ANALYZE TABLE stock COMPUTE STATISTICS FOR COLUMNS s_i_id, s_w_id,
s_quantity;
ANALYZE TABLE stock COMPUTE STATISTICS FOR ALL INDEXES;
COMMIT;
```

analyze table warehouse.sql

```
ANALYZE TABLE warehouse;
COMMIT;
```

audit.sql

```
DROP TABLE IF EXISTS AUDIT;

CREATE TABLE AUDIT (
    START_D TIMESTAMP
) TABLESPACE TPCC_DATA_TBS;

COMMIT;

INSERT INTO AUDIT VALUES ( systimestamp );
COMMIT;
```

count.sql

```
\set timing on
select count(w_id) from warehouse;
select count(d_w_id) from district;
select count(c_w_id) from customer;
select count(h_c_id) from history;
select count(no_w_id) from new_order;
select count(o_w_id) from orders;
select count(ol_w_id) from order_line;
select count(i_id) from item;
select count(s_w_id) from stock;
```

create index.sql

```
ALTER TABLE WAREHOUSE ADD PRIMARY KEY (W_ID) INDEX WAREHOUSE_PK_IDX
PCTFREE 15;
COMMIT;

ALTER TABLE DISTRICT ADD PRIMARY KEY (D_W_ID, D_ID) INDEX
DISTRICT_PK_IDX PCTFREE 15;
COMMIT;

ALTER TABLE CUSTOMER ADD PRIMARY KEY(C_W_ID, C_D_ID, C_ID) INDEX
CUSTOMER_PK_IDX PCTFREE 15;
COMMIT;

CREATE INDEX CUSTOMER_IDX_W_ID_D_ID_LAST ON CUSTOMER( C_W_ID,
C_D_ID, C_LAST ) PCTFREE 15;
COMMIT;

ALTER TABLE NEW_ORDER ADD PRIMARY KEY(NO_W_ID, NO_D_ID, NO_O_ID)
INDEX NEW_ORDER_PK_IDX PCTFREE 15;
COMMIT;

ALTER TABLE ORDERS ADD PRIMARY KEY(O_W_ID, O_D_ID, O_ID) INDEX
ORDERS_PK_IDX PCTFREE 15;
COMMIT;

CREATE INDEX ORDERS_IDX2 ON ORDERS( O_W_ID, O_D_ID, O_C_ID, O_ID )
PCTFREE 15;
COMMIT;

ALTER TABLE ORDER_LINE ADD PRIMARY KEY(OL_W_ID, OL_D_ID, OL_O_ID,
OL_NUMBER) INDEX ORDER_LINE_PK_IDX PCTFREE 15;
COMMIT;

ALTER TABLE ITEM ADD PRIMARY KEY(I_ID) INDEX ITEM_PK_IDX PCTFREE 15;
COMMIT;

ALTER TABLE STOCK ADD PRIMARY KEY(S_W_ID, S_I_ID) INDEX
STOCK_PK_IDX PCTFREE 15;
COMMIT;
```

create procedure.sql

```
CREATE OR REPLACE PROCEDURE NewOrder
(
    IN_mW_ID                IN   INTEGER,
    IN_mD_ID                IN   INTEGER,
    IN_mC_ID                IN   INTEGER,
    IN_mOL_CNT              IN   INTEGER,
    IN_mALL_LOCAL           IN   INTEGER,
    IN_ItemString           IN   VARCHAR(2000),
    OUT_mC_LAST             OUT  VARCHAR(16),
    OUT_mC_CREDIT           OUT  VARCHAR(2),
    OUT_mC_DISCOUNT         OUT  NUMERIC(4, 4),
    OUT_mW_TAX              OUT  NUMERIC(4, 4),
    OUT_mD_TAX              OUT  NUMERIC(4, 4),
    OUT_mO_ID               OUT  NATIVE_INTEGER,
    OUT_mO_ENTRY_D          OUT  VARCHAR(21),
    OUT_mTotalAmount        OUT  NUMERIC(15, 2),
    OUT_mItemString         OUT  VARCHAR(2000),
    OUT_mISROLLBACK         OUT  NATIVE_INTEGER,
    OUT_mSuccess            OUT  NATIVE_INTEGER,
    OUT_mMessage            OUT  VARCHAR(2000)
)
IS
    total                       NUMERIC(15, 2);
    sNativeError                INTEGER;

    sIsRollbacked               CHAR(1);
    sItemString                 VARCHAR(2000);
    NO_w_id                     INTEGER;
    NO_d_id                     INTEGER;
    NO_c_id                     INTEGER;
    NO_o_ol_cnt                 INTEGER;
    NO_o_all_local              NUMERIC(1);
    NO_c_discount               NUMERIC(4, 4);
    NO_c_last                   VARCHAR(16);
    NO_c_credit                 CHAR(2);
    NO_w_tax                    NUMERIC(4, 4);

    NO_d_tax                    NUMERIC(4, 4);
    NO_o_entry_d                VARCHAR(21);
    NO_o_id                     INTEGER;
    NO_i_name                   VARCHAR(25);
    NO_i_price                  NUMERIC(5, 2);
    NO_i_data                   VARCHAR(50);
    NO_ol_i_id                  INTEGER;
    NO_s_quantity               NUMERIC(4);
    NO_org_quantity             NUMERIC(4);

    NO_s_data                   VARCHAR(50);
    NO_ol_dist_info             CHAR(24);
    NO_ol_supply_w_id           INTEGER;
    NO_ol_amount                NUMERIC(6, 2);
    NO_ol_number                INTEGER;
    NO_ol_quantity              NUMERIC(2);
    NO_s_remote_cnt_increment   NUMERIC(8);
    NO_s_BG                     CHAR(1);

    sTotalAmount                NUMERIC(15, 2);

    sTmpTxt                     VARCHAR(4000);

    PROCEDURE SETNODATA
    (
        aMessage IN VARCHAR(2000)
    )
    IS
    BEGIN
        sNativeError := SQLCODE;
        OUT_mMessage := aMessage;
    END;

    PROCEDURE SETERR
    IS
    BEGIN
        sNativeError := SQLCODE;
        OUT_mMessage := SQLERRM;
    END;
BEGIN

    ------------------------------------------------
    -- Assign Input Param, Initialize PSM Variable
    ------------------------------------------------

    NO_w_id        := IN_mW_ID;
    NO_d_id        := IN_mD_ID;
    NO_c_id        := IN_mC_ID;
    NO_o_ol_cnt    := IN_mOL_CNT;
    NO_o_all_local := IN_mALL_LOCAL;
    NO_o_entry_d   := TO_CHAR(sysdate, 'yyyy-mm-dd hh24:mi:ss');
    sItemString    := IN_ItemString;

    << RAMP_RETRY >>

    sIsRollbacked  := 0;
    total          := 0;
    sTmpTxt        := '';
    sNativeError   := 0;

    BEGIN
        SELECT /*+ USE_NL(WAREHOUSE, CUSTOMER) INDEX( WAREHOUSE,
WAREHOUSE_PK_IDX )
                   INDEX( CUSTOMER, CUSTOMER_PK_IDX ) */
               c_discount, c_last, c_credit, w_tax
          INTO NO_c_discount,
               NO_c_last,
               NO_c_credit,
               NO_w_tax
          FROM warehouse, customer
         WHERE w_id   = NO_w_id AND
               c_w_id = w_id AND
               c_d_id = NO_d_id AND
               c_id   = NO_c_id ;
    EXCEPTION WHEN NO_DATA_FOUND THEN SETNODATA('NO DATA FOUND.
INVALID W_ID or D_ID or C_ID.');goto SQL_FINISH;
              WHEN OTHERS        THEN SETERR;goto SQL_FINISH;
    END;


    BEGIN
        UPDATE district
           SET d_next_o_id = d_next_o_id + 1
         WHERE d_id   = NO_d_id AND
               d_w_id = NO_w_id
        RETURNING OLD d_next_o_id, d_tax
        INTO NO_o_id, NO_d_tax;
    EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
    END;


    BEGIN
        INSERT INTO ORDERS (o_id, o_d_id, o_w_id, o_c_id,
                            o_entry_d, o_ol_cnt, o_all_local)
        VALUES (NO_o_id, NO_d_id, NO_w_id, NO_c_id,
TO_DATE(NO_o_entry_d,'yyyy-mm-dd hh24:mi:ss'), NO_o_ol_cnt,
NO_o_all_local);
    EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
    END;


    BEGIN
        INSERT INTO NEW_ORDER (no_o_id, no_d_id, no_w_id)
        VALUES (NO_o_id, NO_d_id, NO_w_id);
    EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
    END;


    FOR NO_ol_number IN 1 .. NO_o_ol_cnt
    LOOP
        NO_ol_i_id        := SPLIT_PART( sItemString, '|',
(NO_ol_number - 1) * 3 + 1);
        NO_ol_supply_w_id := SPLIT_PART( sItemString, '|',
(NO_ol_number - 1) * 3 + 2);
        NO_ol_quantity    := SPLIT_PART( sItemString, '|',
(NO_ol_number - 1) * 3 + 3);

        BEGIN
            SELECT
                   i_price, NVL(i_name, ' '), i_data
              INTO NO_i_price,
                   NO_i_name,
                   NO_i_data
              FROM item
             WHERE i_id = NO_ol_i_id;
        EXCEPTION WHEN NO_DATA_FOUND THEN sIsRollbacked := 1;
CONTINUE;
                  WHEN OTHERS THEN SETERR;goto SQL_FINISH;
        END;


        BEGIN
            SELECT
                   s_data,
                   s_quantity,
                   CASE NO_d_id
                       WHEN 1 THEN  s_dist_01
                       WHEN 2 THEN  s_dist_02
                       WHEN 3 THEN  s_dist_03
                       WHEN 4 THEN  s_dist_04
                       WHEN 5 THEN  s_dist_05
                       WHEN 6 THEN  s_dist_06
                       WHEN 7 THEN  s_dist_07
                       WHEN 8 THEN  s_dist_08
                       WHEN 9 THEN  s_dist_09
                       WHEN 10 THEN  s_dist_10
                       ELSE NULL
                   END,
                   CASE
                       WHEN ((POSITION( 'ORIGINAL' IN NO_i_data)
<> 0) AND (POSITION( 'ORIGINAL' IN s_data) <> 0 ) )
                       THEN 'B' ELSE 'G'
                   END,
```

```
                    CASE WHEN (s_quantity - NO_ol_quantity >=
10  )
                         THEN (s_quantity - NO_ol_quantity)
                         ELSE s_quantity - NO_ol_quantity + 91
                    END,
                    CASE WHEN (NO_ol_supply_w_id = NO_w_id ) THEN
0 ELSE 1 END,
                    (NO_ol_quantity * NO_i_price),
                    (total + (NO_ol_quantity * NO_i_price) )
            INTO
                  NO_s_data,
                  NO_org_quantity,
                  NO_ol_dist_info,
                  NO_s_BG,
                  NO_s_quantity,
                  NO_s_remote_cnt_increment,
                  NO_ol_amount,
                  total
            FROM stock
           WHERE s_i_id = NO_ol_i_id AND
                 s_w_id = NO_ol_supply_w_id
             FOR UPDATE;
       EXCEPTION WHEN NO_DATA_FOUND THEN SETNODATA('NO DATA FOUND.
INVALID S_W_ID.');goto SQL_FINISH;
                 WHEN OTHERS        THEN SETERR;goto SQL_FINISH;
       END;


       BEGIN
           UPDATE stock
              SET s_quantity   = NO_s_quantity,
                  s_ytd        = s_ytd + NO_ol_quantity,
                  s_order_cnt  = s_order_cnt + 1,
                  s_remote_cnt = s_remote_cnt +
NO_s_remote_cnt_increment
            WHERE s_i_id = NO_ol_i_id AND
                  s_w_id = NO_ol_supply_w_id ;
       EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
       END;


       BEGIN
           INSERT INTO order_line (ol_o_id, ol_d_id, ol_w_id,
                                   ol_number, ol_i_id,
                                   ol_supply_w_id, ol_quantity,
                                   ol_amount, ol_dist_info)
           VALUES (NO_o_id, NO_d_id, NO_w_id,
                   NO_ol_number, NO_ol_i_id,
                   NO_ol_supply_w_id, NO_ol_quantity,
                   NO_ol_amount, NO_ol_dist_info) ;
       EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
       END;

       ------------------------------------------------------------
       -- Make Output-Array
       ------------------------------------------------------------
       sTmpTxt := sTmpTxt         ||
                  NO_i_name       || '|' ||
                  NO_org_quantity || '|' ||
                  NO_s_BG         || '|' ||
                  NO_i_price      || '|' ||
                  NO_ol_amount    || '|';

   END LOOP;

   ------------------------------------------------------------
   -- Get Total
    ------------------------------------------------------------
   sTotalAmount := total * (1 + NO_w_tax + NO_d_tax) * (1 -
NO_c_discount);

   ------------------------------------------------------------
   -- Output Param
    ------------------------------------------------------------
   OUT_mC_LAST      := NO_c_last;
   OUT_mC_CREDIT    := NO_c_credit;
   OUT_mC_DISCOUNT  := NO_c_discount;
   OUT_mW_TAX       := NO_w_tax;
   OUT_mD_TAX       := NO_d_tax;
   OUT_mO_ID        := NO_o_id;
   OUT_mO_ENTRY_D   := NO_o_entry_d;
   OUT_mTotalAmount := sTotalAmount;
   OUT_mItemString  := sTmpTxt;
   OUT_mISROLLBACK  := sIsRollbacked;
   OUT_mSuccess     := 1;

   IF ( sIsRollbacked = 0 )
   THEN
       COMMIT;
   ELSE
       ROLLBACK;
   END IF;

   goto END_STEP;

   << SQL_FINISH >>

   ROLLBACK;

   IF ( (sNativeError = -14007 ) OR (sNativeError = -14032 ) )
   THEN
       goto RAMP_RETRY;
   END IF;

   OUT_mSuccess := 0;

   << END_STEP >>
   NULL;
END;
/

COMMIT;

CREATE OR REPLACE PROCEDURE Payment
(
    IN_mW_ID             IN INTEGER,
    IN_mD_ID             IN INTEGER,
    IN_mC_ID             IN INTEGER,
    IN_mC_W_ID           IN INTEGER,
    IN_mC_D_ID           IN INTEGER,
    IN_mC_LAST           IN VARCHAR(16),
    IN_mH_AMOUNT         IN NUMERIC(6, 2),
    OUT_mW_STREET_1      OUT VARCHAR(20),
    OUT_mW_STREET_2      OUT VARCHAR(20),
    OUT_mW_CITY          OUT VARCHAR(20),
    OUT_mW_STATE         OUT CHAR(2),
    OUT_mW_ZIP           OUT CHAR(9),
    OUT_mH_DATE          OUT VARCHAR(21),
    OUT_mD_STREET_1      OUT VARCHAR(20),
    OUT_mD_STREET_2      OUT VARCHAR(20),
    OUT_mD_CITY          OUT VARCHAR(20),
    OUT_mD_STATE         OUT CHAR(2),
    OUT_mD_ZIP           OUT CHAR(9),
    OUT_mC_ID            OUT NATIVE_INTEGER,
    OUT_mC_FIRST         OUT VARCHAR(16),
    OUT_mC_MIDDLE        OUT CHAR(2),
    OUT_mC_LAST          OUT VARCHAR(16),
    OUT_mC_STREET_1      OUT VARCHAR(20),
    OUT_mC_STREET_2      OUT VARCHAR(20),
    OUT_mC_CITY          OUT VARCHAR(20),
    OUT_mC_STATE         OUT CHAR(2),
    OUT_mC_ZIP           OUT CHAR(9),
    OUT_mC_PHONE         OUT CHAR(16),
    OUT_mC_CREDIT        OUT CHAR(2),
    OUT_mC_CREDIT_LIM    OUT NUMERIC(12, 2),
    OUT_mC_DISCOUNT      OUT NUMERIC(4, 4),
    OUT_mC_BALANCE       OUT NUMERIC(15, 2),
    OUT_mC_SINCE         OUT VARCHAR(21),
    OUT_mC_DATA          OUT VARCHAR(200),
    OUT_mSuccess         OUT NATIVE_INTEGER,
    OUT_mMessage         OUT VARCHAR(2000)
)
IS
    sNativeError      INTEGER;

    PM_w_id           INTEGER;
    PM_d_id           INTEGER;
    PM_c_id           INTEGER;
    PM_w_name         VARCHAR(10);
    PM_w_street_1     VARCHAR(20);
    PM_w_street_2     VARCHAR(20);
    PM_w_city         VARCHAR(20);
    PM_w_state        CHAR(2);
    PM_w_zip          CHAR(9);
    PM_c_d_id         INTEGER;
    PM_c_w_id         INTEGER;
    PM_c_first        VARCHAR(16);
    PM_c_middle       CHAR(2);
    PM_c_last         VARCHAR(16);
    PM_c_street_1     VARCHAR(20);
    PM_c_street_2     VARCHAR(20);
    PM_c_city         VARCHAR(20);
    PM_c_state        CHAR(2);
    PM_c_zip          CHAR(9);
    PM_c_phone        CHAR(16);
    PM_c_since        VARCHAR(21);
    PM_c_credit       CHAR(2);
    PM_c_credit_lim   NUMERIC(12, 2);
    PM_c_discount     NUMERIC(4, 4);
    PM_c_balance      NUMERIC(15, 2);
    PM_c_data         VARCHAR(200);
    PM_h_date         VARCHAR(21);
    PM_h_amount       NUMERIC(6, 2);
    PM_d_name         VARCHAR(10);
    PM_d_street_1     VARCHAR(20);
    PM_d_street_2     VARCHAR(20);
    PM_d_city         VARCHAR(20);
    PM_d_state        CHAR(2);
    PM_d_zip          CHAR(9);
    PM_namecnt        INTEGER;
```

```
    PROCEDURE SETNODATA
    (
        aMessage IN VARCHAR(2000)
    )
    IS
    BEGIN
        sNativeError := SQLCODE;
        OUT_mMessage := aMessage;
    END;

    PROCEDURE SETERR
    IS
    BEGIN
        sNativeError := SQLCODE;
        OUT_mMessage := SQLERRM;
    END;

    CURSOR C1 IS SELECT c_id, c_first, c_middle,
                        c_street_1, c_street_2, c_city, c_state,
c_zip,
                        c_phone, c_credit, c_credit_lim,
                        c_discount, c_balance,
TO_CHAR(c_since,'yyyy-mm-dd hh24:mi:ss')
                   FROM customer
                   WHERE c_w_id = PM_c_w_id AND
                         c_d_id = PM_c_d_id AND
                         c_last = PM_c_last
                   ORDER BY c_first;
BEGIN

    ------------------------------------------------
    -- Input Param, Initialize
    ------------------------------------------------
    PM_w_id      := IN_mW_ID;
    PM_d_id      := IN_mD_ID;
    PM_c_id      := IN_mC_ID;
    PM_c_w_id    := IN_mC_W_ID;
    PM_c_d_id    := IN_mC_D_ID;
    PM_c_last    := IN_mC_LAST;
    PM_h_amount  := IN_mH_AMOUNT;

    << RAMP_RETRY >>

    PM_c_data    := '';
    sNativeError := 0;

    ------------------------------------------------
    -- Get Time
    ------------------------------------------------
    PM_h_date := to_char(sysdate, 'yyyy-mm-dd hh24:mi:ss');

    BEGIN
        UPDATE warehouse
        SET w_ytd = w_ytd + PM_h_amount
        WHERE w_id = PM_w_id
        RETURNING w_street_1,
                  w_street_2,
                  w_city,
                  w_state,
                  w_zip,
                  w_name
        INTO PM_w_street_1,
             PM_w_street_2,
             PM_w_city,
             PM_w_state,
             PM_w_zip,
             PM_w_name;

        IF SQL%ROWCOUNT = 0 THEN SETNODATA('NO DATA FOUND. INVALID
W_ID.');goto SQL_FINISH;
        END IF;

    EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
    END;


    BEGIN
        UPDATE district
        SET d_ytd = d_ytd + PM_h_amount
        WHERE d_w_id = PM_w_id AND
              d_id   = PM_d_id
        RETURNING d_street_1,
                  d_street_2,
                  d_city,
                  d_state,
                  d_zip,
                  d_name
        INTO PM_d_street_1,
             PM_d_street_2,
             PM_d_city,
             PM_d_state,
             PM_d_zip,
             PM_d_name;

        IF SQL%ROWCOUNT = 0 THEN SETNODATA('NO DATA FOUND. INVALID
D_ID.');goto SQL_FINISH;
        END IF;

    EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
    END;


    IF ( PM_c_id = 0 )
    THEN
        BEGIN
            SELECT count(c_id)
            INTO PM_namecnt
            FROM customer
            WHERE c_last = PM_c_last AND
                  c_d_id = PM_c_d_id AND
                  c_w_id = PM_c_w_id ;

            IF PM_namecnt = 0 THEN SETNODATA('NO DATA FOUND.
INVALID C_W_ID or C_D_ID or C_LAST.');goto SQL_FINISH;
            END IF;

        EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
        END;


        BEGIN
            OPEN C1;

            IF ( MOD( PM_namecnt, 2) <> 0 ) THEN PM_namecnt :=
PM_namecnt + 1;
            END IF;

            FOR i IN 0 .. ( (PM_namecnt / 2 ) - 1)
            LOOP
                FETCH c1 INTO PM_c_id,
                              PM_c_first,
                              PM_c_middle,
                              PM_c_street_1,
                              PM_c_street_2,
                              PM_c_city,
                              PM_c_state,
                              PM_c_zip,
                              PM_c_phone,
                              PM_c_credit,
                              PM_c_credit_lim,
                              PM_c_discount,
                              PM_c_balance,
                              PM_c_since;
                EXIT WHEN C1%NOTFOUND;
            END LOOP;

            CLOSE C1;
        EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
        END;
    ELSE
        BEGIN
            SELECT c_first, c_middle, c_last,
                   c_street_1, c_street_2, c_city, c_state, c_zip,
                   c_phone, c_credit, c_credit_lim,
                   c_discount, c_balance, TO_CHAR(c_since,'yyyy-mm-
dd hh24:mi:ss')
            INTO  PM_c_first,
                  PM_c_middle,
                  PM_c_last,
                  PM_c_street_1,
                  PM_c_street_2,
                  PM_c_city,
                  PM_c_state,
                  PM_c_zip,
                  PM_c_phone,
                  PM_c_credit,
                  PM_c_credit_lim,
                  PM_c_discount,
                  PM_c_balance,
                  PM_c_since
            FROM customer
            WHERE c_w_id = PM_c_w_id AND
                  c_d_id = PM_c_d_id AND
                  c_id   = PM_c_id ;
        EXCEPTION WHEN NO_DATA_FOUND THEN SETNODATA('NO DATA FOUND.
INVALID C_W_ID or C_D_ID or C_ID.');goto SQL_FINISH;
                  WHEN OTHERS        THEN SETERR;goto SQL_FINISH;
        END;
    END IF;


    IF ( POSITION ( 'BC' IN PM_c_credit ) <> 0 )
    THEN

        BEGIN
            UPDATE customer
            SET c_balance = c_balance - PM_h_amount,
                c_ytd_payment = c_ytd_payment + PM_h_amount,
                c_payment_cnt = c_payment_cnt + 1,
```

```
            c_data = SUBSTR((TO_CHAR(PM_c_id) || ' ' ||
                             TO_CHAR(PM_c_d_id) || ' ' ||
                             TO_CHAR(PM_c_w_id) || ' ' ||
                             TO_CHAR(PM_d_id) || ' ' ||
                             TO_CHAR(PM_w_id) || ' ' ||
                             TO_CHAR(PM_h_amount, '999.99') ||
' | ' )
                             || c_data, 1, 500)
        WHERE c_w_id = PM_c_w_id AND
              c_d_id = PM_c_d_id AND
              c_id   = PM_c_id
        RETURNING SUBSTR(c_data, 1, 200)
        INTO PM_c_data;

        IF SQL%ROWCOUNT = 0 THEN SETERR;goto SQL_FINISH;
        END IF;

    EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
    END;
ELSE
    BEGIN
        UPDATE customer
        SET c_balance = c_balance - PM_h_amount,
            c_ytd_payment = c_ytd_payment + PM_h_amount,
            c_payment_cnt = c_payment_cnt + 1
        WHERE c_w_id = PM_c_w_id AND
              c_d_id = PM_c_d_id AND
              c_id   = PM_c_id;

        IF SQL%ROWCOUNT = 0 THEN SETERR;goto SQL_FINISH;
        END IF;

    EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
    END;
END IF;

BEGIN
    INSERT INTO history
                (h_c_d_id, h_c_w_id, h_c_id, h_d_id,
                 h_w_id, h_date, h_amount, h_data )
    VALUES (PM_c_d_id,
            PM_c_w_id,
            PM_c_id,
            PM_d_id,
            PM_w_id,
            TO_DATE(PM_h_date,'yyyy-mm-dd hh24:mi:ss'),
            PM_h_amount,
            PM_w_name || '    ' || PM_d_name);
EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
END;


-------------------------------------------------
-- Output Param
-------------------------------------------------
OUT_mW_STREET_1   := PM_w_STREET_1;
OUT_mW_STREET_2   := PM_w_STREET_2;
OUT_mW_CITY       := PM_w_CITY;
OUT_mW_STATE      := PM_w_STATE;
OUT_mW_ZIP        := PM_w_ZIP;
OUT_mH_DATE       := PM_h_date;
OUT_mD_STREET_1   := PM_d_STREET_1;
OUT_mD_STREET_2   := PM_d_STREET_2;
OUT_mD_CITY       := PM_d_CITY;
OUT_mD_STATE      := PM_d_STATE;
OUT_mD_ZIP        := PM_d_ZIP;
OUT_mC_ID         := PM_c_id;
OUT_mC_FIRST      := PM_c_first;
OUT_mC_MIDDLE     := PM_c_middle;
OUT_mC_LAST       := PM_c_last;
OUT_mC_STREET_1   := PM_c_street_1;
OUT_mC_STREET_2   := PM_c_street_2;
OUT_mC_CITY       := PM_c_city;
OUT_mC_STATE      := PM_c_state;
OUT_mC_ZIP        := PM_c_zip;
OUT_mC_PHONE      := PM_c_phone;
OUT_mC_CREDIT     := PM_c_credit;
OUT_mC_CREDIT_LIM := PM_c_credit_lim;
OUT_mC_DISCOUNT   := PM_c_discount;
OUT_mC_BALANCE    := PM_c_balance;
OUT_mC_SINCE      := PM_c_since;
OUT_mC_DATA       := PM_c_data;
OUT_mSuccess      := 1;

COMMIT;

goto END_STEP;

<< SQL_FINISH >>

IF C1%ISOPEN IS TRUE
THEN
    CLOSE c1;
END IF;

    ROLLBACK;

    IF ( sNativeError = -14007 OR sNativeError = -14032 )
    THEN
        goto RAMP_RETRY;
    END IF;

    OUT_mSuccess := 0;

    << END_STEP >>
    NULL;
END;
/

COMMIT;

CREATE OR REPLACE PROCEDURE OrderStatus
(
    IN_mW_ID                 IN     INTEGER,
    IN_mD_ID                 IN     INTEGER,
    IN_mC_ID                 IN     INTEGER,
    IN_mC_LAST               IN     VARCHAR(16),
    OUT_mO_ID               OUT     NATIVE_INTEGER,
    OUT_mC_ID               OUT     NATIVE_INTEGER,
    OUT_mC_FIRST            OUT     VARCHAR(16),
    OUT_mC_MIDDLE           OUT     CHAR(2),
    OUT_mC_LAST             OUT     VARCHAR(16),
    OUT_mO_ENTRY_D          OUT     VARCHAR(21),
    OUT_mC_BALANCE          OUT     NUMERIC(15, 2),
    OUT_mO_CARRIER_ID       OUT     NATIVE_INTEGER,
    OUT_mO_OL_COUNT         OUT     NATIVE_INTEGER,
    OUT_mItemRes            OUT     VARCHAR(4000),
    OUT_mSuccess            OUT     NATIVE_INTEGER,
    OUT_mMessage            OUT     VARCHAR(2000)
)
IS
    sCount                  INTEGER;
    sNativeError            INTEGER;

    OS_w_id   INTEGER;
    OS_d_id   INTEGER;
    OS_c_id   INTEGER;

    OS_c_d_id    INTEGER;
    OS_c_w_id    INTEGER;
    OS_c_first   VARCHAR(16);
    OS_c_middle  CHAR(2);
    OS_c_last    VARCHAR(16);
    OS_c_balance NUMERIC(15, 2);
    OS_o_id      INTEGER;

    OS_o_entry_d          VARCHAR(21);
    OS_o_carrier_id       INTEGER;
    OS_ol_i_id            INTEGER;
    OS_ol_supply_w_id     INTEGER;
    OS_ol_quantity        INTEGER;
    OS_ol_amount          NUMERIC(6, 2);
    OS_ol_delivery_d      VARCHAR(21);
    OS_namecnt            INTEGER;

    sOutput  VARCHAR(4000);

    PROCEDURE SETNODATA
    (
        aMessage IN VARCHAR(2000)
    )
    IS
    BEGIN
        sNativeError := SQLCODE;
        OUT_mMessage := aMessage;
    END;

    PROCEDURE SETERR
    IS
    BEGIN
        sNativeError := SQLCODE;
        OUT_mMessage := SQLERRM;
    END;

    CURSOR c1 IS SELECT c_balance, c_first, c_middle, c_id
                   FROM customer
                  WHERE c_last = OS_c_last AND
                        c_d_id = OS_d_id AND
                        c_w_id = OS_w_id
                  ORDER BY c_first;

    CURSOR c2 IS SELECT ol_i_id, ol_supply_w_id, ol_quantity,
                        ol_amount, TO_CHAR(ol_delivery_d, 'yyyy-mm-
dd')
                   FROM order_line
                  WHERE ol_o_id = OS_o_id AND
                        ol_d_id = OS_d_id AND
                        ol_w_id = OS_w_id;
BEGIN
    -------------------------------------------------
```

```
    -- Assign Input Param, Initialize PSM Variable.
    ------------------------------------------------
    OS_w_id      := IN_mW_ID;
    OS_d_id      := IN_mD_ID;
    OS_c_w_id    := IN_mW_ID;
    OS_c_d_id    := IN_mD_ID;
    OS_c_id      := IN_mC_ID;
    OS_c_last    := IN_mC_LAST;

    << RAMP_RETRY >>

    sCount       := 0;
    sNativeError := 0;
    sOutput      := '';

    IF OS_c_id = 0
    THEN
        BEGIN
            SELECT count(c_id)
            INTO OS_namecnt
            FROM customer
            WHERE c_last = OS_c_last AND
                  c_d_id = OS_d_id AND
                  c_w_id = OS_w_id ;

            IF OS_namecnt = 0 THEN SETNODATA('NO DATA FOUND.
INVALID W_ID or D_ID or C_LAST.');goto SQL_FINISH;
            END IF;

        EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
        END;


        BEGIN
            OPEN c1;

            IF ( MOD( OS_namecnt, 2) <> 0 ) THEN OS_namecnt :=
OS_namecnt + 1;
            END IF;


            FOR i IN 0 .. ( ( OS_namecnt / 2 ) - 1 )
            LOOP
                FETCH c1 INTO OS_c_balance, OS_c_first, OS_c_middle,
OS_c_id;
                EXIT WHEN C1%NOTFOUND;
            END LOOP;


            CLOSE c1;
        EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
        END;
    ELSE
        BEGIN
            SELECT c_balance, c_first, c_middle, c_last
              INTO OS_c_balance,
                   OS_c_first,
                   OS_c_middle,
                   OS_c_last
             FROM customer
            WHERE c_id   = OS_c_id AND
                  c_d_id = OS_d_id AND
                  c_w_id = OS_w_id ;

        EXCEPTION WHEN NO_DATA_FOUND THEN SETNODATA('NO DATA FOUND.
INVALID W_ID or D_ID or C_ID.');goto SQL_FINISH;
                  WHEN OTHERS        THEN SETERR;goto SQL_FINISH;
        END;

    END IF;


    BEGIN
        SELECT /*+ INDEX_DESC(ORDERS, ORDERS_IDX2) */
               o_id,
               CASE WHEN o_carrier_id IS NULL THEN 0 ELSE
o_carrier_id END,
               TO_CHAR(o_entry_d, 'yyyy-mm-dd hh24:mi:ss')
        INTO  OS_o_id, OS_o_carrier_id, OS_o_entry_d
        FROM orders
        WHERE o_w_id = OS_c_w_id AND
              o_d_id = OS_d_id AND
              o_c_id = OS_c_id
        FETCH 1;
    EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
    END;


    BEGIN
        OPEN C2;

        LOOP
            FETCH C2 INTO OS_ol_i_id, OS_ol_supply_w_id,
OS_ol_quantity, OS_ol_amount, OS_ol_delivery_d;
            EXIT WHEN C2%NOTFOUND;


            IF ( OS_ol_delivery_d IS NULL ) THEN sOutput := sOutput
||

    OS_ol_supply_w_id  || '|' ||

    OS_ol_i_id         || '|' ||

    OS_ol_quantity     || '|' ||

    OS_ol_amount       || '|' || '|';
                                                ELSE sOutput := sOutput
||

    OS_ol_supply_w_id  || '|' ||

    OS_ol_i_id         || '|' ||

    OS_ol_quantity     || '|' ||

    OS_ol_amount       || '|' ||

    OS_ol_delivery_d   || '|';
            END IF;

            sCount := sCount + 1;

        END LOOP;

        CLOSE C2;
    EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
    END;


    ------------------------------------------------
    -- Output
    ------------------------------------------------
    OUT_mO_ID         := OS_o_id;
    OUT_mC_ID         := OS_c_id;
    OUT_mC_FIRST      := OS_c_first;
    OUT_mC_MIDDLE     := OS_c_middle;
    OUT_mC_LAST       := OS_c_last;
    OUT_mO_ENTRY_D    := OS_o_entry_d;
    OUT_mC_BALANCE    := OS_c_balance;
    OUT_mO_CARRIER_ID := OS_o_carrier_id;
    OUT_mO_OL_COUNT   := sCount;
    OUT_mItemRes      := sOutput;
    OUT_mSuccess      := 1;

    COMMIT;

    goto END_STEP;

    << SQL_FINISH >>

    IF C1%ISOPEN = TRUE
    THEN
        CLOSE c1;
    END IF;

    IF C2%ISOPEN = TRUE
    THEN
        CLOSE c2;
    END IF;

    ROLLBACK;

    IF( (sNativeError = -14007 ) OR (sNativeError = -14032 ) )
    THEN
        goto RAMP_RETRY;
    END IF;

    OUT_mSuccess := 0;

    << END_STEP >>
    NULL;
END;
/

COMMIT;

CREATE OR REPLACE PROCEDURE Delivery
(
    IN_mW_ID           IN  INTEGER,
    IN_mO_CARRIER_ID   IN  INTEGER,
    OUT_Result         OUT VARCHAR(4000),
    OUT_mSuccess       OUT NATIVE_INTEGER
)
IS
    sNativeError     INTEGER;

    DV_w_id          INTEGER;
    DV_o_carrier_id  INTEGER;
    DV_d_id          INTEGER;
```

```
    DV_c_id          INTEGER;
    DV_no_o_id       INTEGER;
    DV_o_total       NUMERIC(15,2);

    sBuffer          VARCHAR(4000);

    PROCEDURE SETERR
    IS
    BEGIN
        sNativeError := SQLCODE;
    END;
BEGIN

    ------------------------------------------
    -- INPUT ARGS, Initialize Variable
    ------------------------------------------
    DV_w_id         := IN_mW_ID;
    DV_o_carrier_id := IN_mO_CARRIER_ID;

    << RAMP_RETRY >>

    sNativeError    := 0;
    sBuffer         := '';

    ------------------------------------------
    -- GET OUTPUT
    ------------------------------------------
    FOR DV_d_id IN 1 .. 10
    LOOP
        BEGIN
            SELECT no_o_id INTO DV_no_o_id
            FROM new_order
            WHERE no_w_id = DV_w_id AND
                  no_d_id = DV_d_id
            ORDER BY no_o_id
            FETCH 1 ;
        EXCEPTION WHEN NO_DATA_FOUND THEN sBuffer := sBuffer || -1
|| '|' || DV_d_id || '|';
                                          CONTINUE;
                  WHEN OTHERS THEN SETERR;goto SQL_FINISH;
        END;

        BEGIN
            DELETE FROM new_order
            WHERE no_o_id = DV_no_o_id AND
                  no_d_id = DV_d_id AND
                  no_w_id = DV_w_id;
        EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
        END;

        BEGIN
            SELECT o_c_id INTO DV_c_id
            FROM orders
            WHERE o_id   = DV_no_o_id AND
                  o_d_id = DV_d_id AND
                  o_w_id = DV_w_id;
        EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
        END;

        BEGIN
            UPDATE orders
            SET o_carrier_id = DV_o_carrier_id
            WHERE o_id   = DV_no_o_id AND
                  o_d_id = DV_d_id AND
                  o_w_id = DV_w_id ;
        EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
        END;

        BEGIN
            UPDATE order_line
            SET ol_delivery_d = sysdate
            WHERE ol_o_id = DV_no_o_id AND
                  ol_d_id = DV_d_id AND
                  ol_w_id = DV_w_id;
        EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
        END;

        BEGIN
            SELECT SUM(ol_amount) INTO DV_o_total
            FROM order_line
            WHERE ol_o_id = DV_no_o_id AND
                  ol_d_id = DV_d_id AND
                  ol_w_id = DV_w_id;
        EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
        END;

        BEGIN
            UPDATE customer
            SET c_balance = c_balance + DV_o_total,
                c_delivery_cnt = c_delivery_cnt + 1
            WHERE c_id   = DV_c_id AND
                  c_d_id = DV_d_id AND
                  c_w_id = DV_w_id;
        EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
        END;

            ------------------------------------------
            -- Make Output-String
            ------------------------------------------
            sBuffer := sBuffer || DV_no_o_id || '|' || DV_d_id || '|';
        END LOOP;


        ------------------------------------------
        -- Output Param
        ------------------------------------------
        OUT_Result   := sBuffer;
        OUT_mSuccess := 1;

        COMMIT;

        goto END_STEP;

        << SQL_FINISH >>

        ROLLBACK;

        IF ( (sNativeError = -14007) OR (sNativeError = -14032) )
        THEN
            goto RAMP_RETRY;
        END IF;

        OUT_mSuccess := 0;

        << END_STEP >>
        NULL;
    END;
    /

    COMMIT;

    CREATE OR REPLACE PROCEDURE StockLevel
    (
        IN_mW_ID            IN   INTEGER,
        IN_mD_ID            IN   INTEGER,
        IN_mThreshold       IN   INTEGER,
        OUT_O_ID           OUT   NATIVE_INTEGER,
        OUT_STOCK_COUNT    OUT   NATIVE_INTEGER,
        OUT_mSuccess       OUT   NATIVE_INTEGER,
        OUT_mMessage       OUT   VARCHAR(2000)
    )
    IS
        sNativeError       INTEGER;

        SL_w_id            INTEGER;
        SL_d_id            INTEGER;
        SL_threshold       INTEGER;
        SL_o_id            INTEGER;
        SL_stock_count     INTEGER;

        PROCEDURE SETNODATA
        (
            aMessage IN VARCHAR(2000)
        )
        IS
        BEGIN
            sNativeError := SQLCODE;
            OUT_mMessage := aMessage;
        END;

        PROCEDURE SETERR
        IS
        BEGIN
            sNativeError := SQLCODE;
            OUT_mMessage := SQLERRM;
        END;
    BEGIN

        ------------------------------------------------
        -- Assign Input Param, Initialize PSM Variable
        ------------------------------------------------
        SL_w_id      := IN_mW_ID;
        SL_d_id      := IN_mD_ID;
        SL_threshold := IN_mThreshold;

        << RAMP_RETRY >>

        sNativeError := 0;

        BEGIN
            SELECT d_next_o_id
              INTO SL_o_id
              FROM district
             WHERE d_w_id = SL_w_id AND
                   d_id   = SL_d_id;
        EXCEPTION WHEN NO_DATA_FOUND THEN SETNODATA('NO DATA FOUND.
    INVALID W_ID or D_ID.');goto SQL_FINISH;
                  WHEN OTHERS        THEN SETERR;goto SQL_FINISH;
        END;
```

```
    BEGIN
        SELECT /*+ USE_NL(ORDER_LINE, STOCK) INDEX(ORDER_LINE,
ORDER_LINE_PK_IDX) INDEX(STOCK, STOCK_PK_IDX) */
               COUNT(DISTINCT(s_i_id))
          INTO SL_stock_count
          FROM order_line, stock
         WHERE ol_w_id = SL_w_id AND
               ol_d_id = SL_d_id AND
               ol_o_id < SL_o_id AND
               ol_o_id >= SL_o_id - 20 AND
               s_w_id = SL_w_id AND
               s_i_id = ol_i_id AND
               s_quantity < SL_threshold ;
    EXCEPTION WHEN OTHERS THEN SETERR;goto SQL_FINISH;
    END;


    ------------------------------------------------
    -- Assign Output Param
    ------------------------------------------------
    OUT_O_ID        := SL_o_id;
    OUT_STOCK_COUNT := SL_stock_count;
    OUT_mSuccess    := 1;

    COMMIT;

    goto END_STEP;

    << SQL_FINISH >>

    ROLLBACK;

    IF ( (sNativeError = -14007) OR (sNativeError = -14032) )
    THEN
        goto RAMP_RETRY;
    END IF;

    OUT_mSuccess := 0;

    << END_STEP >>
    NULL;

END;
/

COMMIT;
```

create_table.sql

```
DROP TABLE IF EXISTS WAREHOUSE;

CREATE TABLE WAREHOUSE
(
    W_ID       INTEGER,
    W_NAME     VARCHAR(10),
    W_STREET_1 VARCHAR(20),
    W_STREET_2 VARCHAR(20),
    W_CITY     VARCHAR(20),
    W_STATE    CHAR(2),
    W_ZIP      CHAR(9),
    W_TAX      NUMERIC(4,4),
    W_YTD      NUMERIC(15,2)
) PCTFREE 98 TABLESPACE TPCC_DATA_TBS;

COMMIT;

DROP TABLE IF EXISTS DISTRICT;

CREATE TABLE DISTRICT (
    D_ID        INTEGER,
    D_W_ID      INTEGER,
    D_NAME      VARCHAR(10),
    D_STREET_1  VARCHAR(20),
    D_STREET_2  VARCHAR(20),
    D_CITY      VARCHAR(20),
    D_STATE     CHAR(2),
    D_ZIP       CHAR(9),
    D_TAX       NUMERIC(4,4),
    D_YTD       NUMERIC(15,2),
    D_NEXT_O_ID INTEGER
) TABLESPACE TPCC_DATA_TBS;

COMMIT;

DROP TABLE IF EXISTS CUSTOMER;

CREATE TABLE CUSTOMER
(
    C_ID           INTEGER,
    C_D_ID         INTEGER,
    C_W_ID         INTEGER,
    C_FIRST        VARCHAR(16),
    C_MIDDLE       CHAR(2),
    C_LAST         VARCHAR(16),
    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   NUMERIC(12,2),
    C_DISCOUNT     NUMERIC(4,4),
    C_BALANCE      NUMERIC(15,2),
    C_YTD_PAYMENT  NUMERIC(15,2),
    C_PAYMENT_CNT  NUMERIC(8),
    C_DELIVERY_CNT NUMERIC(8),
    C_DATA         VARCHAR(500)
) TABLESPACE TPCC_DATA_TBS;

COMMIT;

DROP TABLE IF EXISTS HISTORY;

CREATE TABLE HISTORY
(
    H_C_ID   INTEGER,
    H_C_D_ID INTEGER,
    H_C_W_ID INTEGER,
    H_D_ID   INTEGER,
    H_W_ID   INTEGER,
    H_DATE   TIMESTAMP,
    H_AMOUNT NUMERIC(6,2),
    H_DATA   VARCHAR(24)
) TABLESPACE TPCC_DATA_TBS;

COMMIT;

DROP TABLE IF EXISTS NEW_ORDER;

CREATE TABLE NEW_ORDER
(
    NO_O_ID INTEGER,
    NO_D_ID INTEGER,
    NO_W_ID INTEGER
) TABLESPACE TPCC_DATA_TBS;

COMMIT;

DROP TABLE IF EXISTS ORDERS;

CREATE TABLE ORDERS
(
    O_ID         INTEGER,
    O_D_ID       INTEGER,
    O_W_ID       INTEGER,
    O_C_ID       INTEGER,
    O_ENTRY_D    TIMESTAMP,
    O_CARRIER_ID INTEGER,
    O_OL_CNT     NUMERIC(8),
    O_ALL_LOCAL  NUMERIC(1)
) TABLESPACE TPCC_DATA_TBS;

COMMIT;

DROP TABLE IF EXISTS ORDER_LINE;

CREATE TABLE ORDER_LINE
(
    OL_O_ID        INTEGER,
    OL_D_ID        INTEGER,
    OL_W_ID        INTEGER,
    OL_NUMBER      INTEGER,
    OL_I_ID        INTEGER,
    OL_SUPPLY_W_ID INTEGER,
    OL_DELIVERY_D  TIMESTAMP,
    OL_QUANTITY    NUMERIC(2),
    OL_AMOUNT      NUMERIC(6,2),
    OL_DIST_INFO   CHAR(24)
) STORAGE( NEXT 50M ) TABLESPACE TPCC_DATA_TBS;

COMMIT;

DROP TABLE IF EXISTS ITEM;

CREATE TABLE ITEM
(
    I_ID    INTEGER,
    I_IM_ID INTEGER,
    I_NAME  VARCHAR(24),
    I_PRICE NUMERIC(5,2),
    I_DATA  VARCHAR(50)
) TABLESPACE TPCC_DATA_TBS;

COMMIT;

DROP TABLE IF EXISTS STOCK;
```

```
CREATE TABLE STOCK
(
    S_I_ID       INTEGER,
    S_W_ID       INTEGER,
    S_QUANTITY   NUMERIC(4),
    S_DIST_01    CHAR(24),
    S_DIST_02    CHAR(24),
    S_DIST_03    CHAR(24),
    S_DIST_04    CHAR(24),
    S_DIST_05    CHAR(24),
    S_DIST_06    CHAR(24),
    S_DIST_07    CHAR(24),
    S_DIST_08    CHAR(24),
    S_DIST_09    CHAR(24),
    S_DIST_10    CHAR(24),
    S_YTD        NUMERIC(12),
    S_ORDER_CNT  NUMERIC(12),
    S_REMOTE_CNT NUMERIC(8),
    S_DATA       VARCHAR(50)
) TABLESPACE TPCC_DATA_TBS;

COMMIT;
```

create tablespace.sql

```
-- for 11000 warehouse

CREATE TABLESPACE TPCC_DATA_TBS DATAFILE
'/data/db/db2/tpcc_data_01.dbf' SIZE 15G REUSE,
'/data/db/db3/tpcc_data_02.dbf' SIZE 15G REUSE,
'/data/db/db4/tpcc_data_03.dbf' SIZE 15G REUSE,
'/data/db/db5/tpcc_data_04.dbf' SIZE 15G REUSE,
'/data/db/db1/tpcc_data_05.dbf' SIZE 15G REUSE,
'/data/db/db2/tpcc_data_06.dbf' SIZE 15G REUSE,
'/data/db/db3/tpcc_data_07.dbf' SIZE 15G REUSE,
'/data/db/db4/tpcc_data_08.dbf' SIZE 15G REUSE,
'/data/db/db5/tpcc_data_09.dbf' SIZE 15G REUSE,
'/data/db/db1/tpcc_data_10.dbf' SIZE 15G REUSE,
'/data/db/db2/tpcc_data_11.dbf' SIZE 15G REUSE,
'/data/db/db3/tpcc_data_12.dbf' SIZE 15G REUSE,
'/data/db/db4/tpcc_data_13.dbf' SIZE 15G REUSE,
'/data/db/db5/tpcc_data_14.dbf' SIZE 15G REUSE,
'/data/db/db1/tpcc_data_15.dbf' SIZE 15G REUSE,
'/data/db/db2/tpcc_data_16.dbf' SIZE 15G REUSE,
'/data/db/db3/tpcc_data_17.dbf' SIZE 15G REUSE,
'/data/db/db4/tpcc_data_18.dbf' SIZE 15G REUSE,
'/data/db/db5/tpcc_data_19.dbf' SIZE 15G REUSE,
'/data/db/db1/tpcc_data_20.dbf' SIZE 15G REUSE,
'/data/db/db2/tpcc_data_21.dbf' SIZE 15G REUSE,
'/data/db/db3/tpcc_data_22.dbf' SIZE 15G REUSE,
'/data/db/db4/tpcc_data_23.dbf' SIZE 15G REUSE,
'/data/db/db5/tpcc_data_24.dbf' SIZE 15G REUSE,
'/data/db/db1/tpcc_data_25.dbf' SIZE 15G REUSE,
'/data/db/db2/tpcc_data_26.dbf' SIZE 15G REUSE,
'/data/db/db3/tpcc_data_27.dbf' SIZE 15G REUSE,
'/data/db/db4/tpcc_data_28.dbf' SIZE 15G REUSE,
'/data/db/db5/tpcc_data_29.dbf' SIZE 15G REUSE,
'/data/db/db1/tpcc_data_30.dbf' SIZE 15G REUSE,
'/data/db/db2/tpcc_data_31.dbf' SIZE 15G REUSE,
'/data/db/db3/tpcc_data_32.dbf' SIZE 15G REUSE,
'/data/db/db4/tpcc_data_33.dbf' SIZE 15G REUSE,
'/data/db/db5/tpcc_data_34.dbf' SIZE 15G REUSE,
'/data/db/db1/tpcc_data_35.dbf' SIZE 15G REUSE,
'/data/db/db2/tpcc_data_36.dbf' SIZE 15G REUSE,
'/data/db/db3/tpcc_data_37.dbf' SIZE 15G REUSE,
'/data/db/db4/tpcc_data_38.dbf' SIZE 15G REUSE,
'/data/db/db5/tpcc_data_39.dbf' SIZE 15G REUSE,
'/data/db/db1/tpcc_data_40.dbf' SIZE 15G REUSE,
'/data/db/db2/tpcc_data_41.dbf' SIZE 15G REUSE,
'/data/db/db3/tpcc_data_42.dbf' SIZE 15G REUSE,
'/data/db/db4/tpcc_data_43.dbf' SIZE 15G REUSE,
'/data/db/db5/tpcc_data_44.dbf' SIZE 15G REUSE,
'/data/db/db1/tpcc_data_45.dbf' SIZE 15G REUSE,
'/data/db/db2/tpcc_data_46.dbf' SIZE 15G REUSE,
'/data/db/db3/tpcc_data_47.dbf' SIZE 15G REUSE,
'/data/db/db4/tpcc_data_48.dbf' SIZE 15G REUSE,
'/data/db/db5/tpcc_data_49.dbf' SIZE 15G REUSE,
'/data/db/db1/tpcc_data_50.dbf' SIZE 15G REUSE,
'/data/db/db2/tpcc_data_51.dbf' SIZE 15G REUSE,
'/data/db/db3/tpcc_data_52.dbf' SIZE 15G REUSE,
'/data/db/db4/tpcc_data_53.dbf' SIZE 15G REUSE,
'/data/db/db5/tpcc_data_54.dbf' SIZE 15G REUSE,
'/data/db/db1/tpcc_data_55.dbf' SIZE 15G REUSE,
'/data/db/db2/tpcc_data_56.dbf' SIZE 15G REUSE,
'/data/db/db3/tpcc_data_57.dbf' SIZE 15G REUSE,
'/data/db/db4/tpcc_data_58.dbf' SIZE 15G REUSE,
'/data/db/db5/tpcc_data_59.dbf' SIZE 15G REUSE,
'/data/db/db1/tpcc_data_60.dbf' SIZE 15G REUSE,
'/data/db/db2/tpcc_data_61.dbf' SIZE 15G REUSE,
'/data/db/db3/tpcc_data_62.dbf' SIZE 15G REUSE,
'/data/db/db4/tpcc_data_63.dbf' SIZE 15G REUSE,
'/data/db/db5/tpcc_data_64.dbf' SIZE 15G REUSE,
'/data/db/db1/tpcc_data_65.dbf' SIZE 15G REUSE,
'/data/db/db2/tpcc_data_66.dbf' SIZE 15G REUSE,
'/data/db/db3/tpcc_data_67.dbf' SIZE 15G REUSE,
'/data/db/db4/tpcc_data_68.dbf' SIZE 15G REUSE,
'/data/db/db5/tpcc_data_69.dbf' SIZE 15G REUSE;

ALTER TABLESPACE MEM_TEMP_TBS ADD MEMORY
'tpcc_index_01.dbf' SIZE 10G,
'tpcc_index_02.dbf' SIZE 10G,
'tpcc_index_03.dbf' SIZE 10G,
'tpcc_index_04.dbf' SIZE 10G,
'tpcc_index_05.dbf' SIZE 10G,
'tpcc_index_06.dbf' SIZE 10G,
'tpcc_index_07.dbf' SIZE 10G,
'tpcc_index_08.dbf' SIZE 10G,
'tpcc_index_09.dbf' SIZE 10G,
'tpcc_index_10.dbf' SIZE 10G,
'tpcc_index_11.dbf' SIZE 10G,
'tpcc_index_12.dbf' SIZE 10G,
'tpcc_index_13.dbf' SIZE 10G,
'tpcc_index_14.dbf' SIZE 10G,
'tpcc_index_15.dbf' SIZE 10G,
'tpcc_index_16.dbf' SIZE 10G,
'tpcc_index_17.dbf' SIZE 10G,
'tpcc_index_18.dbf' SIZE 10G,
'tpcc_index_19.dbf' SIZE 10G,
'tpcc_index_20.dbf' SIZE 10G,
'tpcc_index_21.dbf' SIZE 10G,
'tpcc_index_22.dbf' SIZE 10G,
'tpcc_index_23.dbf' SIZE 10G,
'tpcc_index_24.dbf' SIZE 10G,
'tpcc_index_25.dbf' SIZE 10G;
```

Appendix C: Tunable Parameters

goldilocks.properties.conf

```
TRANSACTION_COMMIT_WRITE_MODE = 1
TRANSACTION_TABLE_SIZE = 1024
UNDO_RELATION_COUNT = 1024
LOG_BUFFER_SIZE = 3G
LOG_FILE_SIZE = 40G
LOG_GROUP_COUNT = 5
PENDING_LOG_BUFFER_COUNT = 8
SPIN_COUNT = 1
BUSY_WAIT_COUNT = 1000
SYSTEM_TABLESPACE_DIR = '/data/db/db1'
SYSTEM_MEMORY_UNDO_TABLESPACE_SIZE = 16G
SYSTEM_MEMORY_TEMP_TABLESPACE_SIZE = 1G
SHARED_MEMORY_STATIC_SIZE = 4G
PARALLEL_IO_FACTOR = 5
PARALLEL_IO_GROUP_1 = '/data/db/db1'
PARALLEL_IO_GROUP_2 = '/data/db/db2'
PARALLEL_IO_GROUP_3 = '/data/db/db3'
PARALLEL_IO_GROUP_4 = '/data/db/db4'
PARALLEL_IO_GROUP_5 = '/data/db/db5'
#PARALLEL_IO_GROUP_6 = '/data/db/db6'
LOG_DIR = '/jsm/wal'
#LOG_DIR = '/log-disk/wal'
CLIENT_MAX_COUNT = 1024
PROCESS_MAX_COUNT = 1024
PARALLEL_LOAD_FACTOR = 16
SHARED_SESSION = NO
```

limit.conf

```
# /etc/security/limits.conf
#
#This file sets the resource limits for the users logged in via PAM.
#It does not affect resource limits of the system services.
#
#Also note that configuration files in /etc/security/limits.d
directory,
#which are read in alphabetical order, override the settings in
this
#file in case the domain is the same or more specific.
#That means for example that setting a limit for wildcard domain
here
#can be overriden with a wildcard setting in a config file in the
#subdirectory, but a user specific setting here can be overriden
only
#with a user specific setting in the subdirectory.
#
#Each line describes a limit for a user in the form:
#
#<domain>        <type>  <item>  <value>
#
#Where:
#<domain> can be:
#        - a user name
#        - a group name, with @group syntax
#        - the wildcard *, for default entry
#        - the wildcard %, can be also used with %group syntax,
#                 for maxlogin limit
#
#<type> can have the two values:
#        - "soft" for enforcing the soft limits
#        - "hard" for enforcing hard limits
#
#<item> can be one of the following:
#        - core - limits the core file size (KB)
#        - data - max data size (KB)
#        - fsize - maximum filesize (KB)
#        - memlock - max locked-in-memory address space (KB)
#        - nofile - max number of open file descriptors
#        - rss - max resident set size (KB)
#        - stack - max stack size (KB)
#        - cpu - max CPU time (MIN)
#        - nproc - max number of processes
#        - as - address space limit (KB)
#        - maxlogins - max number of logins for this user
#        - maxsyslogins - max number of logins on the system
#        - priority - the priority to run user process with
#        - locks - max number of file locks the user can hold
#        - sigpending - max number of pending signals
#        - msgqueue - max memory used by POSIX message queues
(bytes)
#        - nice - max nice priority allowed to raise to values: [-
20, 19]
#        - rtprio - max realtime priority
#
#<domain>      <type>  <item>         <value>
#

#*               soft    core            0
#*               hard    rss             10000
#@student        hard    nproc           20
#@faculty        soft    nproc           20
#@faculty        hard    nproc           50
#ftp             hard    nproc           0
#@student        -       maxlogins       4

# End of file

tpcc         soft    nofile      65536
tpcc         hard    nofile      65536

tpcc         soft    nproc       65536
tpcc         hard    nproc       65536
```

server.xml

```
<?xml version='1.0' encoding='utf-8'?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or
more
  contributor license agreements.  See the NOTICE file distributed
with
  this work for additional information regarding copyright
ownership.
  The ASF licenses this file to You under the Apache License,
Version 2.0
  (the "License"); you may not use this file except in compliance
with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing,
software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
  See the License for the specific language governing permissions
and
  limitations under the License.
-->
<!-- Note:  A "Server" is not itself a "Container", so you may not
     define subcomponents such as "Valves" at this level.
     Documentation at /docs/config/server.html
 -->
<Server port="8005" shutdown="SHUTDOWN">
  <Listener
className="org.apache.catalina.startup.VersionLoggerListener" />
  <!-- Security listener. Documentation at
/docs/config/listeners.html
  <Listener
className="org.apache.catalina.security.SecurityListener" />
  -->
  <!--APR library loader. Documentation at /docs/apr.html -->
  <Listener
className="org.apache.catalina.core.AprLifecycleListener"
SSLEngine="on" />
  <!--Initialize Jasper prior to webapps are loaded. Documentation
at /docs/jasper-howto.html -->
  <Listener className="org.apache.catalina.core.JasperListener" />
  <!-- Prevent memory leaks due to use of particular java/javax
APIs-->
  <Listener
className="org.apache.catalina.core.JreMemoryLeakPreventionListener
" />
  <Listener
className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListe
ner" />
  <Listener
className="org.apache.catalina.core.ThreadLocalLeakPreventionListen
er" />

  <!-- Global JNDI resources
       Documentation at /docs/jndi-resources-howto.html
```

```
  -->
  <GlobalNamingResources>
    <!-- Editable user database that can also be used by
         UserDatabaseRealm to authenticate users
    -->
    <Resource name="UserDatabase" auth="Container"
              type="org.apache.catalina.UserDatabase"
              description="User database that can be updated and
saved"

factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
              pathname="conf/tomcat-users.xml" />
  </GlobalNamingResources>

  <!-- A "Service" is a collection of one or more "Connectors" that
share
       a single "Container" Note:  A "Service" is not itself a
"Container",
       so you may not define subcomponents such as "Valves" at this
level.
       Documentation at /docs/config/service.html
   -->
  <Service name="Catalina">

    <!--The connectors can use a shared executor, you can define
one or more named thread pools-->
    <!--
    <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
        maxThreads="150" minSpareThreads="4"/>
    -->


    <!-- A "Connector" represents an endpoint by which requests are
received
         and responses are returned. Documentation at :
         Java HTTP Connector: /docs/config/http.html (blocking &
non-blocking)
         Java AJP  Connector: /docs/config/ajp.html
         APR (HTTP/AJP) Connector: /docs/apr.html
         Define a non-SSL HTTP/1.1 Connector on port 8080
    -->
    <Connector port="8080"
        acceptCount="50000"
        maxConnections="41000"
        maxThreads="40000"
        connectionTimeout="20000000"
        maxKeepAliveRequests="-1" keepAliveTimeout="-1"
        protocol="org.apache.coyote.http11.Http11NioProtocol"
redirectPort="8443"/>
    <!-- A "Connector" using the shared thread pool-->
    <!--
    <Connector executor="tomcatThreadPool"
               port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
    -->
    <!-- Define a SSL HTTP/1.1 Connector on port 8443
         This connector uses the BIO implementation that requires
the JSSE
         style configuration. When using the APR/native
implementation, the
         OpenSSL style configuration is required as described in
the APR/native
         documentation -->
    <!--
    <Connector port="8443"
protocol="org.apache.coyote.http11.Http11Protocol"
               maxThreads="150" SSLEnabled="true" scheme="https"
secure="true"
               clientAuth="false" sslProtocol="TLS" />
    -->

    <!-- Define an AJP 1.3 Connector on port 8009 -->
    <Connector port="8009" protocol="AJP/1.3" redirectPort="8443"
/>


    <!-- An Engine represents the entry point (within Catalina)
that processes
         every request.  The Engine implementation for Tomcat stand
alone
         analyzes the HTTP headers included with the request, and
passes them
         on to the appropriate Host (virtual host).
         Documentation at /docs/config/engine.html -->

    <!-- You should set jvmRoute to support load-balancing via AJP
ie :
    <Engine name="Catalina" defaultHost="localhost"
jvmRoute="jvm1">
    -->
    <Engine name="Catalina" defaultHost="localhost">

      <!--For clustering, please take a look at documentation at:
          /docs/cluster-howto.html  (simple how to)
          /docs/config/cluster.html (reference documentation) -->
      <!--
      <Cluster
className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
      -->

      <!-- Use the LockOutRealm to prevent attempts to guess user
passwords
           via a brute-force attack -->
      <Realm className="org.apache.catalina.realm.LockOutRealm">
        <!-- This Realm uses the UserDatabase configured in the
global JNDI
             resources under the key "UserDatabase".  Any edits
             that are performed against this UserDatabase are
immediately
             available for use by the Realm.  -->
        <Realm
className="org.apache.catalina.realm.UserDatabaseRealm"
               resourceName="UserDatabase"/>
      </Realm>

      <Host name="localhost"  appBase="webapps"
            unpackWARs="true" autoDeploy="true">

        <!-- SingleSignOn valve, share authentication between web
applications
             Documentation at: /docs/config/valve.html -->
        <!--
        <Valve
className="org.apache.catalina.authenticator.SingleSignOn" />
        -->

        <!-- Access log processes all example.
             Documentation at: /docs/config/valve.html
             Note: The pattern used is equivalent to using
pattern="common" -->
        <!--Valve
className="org.apache.catalina.valves.AccessLogValve"
directory="logs"
               prefix="localhost_access_log." suffix=".txt"
               pattern="%h %l %u %t &amp;quot;%r&amp;quot; %s %b" /-->

      </Host>
    </Engine>
  </Service>
</Server>
```

Sysctl be.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 maxmimum size of a mesage 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 = 68719476736

# Controls the maximum number of shared memory segments, in pages
kernel.shmall = 4294967296

vm.swappiness = 60
```

```
#vm.dirty_background_ratio = 90
```

Sysctl fe.conf

```
# sysctl settings are defined through files in
# /usr/lib/sysctl.d/, /run/sysctl.d/, and /etc/sysctl.d/.
#
# Vendors settings live in /usr/lib/sysctl.d/.
# To override a whole file, create a new file with the same in
# /etc/sysctl.d/ and put new settings there. To override
# only specific settings, add a file with a lexically later
# name in /etc/sysctl.d/ and put new settings there.
#
# For more information, see sysctl.conf(5) and sysctl.d(5).
net.core.netdev_max_backlog = 65535
net.core.somaxconn = 65535
net.ipv4.tcp_tw_reuse = 1
```

Appendix D: Price Quotations

Quotation

㈜TTA 貴中

Title : Quotation for TTA

참 조 : 박찬림 책임
견적일자 : 2017년 02월 16일
유효기간 : 견적일로부터 3개월

SUNJE SOFT
(주)선재소프트

대표이사 : 김 기 완 (인)

주 소 : 서울시 마포구 서교동
385-12 지석빌딩 201,202호
영업대표 : 사업본부 이용범 상무
전화번호 : 010-3911-5911
e-mail : yb.lee@sunjesoft.com

*** Goldilocks Standard Edition for LINUX 1식 (Intel Xeon E5 2.xGhz이상 16Cores)** (단위 : 원)

No.	Description	Unit List Price	Q'ty	Total Amount Price	Offer Price
1	Goldilocks Ver 3.1 DBMS Standard Edition	₩96,000,000	1 Set(s)	₩96,000,000	₩32,000,000
	- Query Processes Module				
	- Storage Management Module				
	Goldilocks DBMS License Fee	License Proposal Price			₩32,000,000
2	Goldilocks DBMS Implementaion & Supports	₩10,000,000	3 Set(s)	₩30,000,000	₩14,400,000
	Goldilocks Technical Supports Fee(3yr)	Support Proposal Price			₩14,400,000
Total Amount(VAT Exclude)				₩126,000,000	**₩46,400,000**

Goldilocks Total Amount (Offer Price)	**₩46,400,000**

* For Technical suports, it indicates 24 x 7 x 4 hours of support

Quotation

Project : Quotation for TTA	
Client : TTA	
H.P : 010-5110-2692	
The person in charge : Park,Chan-lim (charliepark@tta.or.kr)	

Company	TAEJIN INFOTECH
CEO	Cho, Byong-Chul
Address	2~4F, Namgyeong Building, 37 Seobinggo-dong, Yongsan-gu, Seoul, Korea
Category	Manufacturing, semiconductor system, manufacturing, SI consulting

Please refer to below

Delivery date: 4 weeks after PO	Estimate NO. : Jet-Speed-20170420-01
Warranty : 3 years	DATE : 4/20/2017
Payment terms : TT	Estimate valid : 15days from estidate date
Delivery Place : Designated place	The person in charge : Park,Jae-sung (jspark@taejin.co.kr / 010-3615-2486)

Total Price ₩ 141,530,000- (not included VAT)

IT Said One billion Fourty one million and Five hundred thirty thounsand Korea Won only (Excluding VAT)

NO.	Model	Discription	Quantity	Price(KRW)	Amount(KRW)	Note
1	**DB Server**	**Jet-speed™ HHA2212**	**1**	36,500,000	36,500,000	
	HHA2212	Jet-speed™ HHA2212 Barebone Kit with 800W Redundant PSU	1		-	
	CPU	E5-2630V3 Intel Xeon 2.4GHz 8_Core L3_20M	2			
	Memory	Memory 64GB DDR4 PC4-2133 ECC Reg	24			
	Disk	HDD SAS 300GB 15k	2			
	FC HBA	HBA QLE-2562 8Gb	1			
	RAID Controller	SAS/SATA 8ch 12G LSI MegaRAID 9361-8i	1			
	NIC	Intel X520-SR2	1			
	UTP Cabel	UTP CAT5e Ethernet Cable 1M	1			
	Power Cord	Power Cord, NICETECH, 2.5M	2			
	Keyboard	Dell KB216 Eng/Kor Keyboard	1			
	Mouse	Optical Mouse, Two Buttons, USB	1			
	Monitor	Monitor 27 inch	1			
	Maintenance	3-yrs 24x7x4hours Onsite Support Service (Server)	1	5,700,000	5,700,000	
2	**WAS Server**	**TJS104**	**3**	5,500,000	16,500,000	
	TJS104	TJS104 Barebone Kit with 800W Redundant PSU	1		-	
	CPU	E5-2630V3 Intel Xeon 2.4GHz 8_Core L3_20M	2			
	Memory	Memory 16GB DDR4 PC4-2133 ECC Reg	2			
	Disk	HDD SAS 300GB 15k	2			
	NIC	Intel X520-SR2	1			
	UTP Cabel	UTP CAT5e Ethernet Cable 1M	1			
	Power Cord	Power Cord, NICETECH, 2.5M	2			
	Maintenance	3-yrs 24x7x4hours Onsite Support Service (Server)	1	2,640,000	2,640,000	
3	**Storage**	**NGS500**	**1**	41,580,000	41,580,000	
	NGS500	NGS500 Controller with 800W Redundant PSU	1		-	
	SSD	SSD SATA 480 6G	2			
	HDD	SAS 1.2TB 10K	10			
	FC	FC 8Gb Target Port	2			
	Mgmt port	Network Management Port	1			
	FC Cable	OM3 LC-LC 5M Cable	2			
	UTP Cabel	UTP CAT5e Ethernet Cable 1M	1			
	Mgmt sw	Storage Management sw	1			
	Power Cord	Power Cord, NICETECH, 2.5M	2			
	Maintenance	3-yrs 24x7x4hours Onsite Support Service (Server)	1	9,900,000	9,900,000	
4	**Storage**	**JS2800**	**1**	23,160,000	23,160,000	
	JS2800	JS2800 Barebone kit with 800W Redundant PSU	1		-	
	DRAM-SSD	DRAM-SSD 64GB	8			
	Externel card	PCI-Express Externel Switch Card	1			
	Externel cable	PCI-Express Externel Cable	1			
	Power Cord	Power Cord, NICETECH, 2.5M	2			
	Maintenance	3-yrs 24x7x4hours Onsite Support Service (Server)	1	5,550,000	5,550,000	
Total						141,530,000
Total (not included VAT)						**₩ 141,530,000.00**

※ New solution for strengthening corporate competitiveness Jet-Speed

- Excellent procurement product registration and green technology certification enterprise
- Products: high performance server, hybrid semiconductor server, general x86 server, storage
- Differentiation technology: Hybrid semiconductor technology improves performance
- Competitiveness: securing global patent and source technology

견 적 서

주 문 처	한국정보통신기술협회 / 유재현 팀장님	등록번호	220 - 88 - 56770		
	010-5111-1272 / garcia@tta.or.kr	상 호	㈜에스유소프트	성명	박 성 수
견적일자	2017년 05월 22일	주 소	경기도 성남시 분당구 정자일로 158 (백궁프라자2 606호)		
견적조건	검수후 15일 견적번호 SU-2017051851	업 태	서비스		
유효기간	견적일로부터 1개월	종 목	소프트웨어 개발/공급 및 자문		
연 락 처	담당: 김기홍 이사, 직통전화: 031-717-1312(FAX: 02-6280-2663), k2hong@linux.co.kr				

품목	제품번호	제품명 및 제원	Qty.	Unit Price		Price
기술지원		**Linux Maintenance** 1. Item: CentOS 6.6 Server and above 2. Term: 3 years 3. Maintenance Service - Troubleshooting Support - Patch/Update Service - Security/Vunerability Support - Settings/Tuning Support 4. Management Rule - Call/E-mail Support: 24/7, 4hrs response time - Remote Support: 4hrs response time - On-Site Support: 2 times/year	4	7,200,000		28,800,000
		JBoss Maintenance 1. Item: JBoss Maintenance for CentOS 6.6 Server and above 2. Term: 3 years 3. Maintenance Service - Same as the Linux Maintenance 4. Management Rule - Same as the Linux Maintenance	3	12,600,000		37,800,000
할인						***Discount 20%***
합계						53,280,000
VAT.						5,328,000
공급가						**58,610,000**

상기와 같이 견적합니다.
2017년 5월 22일 ㈜에스유소프트

(본 견적서는 발주서 대용으로 사용가능합니다. : FAX : 02-6280-2663 으로 FAX바랍니다.)

상기와 같이 발주합니다. 발주일: 년 월 일

납품장소: 연락처:

납품요청일: 담당자: (인)

MJ네트웍

로그인 | 회원가입 | 장바구니 | 주문조회 | 마이쇼핑 | 게시판

홈 > 네트워크 장비 > 허브

HP 1420-24G-2SFP JH017A 24포트 스위치 허브

상품요약설명	10/100/1000 기가 스위치 / SFP 2포트 / 메모리 및 프로세서 1MB 플래쉬 / 크기 440 X 173 X 44mm / 무게 2.2kg
판매가	242,000원
제조사	HP
원산지	중국
상품코드	P0000JPO
수량	1
QR코드	이미지저장 | 코드URL복사
QR코드 보내기	
배송비	주문시 결제(선결제) 2,500원 (500,000원 이상 구매 시 무료)

장바구니담기 | 관심상품등록 | 추천메일보내기

쇼핑계속하기 | 바로구매하기

NAVER 네이버ID로 간편구매 네이버페이 | N Pay 구매 | 찜

[간편결제] 계좌로 결제하면 적립이 두 배!