HP ProLiant DL785 G6
using
Red Hat Enterprise Linux 5.3
and
Sybase IQ Single Application Server Edition v15.1 ESD#1

TPC Benchmark™ H
Full Disclosure Report

First Edition

February 1, 2010

First Edition - February 1, 2010

Hewlett-Packard Company, the sponsor of this benchmark test, believes that the information in this document is accurate as of the publication date. The information in this document is subject to change without notice. The sponsors assume 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 sponsors provide no warranty of the pricing information in this document.

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

All performance data contained in this report was obtained in a rigorously controlled environment. Results obtained in other operating environments may vary significantly. No warranty of system performance or price/performance is expressed or implied in this report.

Printed in U.S.A., February 1, 2010.

Overview

This report documents the methodology and results of the TPC Benchmark™ H test conducted on the HP ProLiant DL785 G6, in conformance with the requirements of the TPC Benchmark™ H Standard Specification, Revision 2.9.0. The operating system used for the benchmark was Red Hat Enterprise Linux 5.3; the DBMS was Sybase IQ Single Application Server Edition v15.1 ESD#1.

Standard and Executive Summary Statements

The pages following this preface contain the Executive Summary and Numerical Quantities Summary of the benchmark results.

Auditor

The benchmark configuration, environment and methodology used to produce and validate the test results and the pricing model used to calculate the cost per QphH was audited by Francois Raab, InfoSizing, to verify compliance with the relevant TPC specifications.

TPC Benchmark H Overview

The TPC Benchmark ™ H (TPC-H) is a decision support benchmark. It consists of a suite of business oriented ad-hoc queries and concurrent data modifications. The queries and the data populating the database have been chosen to have broad industry-wide relevance while maintaining a sufficient degree of ease of implementation. This benchmark illustrates decision support systems that

Examine large volumes of data;

Execute queries with a high degree of complexity;

Give answers to critical business questions.

TPC-H evaluates the performance of various decision support systems by the execution of sets of queries against a standard database under controlled conditions. The TPC-H queries:

Give answers to real-world business questions;

Simulate generated ad-hoc queries(e.g., via a point and click GUI interface);

Are far more complex than most OLTP transactions;

Include a rich breadth of operators and selectivity constraints;

Generate intensive activity on the part of the database server component of the system under test;

Are executed against a database complying to specific population and scaling requirements;

Are implemented with constraints derived from staying closely synchronized with an on-line production database.

The TPC-H operations are modeled as follows:

The database is continuously available 24 hours a day, 7 days a week, for ad-hoc queries from multiple end users and updates against all tables, except possibly during infrequent (e.g., once a month) maintenance sessions;

The TPC-H database tracks, possibly with some delay, the state of the OLTP database through on-going updates which batch together a number of modifications impacting some part of the decision support database;

Due to the world-wide nature of the business data stored in the TPC-H database, the queries and the updates may be executed against the database at any time, especially in relation to each other. In addition, this mix of queries and updates is subject to specific ACID requirements, since queries and updates may execute concurrently;

To achieve the optimal compromise between performance and operational requirements the database administrator can set, once and for all, the locking levels and the concurrent scheduling rules for queries and updates.

The minimum database required to run the benchmark holds business data from 10,000 suppliers. It contains almost ten million rows representing a raw storage capacity of about 1 GB. Compliant benchmark implementations may also use one of the larger permissible database populations (e.g. 1000 GB), as defined in Clause 4.1.3.

The performance metrics reported by TPC-H measure multiple aspects of the capability of the system to process queries. The TPC-H metric at the selected size (QphH@Size) is the performance metric. To be compliant with the TPC-H standard, all references to TPC-H results for a given configuration must include all required reporting components (see Clause 5.4.7). The TPC believes that comparisons of TPC-H results measured against different database sizes are misleading and discourages such comparisons.

The TPC-H database must be implemented using a commercially available database management system (DBMS), and the queries executed via an interface using dynamic SQL. The specification provides for variants of SQL, as implementers are not required to have implemented a specific SQL standard in full. TPC-D uses terminology and metrics that are similar to other benchmarks, originated by the TPC and others. Such similarity in terminology does not in any way imply that TPC-H results are comparable to other benchmarks. The only benchmark results comparable to TPC-H are other TPC-H results compliant with the same revision.

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

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

Benchmark sponsors are permitted several possible system designs, provided that they adhere to the model described in Clause 6. A full disclosure report (FDR) of the implementation details, as specified in Clause 8, must be made available along with the reported results.

General Implementation Guidelines

The purpose of TPC benchmarks is to provide relevant, objective performance data to industry users. To achieve that purpose, TPC benchmark specifications require that benchmark tests be implemented with systems, products, technologies and pricing that:

Are generally available to users;

Are relevant to the market segment that the individual TPC benchmark models or represents (e.g. TPC-H models and represents complex, high data volume, decision support environments);

Would plausibly be implemented by a significant number of users in the market segment the benchmark models or represents.

Hewlett-Packard Company does not warrant or represent that a user can or will achieve performance similar to the benchmark results contained in this report. No warranty of system performance or price/performance is expressed or implied by this report.

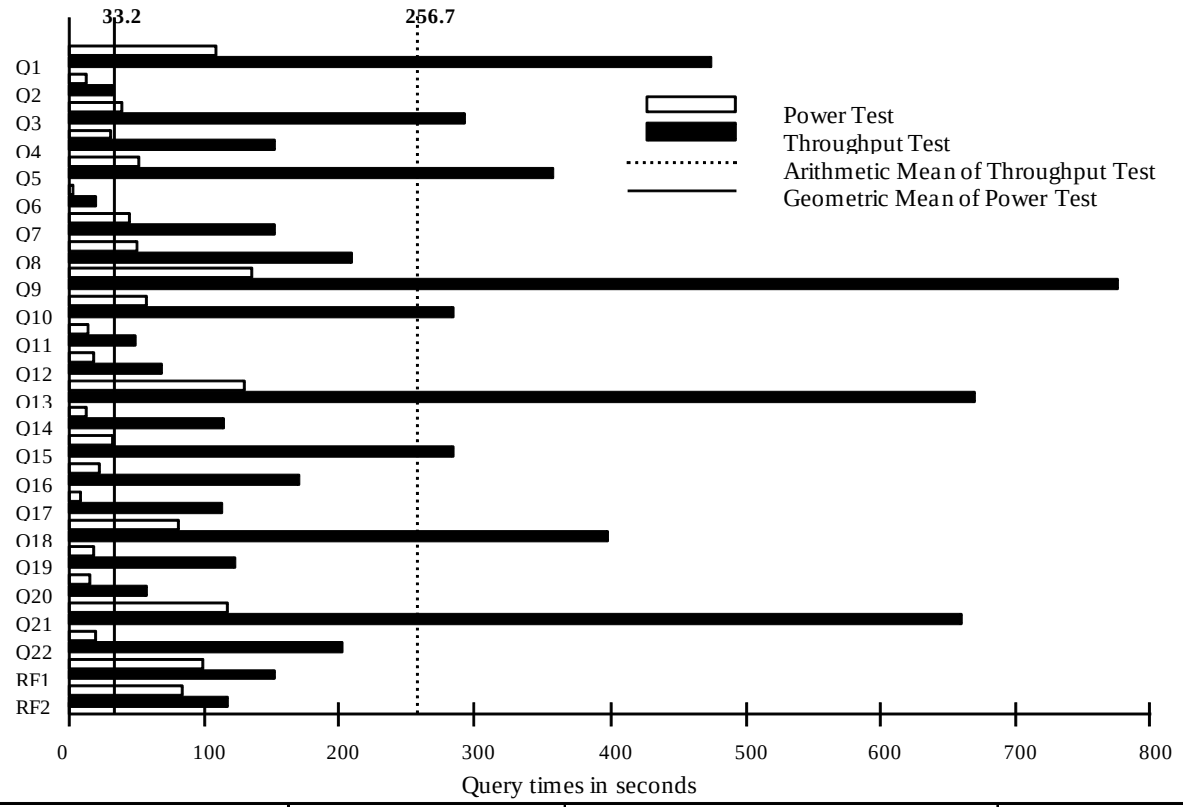
HP ProLiant DL785 G6

TPC-H Rev 2.9.0

Report Date: February 1, 2010

Total System Cost	Composite Query per Hour Metric	Price/Performance
$371,721 USD	**102,375.3** QphH@1000GB	**$3.63 USD** Price/QphH@1000GB

Database Size	Database Manager	Operating System	Other Software	Availability Date
1000 GB*	**Sybase IQ Single Application Server Edition v15.1 ESD#1**	**Red Hat Enterprise Linux 5.3**	**None**	**2/1/2010**



Database Load Time = 01:40:25	Load Includes Backup: N	Total Data Storage/Database Size = 15.18	Memory Ratio=38.4%
Storage Redundancy Level: 3	(RAID-10 base tables + RAID-10 auxilary data structures + RAID-10 other)		

System Configuration

Number of Nodes:	1
Processors/Cores/Threads/Type:	8/48/48/AMD Opteron 8439 2.8 GHz, 512 KB L3 cache per core
Memory:	384 GB
Disk Drives:	4 MSA2324 with a total of 96, 146GB disks
	8, 146GB internal disks
Total Disk Storage	15184GB (In this number one GB is defined as 1024*1024*1024 bytes)
Lan Controllers	1 Embedded Gigabit Ethernet Adapter

*Database Size includes only raw data (e.g. no temp, index, redundant storage space, etc.)

HP ProLiant DL785 G6

TPC-H Rev 2.9.0

Report Date: February 1, 2010

Description	Part Number	Source	Reference Price	Qty	Extended Price	3 yr Maint Price
Server Hardware						
HP ProLiant DL785 G6 Rack CTO Chassis	AM437A	1	9,999	1	9,999	
HP O8439SE 2.8GHz DL785 4p FIO Kit	575261-L21	1	19,499	1	19,499	
HP O8439SE 2.8GHz DL785 4p Kit	575261-B21		19,499	1	19,499	
HP 64GB Reg PC2-5300 8x8GB	495605-B21	1	7,199	6	43,194	
HP 146GB 10k 2.5 Dual Port HP SAS Drive	418367-B21	1	269	8	2,152	
HP 3year 4h 24x7 Proliant DL785 HW Support	HA104A3 Opt.6XV	1	3,208	1		3,208
HP SA P400 FIO 256MB Cache	405139-B21	1	99	1	99	
HP TFT7600 Rackmount Keyboard 17in Monitor	AG052A	1	1,679	1	1,679	
				Subtotal	**96,121**	**3,208**
Server Software						
Sybase IQ Single App Svr – per cpu core	11518	2	2,595	48	124,560	
Sybase IQ 3 Years Extended Support 24 x 7	98477	2	1,713	48		82,224
RHELAP Ultd SKT 24x7 3Year RHN Nm SW	445214-B21	1	6,747	1	6,747	Included
				Subtotal	**131,307**	**82,224**
Storage						
HP 5642 Unassembled Rack	358254-B21	1	865	1	865	
HP 82Q 8Gb Dual Port PCI-e FC HBA	AJ764A	1	2,250	8	18,000	
HP 2324fc DC Modular Smart Array	AJ797A	1	8,900	4	35,600	
MSA2000 Array Support	HA104A3 Opt. 9CJ	1	1,513	4		6,052
HP 2m Multi-mode OM2 LC/LC FC Cable	221692-B21	1	75	16	1,200	
HP 146GB 3G SAS 15K 2.5in DP ENT HDD	504062-B21	1	529	96	50,784	
				Subtotal	**106,449**	**6,052**
				Total	**333,877**	**91,484**
	Sybase discount				(18,684)	(12,334)
	16 % Large Configuration Discount and Support Prepayment*				(33,475)	(1,482)
				Grand Total	**281,718**	**90,002**

*All discounts are based on US list prices and for similar quantities and configurations

3-yr Cost of Ownership: 371,721

QphH@1000GB: 102,375

Source 1=HP, 2=Sybase

$/QphH@1000GB: 3.63

Audited By: Francois Raab for InfoSizing (www.sizing.com)

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

Measurement Results

Database Scaling (SF/size)	1000
Total Data Storage/Database Size	15.18
Percentage Memory/Database Size	38.4%
Start of Database Load Time	01/22/10 11:52:56
End of Database Load Time	01/22/10 13:33:21
Database Load Time	1:40:25
Query Streams for Throughput Test (S)	7
TPC-H Power	108,436.8
TPC-H Throughput	96,652.7
TPC-H Composite Query-per-Hour Metric (QphH@1000GB)	102,375.3
Total System Price Over 3 Years	371,721
TPC-H Price/Performance Metric ($/QphH@1000GB)	3.63

Measurement Intervals

Measurement Interval in Throughput Test (Ts)	5,736

Duration of Stream Execution:

Power Run	Seed	Query Start Time Query End Time	Duration (sec)	RF1 Start Time RF1 End Time	RF2 Start Time RF2 End Time
	122133321	01/22/10 14:01:14 01/22/10 14:18:10	1,016	01/22/10 13:59:34 01/22/10 14:01:13	01/22/10 14:18:13 01/22/10 14:19:35

Throuput Stream	Seed	Query Start Time Query End Time	Duration (sec)	RF1 Start Time RF1 End Time	RF2 Start Time RF2 End Time
1	122133322	01/22/10 14:19:35 01/22/10 15:54:14	5,679	01/22/10 14:20:39 01/22/10 14:23:56	01/22/10 14:23:56 01/22/10 14:25:40
2	122133323	01/22/10 14:19:35 01/22/10 15:53:06	5,611	01/22/10 14:25:40 01/22/10 14:28:04	01/22/10 14:28:04 01/22/10 14:29:53
3	122133324	01/22/10 14:19:35 01/22/10 15:53:09	5,614	01/22/10 14:29:53 01/22/10 14:32:10	01/22/10 14:32:11 01/22/10 14:34:26
4	122133325	01/22/10 14:19:35 01/22/10 15:54:09	5,674	01/22/10 14:34:30 01/22/10 14:37:31	01/22/10 14:37:31 01/22/10 14:39:30
5	122133326	01/22/10 14:19:35 01/22/10 15:51:57	5,542	01/22/10 14:39:30 01/22/10 14:42:11	01/22/10 14:42:12 01/22/10 14:44:37
6	122133327	01/22/10 14:19:35 01/22/10 15:55:11	5,736	01/22/10 14:44:38 01/22/10 14:46:23	01/22/10 14:46:28 01/22/10 14:47:57
7	122133328	01/22/10 14:19:35 01/22/10 15:54:15	5,680	01/22/10 14:47:58 01/22/10 14:50:14	01/22/10 14:50:14 01/22/10 14:52:10

TPC-H Timing Intervals (in seconds)

Duration of stream execution:

Stream ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Stream 00	108.4	12.4	39.3	30.5	51.5	2.6	44.9	50.2	134.5	57.1	13.2	18.2
Stream 01	347.5	26.1	379.7	94.2	351.8	8.7	142.4	212.5	742.2	248.5	44.2	81.9
Stream 02	597.9	74.9	289.4	154.9	321.0	30.6	149.0	186.9	843.4	330.0	35.3	81.0
Stream 03	503.6	30.1	184.6	159.7	412.1	4.4	215.7	258.6	912.5	264.8	68.7	67.9
Stream 04	512.3	19.9	196.2	119.0	369.2	30.3	107.6	170.4	663.6	265.7	43.2	66.7
Stream 05	446.5	23.3	341.0	221.2	281.3	33.8	208.7	148.5	720.9	260.0	27.2	72.8
Stream 06	534.4	21.3	409.8	152.3	364.6	14.9	135.2	241.1	706.5	374.3	65.2	62.3
Stream 07	381.5	27.5	243.9	157.3	398.4	7.7	105.7	241.7	832.6	240.0	52.7	45.7
Minimum	347.5	19.9	184.6	94.2	281.3	4.4	105.7	148.5	663.6	240.0	27.2	45.7
Maximum	597.9	74.9	409.8	221.2	412.1	33.8	215.7	258.6	912.5	374.3	68.7	81.9
Average	474.8	31.9	292.1	151.2	356.9	18.6	152.0	208.5	774.5	283.3	48.1	68.3

Stream ID	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream 00	128.9	12.3	32.4	22.0	7.8	80.4	17.8	15.0	116.8	19.9	98.6	82.6
Stream 01	719.4	69.3	223.7	214.4	86.0	515.0	124.1	59.3	808.8	178.6	196.3	104.0
Stream 02	679.6	183.5	241.2	202.6	161.7	285.1	107.3	52.6	424.4	177.3	143.5	109.5
Stream 03	581.5	94.0	279.3	104.9	152.4	420.8	152.4	45.7	465.9	233.7	137.1	135.3
Stream 04	698.9	198.9	232.5	149.8	188.6	345.8	130.8	40.3	839.3	284.7	181.1	119.0
Stream 05	547.7	81.8	336.2	150.6	57.1	345.9	178.3	58.8	830.7	169.7	160.7	145.4
Stream 06	686.0	110.1	494.4	197.0	35.7	350.5	102.8	67.6	404.7	204.4	105.6	89.0
Stream 07	771.4	59.8	175.4	165.1	107.3	519.6	64.2	73.7	847.9	159.7	136.6	115.7
Minimum	547.7	59.8	175.4	104.9	35.7	285.1	64.2	40.3	404.7	159.7	105.6	89.0
Maximum	771.4	198.9	494.4	214.4	188.6	519.6	178.3	73.7	847.9	284.7	196.3	145.4
Average	669.2	113.9	283.2	169.2	112.7	397.5	122.8	56.9	660.2	201.2	151.6	116.8

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

Benchmark Sponsor:

Sharada Bose
Performance Manager BCS
Hewlett-Packard
Pruneridge Avenue, MS4105
94065 Cupertino, CA 95014

January 28, 2009

I verified the TPC Benchmark™ H performance of the following configuration:

Platform:	**HP ProLiant DL785 G6**
Database Manager:	**Sybase IQ Single Application Server Edition v15.1 ESD#1**
Operating System:	**Red Hat Enterprise Linux 5.3**

The results were:

CPU (Speed)	Memory	Disks	**QphH@1000GB**
HP ProLiant DL785 G6			
8 x AMD Opteron 8439 (2.8GHz, six-core)	512 KB L3/core 384 GB Main	96 x 146GB ext. 8 x 146GB int.	**102,375.3**

In my opinion, this performance result was produced in compliance with the TPC's requirements for the benchmark. The following verification items were given special attention:

- The database records were defined with the proper layout and size
- The database population was generated using DBGEN
- The database was properly scaled to 1,000GB and populated accordingly
- The compliance of the database auxiliary data structures was verified
- The database load time was correctly measured and reported

- The required ACID properties were verified and met
- The query input variables were generated by QGEN
- The query text was produced using minor modifications and no query variant
- The execution of the queries against the SF1 database produced compliant answers
- A compliant implementation specific layer was used to drive the tests
- The throughput tests involved 7 query streams
- The ratio between the longest and the shortest query was such that none of the query timings were adjusted
- The execution times for queries and refresh functions were correctly measured and reported
- The repeatability of the measured results was verified
- The required amount of database log was configured
- The system pricing was verified for major components and maintenance
- The major pages from the FDR were verified for accuracy

Additional Audit Notes:

None.

Respectfully Yours,

François Raab
President

1 General Items

1.1 Benchmark Sponsor

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

Hewlett-Packard Company is the test sponsor of this TPC Benchmark H benchmark.

1.2 Parameter Settings

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

Database Tuning Options

Optimizer/Query execution options

Query processing tool/language configuration parameters

Recovery/commit options

Consistency/locking options

Operating system and configuration parameters

Configuration parameters and options for any other software component incorporated into the pricing structure;

Compiler optimization options.

Appendix A contains the Red Hat Enterprise Linux 5.3 and Sybase IQ 15.1 parameters used in this benchmark.

1.3 Configuration Diagrams

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

Measured Configuration

- 8, 2.8 GHz AMD Opteron 8439 CPUs each with 512 KB L3 cache per core
- 384 GB Memory
- 8 PCIe Fibre Channel 8GB Adapter (dual-port) Cards
- 1 Embedded Gigabit Ethernet Adapter
- 4 HP MSA2324 (with a total of 96 146GB disks)
- 1 P400 Smart Array (with a total of 8 146GB disks)

Priced Configuration

- 8, 2.8 GHz AMD Opteron 8439 CPUs each with 512 KB L3 cache per core
- 384 GB Memory
- 8 PCIe Fibre Channel 8GB Adapter (dual-port) Cards
- 1 Embedded Gigabit Ethernet Adapter
- 4 HP MSA2324 (with a total of 96 146GB disks)
- 1 P400 Smart Array (with a total of 8 146GB disks)

Differences in Configurations

None

Measured Configuration

Server

Storage

DL785 G6

- 8 2.8 GHz AMD Opteron 8439 CPUs each with 512 KB L3 cache per core
- 384 GB Memory
- 8 PCIe Fibre Channel 8GB Adapter (dual-port) Cards
- 1 Embedded Gigabit Ethernet Adapter
- 1 P400 Smart Array (8 146GB disks)

4 HP MSA2324

- With a total of 96 HP 3G SAS 15K 2.5" dual port 146GB disks

16 Fibre Channel Connections

Priced Configuration

Server

Storage

DL785 G6

- 8 2.8 GHz AMD Opteron 8439 CPUs each with 512 KB L3 cache per core
- 384 GB Memory
- 8 PCIe Fibre Channel 8GB Adapter (dual-port) Cards
- 1 Embedded Gigabit Ethernet Adapter
- 1 P400 Smart Array (8 146GB disks)

4 HP MSA2324

- With a total of 96 HP 3G SAS 15K 2.5" dual port 146GB disks

16 Fibre Channel Connections

2 Clause 1 Logical Database Design Related Items

2.1 Database Definition Statements

Listings must be provided for all table definition statements and all other statements used to set up the test and qualification databases.

Appendix B contains the programs and scripts used to create and load the database..

2.2 Physical Organization

The physical organization of tables and indices, within the test and qualification databases, must be disclosed. If the column ordering of any table is different from that specified in Clause 1.4, it must be noted.

No record clustering or index clustering was used. Columns were reordered in the tables – please refer to the table create statements for the ordering.

2.3 Horizontal Partitioning

Horizontal partitioning of tables and rows in the test and qualification databases (see Clause 1.5.4) must be disclosed.

Horizontal partitioning was used for the LINEITEM and ORDERS tables. The details of this partitioning can be understood by examining the syntax of the table and index definition statements in Appendix B. Similar partitioning was used in the qualification database.

Section 5.2 describes the distribution of tables and logs across all media.

2.4 Replication

Any replication of physical objects must be disclosed and must conform to the requirements of Clause 1.5.6.

No replication was used.

3 Clause 2 Queries and Refresh Functions

3.1 Query Language

The query language used to implement the queries must be identified.

SQL was the query language used to implement all queries.

3.2 Verifying Method for Random Number Generation

The method of verification for the random number generation must be described unless the supplied DBGEN and QGEN were used.

TPC supplied versions 2.9.0 of DBGEN and QGEN were used for this TPC-H benchmark.

3.3 Generating Values for Substitution Parameters

The method used to generate values for substitution parameters must be disclosed. If QGEN is not used for this purpose, then the source code of any non-commercial tool used must be disclosed. If QGEN is used, the version number, release number, modification number, and patch level of QGEN must be disclosed.

QGEN version 2.9.0 was used to generate the substitution parameters.

3.4 Query Text and Output Data from Qualification Database

The executable query text used for query validation must be disclosed along with the corresponding output data generated during the execution of the query text against the qualification database. If minor modifications (see Clause 2.2.3) have been applied to any functional query definition or approved variants in order to obtain executable query text, these modifications must be disclosed and justified. The justification for a particular minor query modification can apply collectively to all queries for which it has been used. The output data for the power and throughput tests must be made available electronically upon request.

Appendix D, "Qualification Query Output", contains the output for each of the queries. The functional query definitions and variants used in this disclosure use the following minor query modifications.

1. In Q1, Q4, Q5, Q6, Q10, Q12, Q14, Q15 and Q20, the “dateadd” function is used to perform date arithmetic
2. In Q7, Q8 and Q9, the “year” function is used to extract part of a date.
3. In Q2, Q3, Q10, Q18 and Q21, the “top” function is used to restrict the number of output rows.

3.5 Query Substitution Parameters and Seeds Used

The query substitution parameters used for all performance tests must be disclosed in tabular format, along with the seeds used to generate these parameters.

Appendix E contains the seed and query substitution parameters.

3.6 Query Isolation Level

The isolation level used to run the queries must be disclosed. If the isolation level does not map closely to the levels defined in Clause 3.4, additional descriptive detail must be provided.

The queries and transactions were run with the isolation level set to repeatable read.

3.7 Source Code of Refresh Functions

The details of how the refresh functions were implemented must be disclosed (including source code of any non-commercial program used).

The refresh function is part of the implementation-specific layer/driver code included in Appendix B.

4 Clause 3 Database System Properties

4.1 ACID Properties

The ACID (Atomicity, Consistency, Isolation, and Durability) properties of transaction processing systems must be supported by the system under test during the timed portion of this benchmark. Since TPC-H is not a transaction processing benchmark, the ACID properties must be evaluated outside the timed portion of the test.

Source code for ACID test is included in Appendix C.

4.2 Atomicity

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

Completed Transaction

Perform the ACID Transaction for a randomly selected set of input data and verify that the appropriate rows have been changed in the ORDERS, LINEITEM, and HISTORY tables.

The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for a randomly selected order key.

The ACID Transaction T1 was performed using the order key from step 1.

The ACID Transaction T1 committed.

The total price and the extended price were retrieved for the same orderkey used in step 1 and step 2. It was verified that: T1.EXTENDEDPRICE = OLD.EXTENDEDPRICE + ((T1.DELTA) * (OLD.EXTENDEDPRICE/OLD.QUANTITY)), T1.TOTALPRICE = OLD.TOTALPRICE + ((T1.EXTENDEDPRICE-OLD.EXTENDEDPRICE)*(1-DISCOUNT)*(1+TAX)), and that the number of records in the history table had increased by 1.

Aborted Transaction

Perform the ACID Transaction for a randomly selected set of input data, substituting a ROLLBACK of the transaction for the COMMIT of the transaction. Verify that the appropriate rows have not been changed in the ORDERS, LINEITEM, and HISTORY tables.

The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for a randomly selected order key.

The ACID Transaction was performed using the order key from step 1. The transaction was stopped prior to the commit.

The ACID Transaction was ROLLED BACK.

The total price and the extended price were retrieved for the same orderkey used in step 1 and step 2. It was verified that the extended price and the total price were the same as in step 1. The number of records in the HISTORY table was retrieved again and verified to be the same as in step1.

4.3 Consistency

Consistency is the property of the application that requires any execution of transactions to take the database from one consistent state to another.

Consistency Test

Verify that ORDERS and LINEITEM tables are initially consistent, submit the prescribed number of ACID Transactions with randomly selected input parameters, and re-verify the consistency of the ORDERS and LINEITEM.

1. The consistency of the ORDERS and LINEITEM tables was verified based on a sample of order keys.
2. 100 ACID Transactions were submitted from each of 10 execution streams.
3. The consistency of the ORDERS and LINEITEM tables was re-verified.

4.4 Isolation

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

Read-Write Conflict with Commit

Demonstrate isolation for the read-write conflict of a read-write transaction and a read-only transaction when the read-write transaction is committed.

An ACID Transaction was started for a randomly selected O_KEY, L_KEY, and DELTA. The ACID Transaction was suspended prior to COMMIT.

An ACID Query was started for the same O_KEY used in step 1. The ACID Query completed and did not see the uncommitted changes made by the ACID Transaction.

The ACID Transaction was resumed, and COMMITTED.

The ACID Query completed. It returned the data as committed by the ACID Transaction.

Read-Write Conflict with Rollback

Demonstrate isolation for the read-write conflict of a read-write transaction and a read-only transaction when the read-write transaction is rolled back.

An ACID Transaction was started for a randomly selected O_KEY, L_KEY, and DELTA. The ACID Transaction was suspended prior to ROLLBACK.

An ACID Query was started for the same O_KEY used in step 1. The ACID Query completed and did not see the uncommitted changes made by the ACID Transaction.

The ACID Transaction was ROLLED BACK.

The ACID Query completed.

Write-Write Conflict with Commit

Demonstrate isolation for the write-write conflict of two update transactions when the first transaction is committed.

An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. The ACID transaction T1 was suspended prior to COMMIT.

Another ACID Transaction, T2, was started using the same O_KEY and L_KEY and a randomly selected DELTA.

T2 waited.

T1 was allowed to COMMIT and T2 completed.

It was verified that T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE
+(DELTA1*(T1.L_EXTENDEDPRICE/T1.L_QUANTITY))

Write-Write Conflict with Rollback

Demonstrate isolation for the write-write conflict of two update transactions when the first transaction is rolled back.

An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. The ACID transaction T1 was suspended prior to ROLLBACK.

Another ACID Transaction, T2, was started using the same O_KEY and L_KEY and a randomly selected DELTA.

T2 waited.

T1 was allowed to ROLLBACK and T2 completed.

It was verified that T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE.

Concurrent Progress of Read and Write on Different Tables

Demonstrate the ability of read and write transactions affecting different database tables to make progress concurrently.

An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. T1 was suspended prior to COMMIT.

Another transaction, T2 was started using random values for PS_PARTKEY and PS_SUPPKEY, all columns of the PARTSUPP table for which PS_PARTKEY and PS_SUPPKEY are equal are returned.

Transaction T2 completed.

T1 was allowed to COMMIT.

It was verified that the appropriate rows in the ORDER, LINEITEM, and HISTORY tables have been changed.

Read-Only Query Conflict with Update Transactions

Demonstrates that the continuous submission of arbitrary (read-only) queries against one or more tables of the database does not indefinitely delay update transactions affecting those tables from making progress.

A Transaction, T1, was started which executed Q1 against the qualification database, was started using a randomly selected DELTA.

An ACID Transaction, T2, was started for a randomly selected O_KEY, L_KEY and DELTA.

T2 completed and appropriate rows in the ORDERS, LINEITEM and HISTORY tables had been changed.

Transaction T1 completed executing Q1.

4.5 Durability

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

Three tests were run to demonstrate durability on the qualification database. Before all three tests, the consistency of the ORDERS and LINEITEM tables was verified. Next, 10 streams were started to submit 200 ACID transactions each.

Failure of a Durable Medium

Guarantee the database and committed updates are preserved across a permanent irrecoverable failure of any single durable medium containing TPC-H database tables or recovery log tables.

In this test, after each of the 10 streams had completed 100 transactions, a disk containing TPC-H tables, indexes and database transaction log data was removed from an array. Since all disks are in RAID-1/0 configuration, the streams continued to submit transactions uninterrupted. After verifying that transactions continued without errors, the system was restarted, consistency of the ORDERS and LINEITEM tables were verified, and the durability success files and HISTORY table counts were verified.

System Crash

Guarantee the database and committed updates are preserved across an instantaneous interruption (system crash/system hang) in processing which requires the system to reboot to recover.

The system crash and memory failure tests were combined. After each of the 10 streams had completed 100 transactions, the power to the system was interrupted using the "cold boot" option. A Cold Boot of the system immediately removes power from the system. This option circumvents graceful operating system shutdown features. After the system rebooted, the database was restarted and the durability success files and HISTORY table counts were verified, and the consistency of the ORDERS and LINEITEM tables was verified.

Memory Failure

Guarantee the database and committed updates are preserved across failure of all or part of memory (loss of contents).

See “System Crash”

Array Controller Failure

Durability across controller failure was demonstrated by turning off one of the arrays after each of the 10 streams had completed 100 transactions. After the database server was restarted, the durability success files and HISTORY table counts were verified, and the consistency of the ORDERS and LINEITEM tables was verified.

5 Clause 4 Scaling and Database Population

5.1 Ending Cardinality of Tables

The cardinality (e.g., the number of rows) of each table of the test database, as it existed at the completion of the database load (see clause 4.2.5) must be disclosed.

Table	Cardinality
ORDER	1,500,000,000
LINEITEM	5,999,989,709
CUSTOMER	150,000,000
PART	200,000,000
SUPPLIER	10,000,000
PARTSUPP	800,000,000
NATION	25
REGION	5

5.2 Distribution of Tables and Logs Across Media

Distribution of tables and logs across media:

Each of the 4 MSA2324 arrays (with 24 disks each) were configured with four MSA2324 vdisks comprising of 6 disks each. The vdisks were configured with RAID10 redundancy. Each MSA2324 vdisk was divided into 4 LUNs equaling a total of 16 LUNs on each MSA2324 array.

LUNs 0-3 (one from each vdisk): Sybase data (eg. tables, indexes, logs)

LUNs 4-7 (one from each vdisk): Sybase temp space

LUNs 8-11(one from each vdisk): flat-file data

LUNs 12-15 (one from each vdisk): Unused (a few used for Qualification database during the ACID/Qualification database phase of the audit)

OS root and the Sybse home directory were configured on four internal disks with RAID1 redundancy.

5.3 Database Partition

The mapping of database partitions/replications must be explicitly described.

Horizontal partitioning was used for Lineitem and Orders table (see Appendix B).

5.4 RAID Feature

Implementation may use some form of RAID to ensure high availability. If used for data, auxiliary storage (e.g. indexes) or temporary space, the level of RAID must be disclosed for each device.

RAID1/0 was used for all data.

5.5 DBGEN Modification

Any modifications to the DBGEN (see clause 4.2.1) source code must be disclosed. In the event that a program other than DBGEN was used to populate the database, it must be disclosed in its entirety.

The supplied DBGEN version 2.9.0 was not modified to generate the database population for this benchmark.

5.6 Database Load Time

The database load time for the test database (see clause 4.3) must be disclosed.

The database load time was 1:40:25.

5.7 Data Storage Ratio

The data storage ratio must be disclosed. It is computed as the ratio between the total amount of priced disk space, and the chosen test database size as defined in Clause 4.1.3.

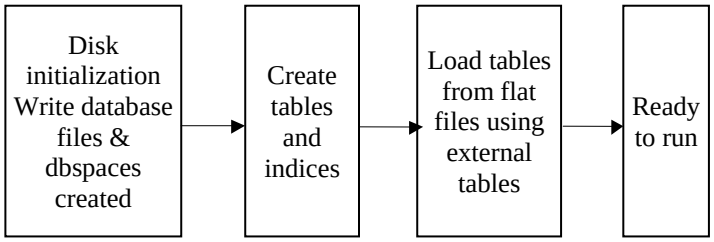
The data storage ratio is computed from the following information:

Type	# Disks	Disk Size (GB)	Total (GB)
1 P400 Smart Array	8	146	1168
4 HP MSA2324	96	146	14,016.0
TOTAL			**15,184.0**
Scale Factor			**1,000**
Storage Ratio			**15.18**

5.8 Database Load Mechanism Details and Illustration

The details of the database load must be described, including a block diagram illustrating the overall process.

The database was loaded using data generation stored on the flat files all on the tested and priced configuration



5.9 Qualification Database Configuration

Any differences between the configuration of the qualification database and the test database must be disclosed.

The qualification database used identical scripts and database options/parameters as the test database. The only difference between the test and qualification database was the fewer LUNs used in the qualification database to adjust for the scale factor.

5.10 Memory to Database Size Percentage

The memory to database size percentage, as defined in clause 8.3.5.10, must be disclosed.

The memory to database size percentage was 38%

6 Clause 5 Performance Metrics and Execution-Rules

6.1 System Activity Between Load and Performance Tests

Any system activity on the SUT that takes place between the conclusion of the load test and the beginning of the performance test must be fully disclosed.

Auditor requested queries were run against the database to verify the correctness of the database load.

All scripts and queries used are included in Appendix F.

The database server was restarted between the load and performance test.

6.2 Steps in the Power Test

The details of the steps followed to implement the power test (e.g., system boot, database restart, etc.) must be disclosed.

The following steps were used to implement the power test:

1. RF1 Refresh Transaction
2. Stream 00 Execution
3. RF2 Refresh Transaction

6.3 Timing Intervals for Each Query and Refresh Functions

The timing intervals for each query for both refresh functions must be reported for the power test.

The timing intervals for each query and both update functions are given in the Executive Summary earlier in this document.

6.4 Number of Streams for the Throughput Test

The number of execution streams used for the throughput test must be disclosed.

7 streams were used for the throughput test.

6.5 Start and End Date/Time of Each Query Stream

The start time and finish time for each query stream must be reported for the throughput test.

The throughput test start time and finish time for each stream are given in the Executive Summary earlier in this document.

6.6 Total Elapsed Time of the Measurement Interval

The total elapsed time of the measurement interval must be reported for the throughput test.

The total elapsed time of the throughput test is given in the Executive Summary earlier in this document.

6.7 Refresh Function Start Date/Time and Finish Date/Time

Start and finish time for each update function in the update stream must be reported for the throughput test.

Start and finish time for each update function in the update stream are given in the Executive Summary earlier in this document.

6.8 Timing Intervals for Each Query and Each Refresh Function for Each Stream

The timing intervals for each query of each stream and for each refresh function must be reported for the throughput test.

The timing intervals for each query and each update function are given in the Executive Summary earlier in this document.

6.9 Performance Metrics

The computed performance metric, related numerical quantities and price performance metric must be reported.

The performance metrics, and the numbers, on which they are based, is given in the Executive Summary earlier in this document.

6.10 The Performance Metric and Numerical Quantities from Both Runs

The performance metric and numerical quantities from both runs must be disclosed.

Performance results from the first two executions of the TPC-H benchmark indicated the following percent difference for the metric points:

	QppH@1000GB	QthH@1000GB	QphH@1000GB
Reported Run	108,436.8	96,652.7	102,375.3
Reproducibility Run	108,011.8	98,228.2	103,003.9
% Difference	0.4%	1.6%	0.6%

6.11 System Activity Between Performance Tests

Any activity on the SUT that takes place between the conclusion of the Reported Run and the beginning of Reproducibility Run must be disclosed.

The database was restarted between the two runs.

7 Clause 6 SUT and Driver Implementation Related Items

7.1 Driver

A detailed description of how the driver performs its functions must be supplied, including any related source code or scripts. This description should allow an independent reconstruction of the driver.

Appendix F contains the scripts used to run the power and throughput tests. The tests were invoked by calling "run_tpch" with the Scale Factor, Scope and Stream count as input parameters. For the power test, the stream 0 SQl script is started along with the refresh function stream such that:

- The SQL for RF1 is submitted and executed by the database
- Then the queries as generated by QGEN are submitted in the order defined by Clause 5.3.5.4
- The SQL for RF2 is then submitted from the same connection used for RF1 and executed by database

7.2 Implementation-Specific Layer (ISL)

If an implementation specific layer is used, then a detailed description of how it performs its functions must be provided. All related source code, scripts and configuration files must be disclosed. The information provided should be sufficient for an independent reconstruction of the implementation specific layer.

The performance tests are performed using dbisqlc and iqisql. dbisqlc and iqisql are Sybase-provided utilities which allow SQL statements to be executed against a Sybase IQ database. Both dbisqlc and iqisql utilities are invoked from the command line on the SUT. They read input from files containing SQL statements and sends results to stdout. dbisqlc uses information in the .odbc.ini file to connect to the database while iqisql uses information in interfaces file for the same.

The ACID tests are performed using dbtest. dbtest is a Sybase-provided utility, similar to dbisqlc, but providing additional scripting capabilities. It is invoked from the command-line on the SUT and uses information in the .odbc.ini file to connect to the database.

7.3 Profile-Directed Optimization

If profile-directed optimization as described in Clause 5.2. is used, such use must be disclosed..

Profile-directed optimization subject to the requirements of 5.2.9 and 5.2.10 was not used.

8 Clause 7 Pricing

8.1 Hardware and Software Used in the Priced System

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

A detailed list of hardware and software used in the priced system is included in the pricing sheet in the executive summary. All prices are currently effective.

8.2 Total 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.

A detailed pricing sheet of all the hardware and software used in this configuration and the 3-year maintenance costs, demonstrating the computation of the total 3-year price of the configuration, is included in the executive summary at the beginning of this document.

8.3 Availability Date

The committed delivery date for general availability of products used in the priced 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.

Server Hardware	Available 02/01/2010
Server Software	Available 02/01/2010
Storage	Available 02/01/2010
Sybase IQ Single Application Server Edition v15.1 ESD#1	Available 02/01/2010*

*For orderablity and pricing, contact: Anne Belt, 925-236-4108

9 Clause 8 Auditor's Information and Attestation Letter

9.1 Auditor's Report

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 implementation of the TPC Benchmark H was audited by Francois Raab for InfoSizing Further information regarding the audit process may be obtained from:

Francois Raab
InfoSizing
125 West Monroe Street
Colorado Springs, CO 80907
(719) 473-7555
(719) 473-7554(719) 473-7555

The auditor’s attestation letter is included at the front of this report.

Appendix A Parameter Settings

A.1 bm_setup

```
#!/bin/ksh
# Sybase Product Environment variables
#
if [ -z "$1" ]
then
 echo 'Usage: bm_setup <SCALE_FACTOR>'
 return 1
fi

# set prompt and terminal type
export PS1="$HOSTNAME:\w $ "
export TERM=xterm

SCALE_FACTOR=$1; export SCALE_FACTOR
VERSION=15_0; export VERSION
DATA_DIR=/flat/scale_${SCALE_FACTOR}; export
DATA_DIR

SYBASE=/sybase/IQ_151ESD1; export SYBASE
TPCH_ROOT=/sybase/tpch_audit_kit; export TPCH_ROOT

#To seperate out generated files from static files
GENERATED_FILES=$TPCH_ROOT/generated_files; export
GENERATED_FILES
CONFIG_DIR=$TPCH_ROOT/config_dir; export
CONFIG_DIR

ODBCINI=$CONFIG_DIR/.odbc.ini; export ODBCINI

DEV_DIR=$TPCH_ROOT/devices; export DEV_DIR
QUERYGEN=$TPCH_ROOT/querygen; export QUERYGEN
#QUERYGEN=$TPCH_ROOT/querygen_debug; export
QUERYGEN
STREAMS=$TPCH_ROOT/streams_${SCALE_FACTOR};
export STREAMS
QUERIES=$TPCH_ROOT/queries_${SCALE_FACTOR};
export QUERIES
RESULTS=$TPCH_ROOT/Results; export RESULTS
TEMPLATES=$TPCH_ROOT/templates; export
TEMPLATES
TEMPFILES=$TPCH_ROOT/tempfiles;export TEMPFILES
SETUP=$TPCH_ROOT/setup; export SETUP
REFERENCE_FILES=$TPCH_ROOT/dbgen_src/reference;
export REFERENCE_FILES
REFERENCE_DATA=$TPCH_ROOT/reference_data/$SCAL
E_FACTOR;export REFERENCE_DATA

SQL=$SETUP/sql; export SQL
SCRIPTS=$SETUP/scripts; export SCRIPTS
SYBASE_OCS="OCS-15_0"; export SYBASE_OCS
SYBROOT="$SYBASE"; export SYBROOT

if [ $VERSION = 12_7 ];
then
        SYBASE_IQ="ASIQ-12_7"; export SYBASE_IQ
        TEST_DBFILE="asiqdemo.db"; export
TEST_DBFILE
        START_SERVER='start_asiq'; export
START_SERVER
        STOP_SERVER='stop_asiq'; export STOP_SERVER
else
        SYBASE_IQ="IQ-15_1"; export SYBASE_IQ
        TEST_DBFILE="iqdemo.db"; export TEST_DBFILE
        START_SERVER='start_iq'; export
START_SERVER
        STOP_SERVER='stop_iq'; export STOP_SERVER
        STOP_SERVER_WITH_CHECKPOINT='stop_iq_ch
eckpoint'; export STOP_SERVER_WITH_CHECKPOINT
fi

#Run the corresponding IQ script.
. $SYBASE/$SYBASE_IQ/$SYBASE_IQ.sh

PATH="${TPCH_ROOT}:${TPCH_ROOT}/dbgen_src/:${TP
CH_ROOT}/bin:${SCRIPTS}:${TPCH_ROOT}/Report_Gene
ration/:${PATH}"
export PATH

INCLUDE="${SYBASE}/${SYBASE_OCS}/include:$INCL
UDE"
export INCLUDE

LIB="${SYBASE}/${SYBASE_OCS}/lib:$LIB"
export LIB

if [ $VERSION = 12_7 ];
then
     alias ISQL="isql -UDBA -PSQL -Stpch_${VERSION} -I
${CONFIG_DIR}/interfaces"
     ISQL="isql -UDBA -PSQL -Stpch_${VERSION} -I
${CONFIG_DIR}/interfaces"; export ISQL
else
     alias ISQL="iqisql -UDBA -Psql -Stpch_${VERSION} -I
${CONFIG_DIR}/interfaces -w 500"
     ISQL="iqisql -UDBA -Psql -Stpch_${VERSION} -I
${CONFIG_DIR}/interfaces -w 500"; export ISQL
fi

if [ -z "$STREAM_COUNT" ];
then
  STREAM_COUNT=`sf_to_streamscount
$SCALE_FACTOR`; export STREAM_COUNT
fi
```

A.2 sysctl.conf

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

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

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

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

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

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

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

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

# Controls the default maxmimum size of a mesage queue
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

# SEMMSL -  maximum number of System V IPC semaphores
per identifier
# SEMMNS -  number of System V IPC system-wide
semaphores
# SEMOPM -  maximum number of semaphore operations
# SEMMNI -  number of System V IPC system-wide
semaphore identifier
kernel.sem = 500 32000 100 128


vm.swappiness = 0
vm.pagecache = 1
```

A.3 tpch.cfg

```
# tpch.cfg
# ------------------------------------------------------------
# Default startup parameters for the ASIQ demo database
# ------------------------------------------------------------

-n  tpch_15_0_sybase
-x  tcpip{port=5788}
-c 64M
-gd all
-gm 25

-gc 5000
-gr 5000
-gp 4096
-tl 0

-iqtss  240

-iqmt 1400
-iqmc 259000
-iqtc 119000

-iqpartition 64
-iqgovern 10
```

A.4 options.sql

```
SET OPTION "PUBLIC".STRING_RTRUNCATION='Off';
SET OPTION "PUBLIC".Allow_Nulls_By_Default='Off';
SET OPTION "PUBLIC".Append_Load='On';
SET OPTION "PUBLIC".Force_No_Scroll_Cursors='On';
SET OPTION "PUBLIC".Load_Memory_Mb=0;
SET OPTION "PUBLIC".Minimize_Storage='On';
SET OPTION "PUBLIC".Notify_Modulus=10000000;
SET OPTION "PUBLIC".Row_Counts='On';
SET OPTION
"public".SignificantDigitsForDoubleEquality=12;
SET OPTION "PUBLIC".QUERY_TEMP_SPACE_LIMIT=0;
SET OPTION "PUBLIC".Hash_Thrashing_Percent=100;

--Performance Options
SET OPTION "PUBLIC".Max_Hash_Rows=5000000;
SET OPTION
"PUBLIC".Default_Having_Selectivity_PPM=10;
SET OPTION "PUBLIC".subquery_flattening_preference=3;
SET OPTION
"PUBLIC".FP_PREDICATE_WORKUNIT_PAGES=50;

SET OPTION "PUBLIC".Garray_Fill_Factor_Percent=3;
SET OPTION
"PUBLIC".Max_IQ_Threads_Per_Connection=500;
SET OPTION "PUBLIC".Max_Query_Parallelism=64;

SET OPTION "PUBLIC".Sweeper_Threads_Percent=8;
SET OPTION "PUBLIC".Wash_Area_Buffers_Percent=30;
SET OPTION "PUBLIC".Prefetch_Threads_Percent=25;
```

Appendix B Build Programs and Scripts

B.1 bld_system

```
#!/bin/ksh
ReadYN()
{
while :
do
	echo -e "$* <Y/N>? \c";

	read ANS

	case "$ANS" in
		Y*|y*)  return 0;;
		N*|n*)  return 1;;
	esac

	echo -e "\nPlease answer 'y' or 'n'\n"
done
}

if [ -z "$1" ]
then
 echo 'usage bld_system <scale_factor>'
 exit 1
fi
#ReadYN "This Script deletes $DEV_DIR/data,
\n\t\t$DEV_DIR/M*,\n\t\t$DEV_DIR/T*,
\n\t\t$DEV_DIR/tpch.db Files. \n Do you want to continue?" ||
exit 2

#ReadYN 'You need to uncomment #drop_dbspace_links.sh in
bld_system if you are using file system devices.\n\n\tDo you
want to Continue?  ' || exit 2

SCALE_FACTOR=$1

. ./bm_setup $SCALE_FACTOR
if [ -z "$DEV_DIR" ]
then
		echo "DEV_DIR is not defined."
		exit 1
else

#echo Drop existing devices.
#drop_dbspace_links.sh

rm -fr $DEV_DIR/data
rm -f $DEV_DIR/M*
rm -f $DEV_DIR/T*
rm -f $DEV_DIR/tpch.db

if [ $? -ne 0 ]; then
		echo "Error during cleanup of $DEV_DIR"
	exit 1
fi

if [ -d $DEV_DIR ]
then
		if [ -f $DEV_DIR/tpch.log ]
		then
			mv $DEV_DIR/tpch.log
$DEV_DIR/tpch_old.`date '+%y%m%d_%H%M%S'`.log
		fi

		if [ -f $DEV_DIR/tpch.iqmsg ]
		then
			mv $DEV_DIR/tpch.iqmsg
$DEV_DIR/tpch_old.`date '+%y%m%d_%H%M%S'`.iqmsg
		fi
fi

fi

mkdir -p $DEV_DIR/data > /dev/null 2>&1
ln -s $DATA_DIR/* $DEV_DIR/data/

if [ $? -ne 0 ]; then
		echo "Error during data file softlink creation at
$DEV_DIR/data"
		exit 1
fi

echo "Replacing the scripts with the correct device dir"
replace.sh

if [ -z "$GENERATED_FILES" ]
then
	echo "GENERATED_FILES is not defined."
	exit 1

else
#Remove and Create GENERATED_FILES Directory
rm -rf $GENERATED_FILES
mkdir -p $GENERATED_FILES
fi

mkdir -p $CONFIG_DIR

gen_interfaces.sh

echo Stop IQ servers if any: `date`
$STOP_SERVER -stop all

sleep 10

echo Startup IQ: `date`
$START_SERVER @$CONFIG_DIR/test.cfg
$SYBASE/$SYBASE_IQ/demo/$TEST_DBFILE
echo IQ started: `date`
echo "  "

# Create softllinks for dbspaces.
```

```
echo Creating soft links for database and dbspace creation.
create_dbspace_links.sh

echo  Creating Database: `date`
echo "  "

dbisqlc -c "DSN=test_$VERSION" -q
$SQL/create_database.sql

echo  Database Created: `date`
echo "  "

echo Shutting down IQ: `date`
$STOP_SERVER -stop one
echo IQ shutdown: `date`
echo "  "

echo Sleeping  5 Seconds
echo "  "
sleep 5


echo Restart IQ with TPCH database: `date`
$START_SERVER @$CONFIG_DIR/tpch.cfg
$DEV_DIR/tpch.db
echo IQ restarted: `date`
echo "  "
echo Adding dbspaces
dbisqlc -c "DSN=tpch_$VERSION" -q
$SQL/create_dbspaces_$VERSION.sql

echo Set Database Options `date`

dbisqlc -c "DSN=tpch_$VERSION" -q $SQL/options.sql

echo "  "

echo Shutting down IQ: `date`
$STOP_SERVER -stop one

#Generate tpch_wait.sql
gen_tpch_wait.sh $SCALE_FACTOR

mv $RESULTS ${RESULTS}_old_`date
'+%m%d%H%M%S'` > /dev/null 2>&1
mkdir $RESULTS > /dev/null 2>&1
mkdir $TPCH_ROOT/plans > /dev/null 2>&1

#Clean up
rm $TEMPFILES/* > /dev/null 2>&1


echo '0'>${DEV_DIR}/data/seq_number
echo "bld_system completed."
```

B.2 check_options.sql

```
set temporary option quoted_identifier = 'on';
sp_iqstatus;
select @@version;
sp_iqcheckoptions;
select * from sys.sysoption order by "option", "user_id" ;
select * from sysiqfile;
select * from sysfile;
-- Detail of DB Device
BEGIN
declare blocksizeX2 unsigned bigint;
select block_size into blocksizeX2 from SYSIQINFO;
select 'DEVLEVEL',dbspace_name as "DB Space Name",
       segment_type as "DB Store",
       sysiqfile.file_name as "UNIX Device Name",
--     block_count as "DB Blocks",
       cast((block_count*blocksizeX2)/(1024.0*1024.0) as
numeric(14,2)) as SizeMb
  from sysiqfile,
       sysfile
 where sysiqfile.file_id = sysfile.file_id
   and segment_type in ('Main','Temp')
order by segment_type;
END;
-- Summary DB Device Info
BEGIN
declare blocksizeX2 unsigned bigint;
select block_size into blocksizeX2 from SYSIQINFO;
select 'SUMLEVEL',segment_type as "DB Store",
--     sum(block_count) as "Total DB Blocks",
       cast(sum((block_count*blocksizeX2)/(1024.0*1024.0)) as
numeric(14,2)) as SizeMb
  from sysiqfile,
       sysfile
 where sysiqfile.file_id = sysfile.file_id
   and segment_type in ('Main','Temp')
group by segment_type;
END;
BEGIN
declare blocksizeX2 unsigned bigint;
select block_size into blocksizeX2 from SYSIQINFO;
-- Total
select 'DBLEVEL',
       cast(sum((block_count*blocksizeX2)/(1024.0*1024.0)) as
numeric(14,2)) as SizeMb
  from sysiqfile,
       sysfile
 where sysiqfile.file_id = sysfile.file_id
   and segment_type in ('Main','Temp');
END;
```

B.3 create_database.sql

```
CREATE DATABASE '/sybase/tpch_audit_kit/devices/tpch.db'
TRANSACTION LOG ON
```

```
COLLATION 'ISO_BINENG'
CASE RESPECT
PAGE SIZE 4096
BLANK PADDING ON
JAVA ON
JCONNECT ON
IQ PATH '/sybase/tpch_audit_kit/devices/M01'
IQ PAGE SIZE 524288
TEMPORARY PATH '/sybase/tpch_audit_kit/devices/T01';
```

B.4 create_dbspace_links.sh

```
# WARNING:
# --------
# IF THE NOTATION IS NOT FOLLOWED, DEVICE INFO
WILL
# BE WRONG IN REPORT.
#
#
================================================
====================
# File: create_dbspace_links.sh
#
================================================
====================
#
# Usage: change the filepaths to reflect actual files.
#
# Please refer setup/create_database.sql
setup/create_dbspaces.sql
# before choosing the device locations.
#
# Device locations should have enough space as specified in the
above
# files.
#
# $DEV_DIR/M01 and $DEV_DIR/T01 store the main store
and temp store
# when the tpch database is created.
#
#
# --------
# Notation
# --------
# For adding/modifying new MAIN dbspaces use M* as name
# for the symbolic link.
# e.g. M05
#
# For adding/modifying new temp dbspaces use T* as name
# for the symbolic link.
# e.g.T05
#
================================================
====================


ln -s /dev/raw/raw1 $DEV_DIR/M01.iq
ln -s /dev/raw/raw5 $DEV_DIR/M02
ln -s /dev/raw/raw9 $DEV_DIR/M03
ln -s /dev/raw/raw13 $DEV_DIR/M04
ln -s /dev/raw/raw2 $DEV_DIR/M05
ln -s /dev/raw/raw6 $DEV_DIR/M06
ln -s /dev/raw/raw10 $DEV_DIR/M07
ln -s /dev/raw/raw14 $DEV_DIR/M08
ln -s /dev/raw/raw3 $DEV_DIR/M09
ln -s /dev/raw/raw7 $DEV_DIR/M10
ln -s /dev/raw/raw11 $DEV_DIR/M11
ln -s /dev/raw/raw15 $DEV_DIR/M12
ln -s /dev/raw/raw4 $DEV_DIR/M13
ln -s /dev/raw/raw8 $DEV_DIR/M14
ln -s /dev/raw/raw12 $DEV_DIR/M15
ln -s /dev/raw/raw16 $DEV_DIR/M16


ln -s /dev/raw/raw17 $DEV_DIR/T01.iqtmp
ln -s /dev/raw/raw21 $DEV_DIR/T02
ln -s /dev/raw/raw25 $DEV_DIR/T03
ln -s /dev/raw/raw29 $DEV_DIR/T04
ln -s /dev/raw/raw18 $DEV_DIR/T05
ln -s /dev/raw/raw22 $DEV_DIR/T06
ln -s /dev/raw/raw26 $DEV_DIR/T07
ln -s /dev/raw/raw30 $DEV_DIR/T08
ln -s /dev/raw/raw19 $DEV_DIR/T09
ln -s /dev/raw/raw23 $DEV_DIR/T10
ln -s /dev/raw/raw27 $DEV_DIR/T11
ln -s /dev/raw/raw31 $DEV_DIR/T12
ln -s /dev/raw/raw20 $DEV_DIR/T13
ln -s /dev/raw/raw24 $DEV_DIR/T14
ln -s /dev/raw/raw28 $DEV_DIR/T15
ln -s /dev/raw/raw32 $DEV_DIR/T16
```

B.5 create_dbspaces_15_0.sql

```
--
================================================
======================
-- Notation:
-- IQ MAIN DBSPACES will be named as iq[0-9]+
-- and
-- IQ TEMP DBSPACES will be named as iqtmp[0-9]+
--
-- This notation is required to help in reporting the main
-- and temp dbspaces in the kit.
--
================================================
======================
alter dbspace IQ_SYSTEM_MAIN ADD
FILE iq2 '/sybase/tpch_audit_kit/devices/M02'
, FILE iq3 '/sybase/tpch_audit_kit/devices/M03'
, FILE iq4 '/sybase/tpch_audit_kit/devices/M04'
, FILE iq5 '/sybase/tpch_audit_kit/devices/M05'
, FILE iq6 '/sybase/tpch_audit_kit/devices/M06'
, FILE iq7 '/sybase/tpch_audit_kit/devices/M07'
```

```
, FILE iq8 '/sybase/tpch_audit_kit/devices/M08'
, FILE iq9 '/sybase/tpch_audit_kit/devices/M09'
, FILE iq10 '/sybase/tpch_audit_kit/devices/M10'
, FILE iq11 '/sybase/tpch_audit_kit/devices/M11'
, FILE iq12 '/sybase/tpch_audit_kit/devices/M12'
, FILE iq13 '/sybase/tpch_audit_kit/devices/M13'
, FILE iq14 '/sybase/tpch_audit_kit/devices/M14'
, FILE iq15 '/sybase/tpch_audit_kit/devices/M15'
, FILE iq16 '/sybase/tpch_audit_kit/devices/M16'
;

alter dbspace IQ_SYSTEM_MAIN
STRIPING ON
STRIPESIZEKB 1024
;

alter dbspace IQ_SYSTEM_TEMP ADD
FILE iqtmp2 '/sybase/tpch_audit_kit/devices/T02'
, FILE iqtmp3 '/sybase/tpch_audit_kit/devices/T03'
, FILE iqtmp4 '/sybase/tpch_audit_kit/devices/T04'
, FILE iqtmp5 '/sybase/tpch_audit_kit/devices/T05'
, FILE iqtmp6 '/sybase/tpch_audit_kit/devices/T06'
, FILE iqtmp7 '/sybase/tpch_audit_kit/devices/T07'
, FILE iqtmp8 '/sybase/tpch_audit_kit/devices/T08'
, FILE iqtmp9 '/sybase/tpch_audit_kit/devices/T09'
, FILE iqtmp10 '/sybase/tpch_audit_kit/devices/T10'
, FILE iqtmp11 '/sybase/tpch_audit_kit/devices/T11'
, FILE iqtmp12 '/sybase/tpch_audit_kit/devices/T12'
, FILE iqtmp13 '/sybase/tpch_audit_kit/devices/T13'
, FILE iqtmp14 '/sybase/tpch_audit_kit/devices/T14'
, FILE iqtmp15 '/sybase/tpch_audit_kit/devices/T15'
, FILE iqtmp16 '/sybase/tpch_audit_kit/devices/T16'
;

alter dbspace IQ_SYSTEM_TEMP
STRIPING ON
STRIPESIZEKB 1024
;
```

B.6 create_refresh_functions.sql

```
--------Create RF1--------------

CREATE PROCEDURE DBA.tpch_rf1 (IN data_directory
varchar(128),
                    IN stream_number varchar(3))
ON EXCEPTION RESUME
BEGIN
  DECLARE sql_cmd long varchar;
  DECLARE c_lf varchar(2);
  DECLARE rf1_start timestamp;
  DECLARE rf1_stop timestamp;
  DECLARE n_seconds numeric(16,5);
  DECLARE cur_sqlstate CHAR(5);

  SET c_lf=char(10);
  SET rf1_start = dateformat(now(*) ,'yyyy-Mm-Dd
hh:nn:ss.sss');
  SELECT 'Stream'||stream_number||' RF1 START TIME --,
'||dateformat(rf1_start,'yyyy-mm-dd hh:nn:ss.sss');

  SET sql_cmd='load table orders ( '+c_lf;
  SET sql_cmd=sql_cmd+' o_orderkey
'+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' o_custkey '+char(39)+'|'+char(39)+',
'+c_lf;
  SET sql_cmd=sql_cmd+' o_orderstatus
'+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' o_totalprice
'+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' o_orderdate
date('+char(39)+'YYYY-MM-DD'+char(39)+'), filler(1), '+c_lf;
  SET sql_cmd=sql_cmd+' o_orderpriority
'+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' o_clerk '+char(39)+'|'+char(39)+',
'+c_lf;
  SET sql_cmd=sql_cmd+' o_shippriority
'+char(39)+'|'+char(39)+','+c_lf;
  SET sql_cmd=sql_cmd+' o_comment
'+char(39)+'|'+char(39)+' ) '+c_lf;
  SET sql_cmd=sql_cmd+'from
'+char(39)+data_directory+'orders.tbl.u'+stream_number+char(
39)+c_lf;
  SET sql_cmd=sql_cmd+'row delimited by
'+char(39)+'\\x0a'+char(39)+' quotes off escapes off preview
on;';
  EXECUTE IMMEDIATE with quotes on sql_cmd;
  SELECT SQLSTATE INTO cur_sqlstate;
  IF cur_sqlstate != '00000' THEN
    ROLLBACK;
    RAISERROR 23002 'RF1 failed while inserting into orders
with SQLSTATE: ', cur_sqlstate;
    RETURN(1);
  END IF;

--------insert into lineitem------------
  SET sql_cmd='load table lineitem ( '+c_lf;
  SET sql_cmd=sql_cmd+' l_orderkey '+char(39)+'|'+char(39)+',
'+c_lf;
  SET sql_cmd=sql_cmd+' l_partkey '+char(39)+'|'+char(39)+',
'+c_lf;
  SET sql_cmd=sql_cmd+' l_suppkey '+char(39)+'|'+char(39)+',
'+c_lf;
  SET sql_cmd=sql_cmd+' l_linenumber
'+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' l_quantity '+char(39)+'|'+char(39)+',
'+c_lf;
  SET sql_cmd=sql_cmd+' l_extendedprice
'+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' l_discount '+char(39)+'|'+char(39)+',
'+c_lf;
  SET sql_cmd=sql_cmd+' l_tax '+char(39)+'|'+char(39)+',
'+c_lf;
```

```
  SET sql_cmd=sql_cmd+' l_returnflag
'+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' l_linestatus
'+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' l_shipdate date('+char(39)+'YYYY-
MM-DD'+char(39)+'), filler(1), '+c_lf;
  SET sql_cmd=sql_cmd+' l_commitdate
date('+char(39)+'YYYY-MM-DD'+char(39)+'), filler(1), '+c_lf;
  SET sql_cmd=sql_cmd+' l_receiptdate
date('+char(39)+'YYYY-MM-DD'+char(39)+'), filler(1), '+c_lf;
  SET sql_cmd=sql_cmd+' l_shipinstruct
'+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' l_shipmode
'+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' l_comment '+char(39)+'|'+char(39)+'
)'+c_lf;
  SET sql_cmd=sql_cmd+'from
'+char(39)+data_directory+'lineitem.tbl.u'+stream_number+cha
r(39)+c_lf;
  SET sql_cmd=sql_cmd+'row delimited by
'+char(39)+'\\x0a'+char(39)+c_lf+'quotes off escapes off
preview on;';
  EXECUTE IMMEDIATE with quotes on sql_cmd;

  SELECT SQLSTATE INTO cur_sqlstate;
  IF cur_sqlstate != '00000' THEN
    rollback;
    RAISERROR 23002 'RF1 failed while inserting into lineitem
with SQLSTATE: ', cur_sqlstate;
    RETURN(1);
  END IF;
  COMMIT;
  SET rf1_stop = dateformat(now(*) ,'yyyy-Mm-Dd
hh:nn:ss.sss');
  SELECT 'Stream'||stream_number||' RF1 STOP TIME --,
'||dateformat(rf1_stop ,'yyyy-mm-dd hh:nn:ss.sss');

---------Find the execution time of rf1.----------------
  SET n_seconds=cast(datediff(millisecond,rf1_start,rf1_stop)
AS numeric(16,5))/1000;
  SET sql_cmd='Stream_'+stream_number+' RF1 Elapsed time
--, '+cast(n_seconds AS varchar(20))+ ' seconds' ;
  SELECT sql_cmd;
  RETURN(0);
END;

---------Create RF2----------------------------
CREATE PROCEDURE DBA.tpch_rf2 (in data_directory
varchar(128),
                        in stream_number varchar(3))
ON exception resume
BEGIN
  DECLARE sql_cmd long varchar;
  DECLARE c_lf varchar(2);
  DECLARE rf2_start timestamp;
  DECLARE rf2_stop timestamp;
  DECLARE n_seconds numeric(16,5);
  DECLARE cur_sqlstate CHAR(5);
  SET c_lf=char(10);
  SET rf2_start = dateformat(now(*) ,'yyyy-Mm-Dd
hh:nn:ss.sss');
  SELECT 'Stream'||stream_number||' RF2 START TIME --,
'||dateformat(rf2_start,'yyyy-mm-dd hh:nn:ss.sss');

  CREATE TABLE #delete_table ( d_orderkey UNSIGNED
BIGINT );

  SET sql_cmd='load table #delete_table  (d_orderkey
'+char(39)+'|'+char(39)+')'+c_lf;
  SET sql_cmd=sql_cmd+'from
'+char(39)+data_directory+'delete.'+stream_number+char(39)+
c_lf;
  SET sql_cmd=sql_cmd+'row delimited by
'+char(39)+'\\x0a'+char(39)+c_lf;
  SET sql_cmd=sql_cmd+'quotes off '+c_lf;
  SET sql_cmd=sql_cmd+'escapes off; '+c_lf;

  EXECUTE IMMEDIATE with quotes on sql_cmd;
  SELECT SQLSTATE INTO cur_sqlstate;
  IF cur_sqlstate != '00000' THEN
    ROLLBACK;
    SET sql_cmd='RF2 failed at Step 1 with SQLSTATE:
'+cur_sqlstate;
    RAISERROR 23002 sql_cmd;
    RETURN(1);
  END IF;

-------- Delete from LineItem---------
  DELETE lineitem FROM lineitem
     WHERE l_orderkey in (select d_orderkey from
#delete_table);
  SELECT SQLSTATE INTO cur_sqlstate;

  IF cur_sqlstate != '00000' THEN
    ROLLBACK;
    SET sql_cmd='RF2 failed at Step 2 with SQLSTATE:
'+cur_sqlstate;
    RAISERROR 23002 sql_cmd;
    RETURN(1);
  END IF;

-----Delete fom Orders.----------
  DELETE orders FROM orders
      WHERE o_orderkey in (select d_orderkey from
#delete_table);
  SELECT SQLSTATE INTO cur_sqlstate;

  IF cur_sqlstate != '00000' THEN
    ROLLBACK;
    SET sql_cmd='RF2 failed at Step 3 with SQLSTATE:
'+cur_sqlstate;
    RAISERROR 23002 sql_cmd;
    RETURN(1);
  END IF;
```

```
  COMMIT;
  DROP TABLE #delete_table;
  SET rf2_stop = dateformat(now(*) ,'yyyy-Mm-Dd
hh:nn:ss.sss');
  SELECT 'Stream'||stream_number||' RF2 STOP TIME --,
'||dateformat(rf2_stop,'yyyy-mm-dd hh:nn:ss.sss');
--------- Calculate execution time for rf2.-------

  SET n_seconds=cast(datediff(millisecond,rf2_start,rf2_stop)
as numeric(16,5))/1000;
  SET sql_cmd='Stream_'+stream_number+' RF2 Elapsed time
--, '+cast(n_seconds as varchar(20))+ ' seconds' ;
  SELECT sql_cmd;
  RETURN(0);
END;
```

B.7 create_schema

```
#!/bin/ksh


echo "  "
echo Create Tables `date`|tee -a $RESULTS/ldeventlog

dbisqlc -c "DSN=tpch_$VERSION" -q
$SQL/drop_tables.sql>>$RESULTS/ldeventlog

dbisqlc -c "DSN=tpch_$VERSION" -q
$SQL/create_tables.sql>>$RESULTS/ldeventlog

echo "  "
echo Dump the IQ Configuration `date`

echo "Dropping refresh functions."|tee -a
$RESULTS/ldeventlog
dbisqlc -c "DSN=tpch_$VERSION" -q
$SQL/drop_refresh_functions.sql>>$RESULTS/ldeventlog
echo "Creating refresh functions."|tee -a $RESULTS/ldeventlog
dbisqlc -c "DSN=tpch_$VERSION" -q
$SQL/create_refresh_functions.sql>>$RESULTS/ldeventlog
dbisqlc -c "DSN=tpch_$VERSION" -q
$GENERATED_FILES/tpch_wait.sql>>$RESULTS/ldeventlo
g
```

B.8 create_tables.sql

```
set temporary option on_error = 'continue';


set temporary option Allow_nulls_by_default='on';

CREATE TABLE region
 (
   r_regionkey          unsigned int,
   r_name               char(25),
   r_comment            varchar(152),
   PRIMARY KEY (r_regionkey)
 );


CREATE TABLE nation
 (
   n_nationkey          unsigned int,
   n_name               char(25),
   n_regionkey          unsigned int ,
   n_comment            varchar(152),
   PRIMARY KEY (n_nationkey)
 );


CREATE TABLE supplier
 (
   s_suppkey            unsigned int,
   s_name               char(25),
   s_address            varchar(40),
   s_nationkey          unsigned int ,
   s_phone              char(15),
   s_acctbal            double precision,
   s_comment            varchar(101),
   PRIMARY KEY (s_suppkey)
 );


CREATE TABLE part
 (
   p_partkey            unsigned int,
   p_name               varchar(55),
   p_mfgr               char(25),
   p_brand              char(10),
   p_type               varchar(25),
   p_size               int,
   p_container          char(10),
   p_retailprice        double precision,
   p_comment            varchar(23),
   PRIMARY KEY(p_partkey)
 );


CREATE TABLE partsupp
 (
   ps_partkey           unsigned int ,
   ps_suppkey           unsigned int ,
   ps_availqty          integer,
   ps_supplycost        double precision,
   ps_comment           varchar(199),
   PRIMARY KEY (ps_partkey, ps_suppkey)
 );

CREATE TABLE customer
```

```
 (
  c_custkey            unsigned int,
  c_name               varchar(25),
  c_address            varchar(40),
  c_nationkey          unsigned int ,
  c_phone              char(15),
  c_acctbal            double precision,
  c_mktsegment         char(10),
  c_comment            varchar(117),
  PRIMARY KEY(c_custkey)
 );


CREATE TABLE orders
 (
  o_orderkey           unsigned bigint,
  o_custkey            unsigned int ,
  o_orderstatus        char(1),
  o_totalprice         double precision,
  o_orderdate          date,
  o_orderpriority      char(15),
  o_clerk              char(15),
  o_shippriority       int,
  o_comment            varchar(79),
  PRIMARY KEY (o_orderkey)
 )partition by range (o_orderdate)
  (
   p0 values <= ('1991-12-31'),
   p1 values <= ('1992-12-31'),
   p2 values <= ('1993-12-31'),
   p3 values <= ('1994-12-31'),
   p4 values <= ('1995-12-31'),
   p5 values <= ('1996-12-31'),
   p6 values <= ('1997-12-31'),
   p7 values <= ('1998-12-31'),
   p8 values <= (MAX)
  );


CREATE TABLE lineitem
 (
  l_orderkey           unsigned bigint ,
  l_partkey            unsigned int,
  l_suppkey            unsigned int,
  l_linenumber         int,
  l_quantity           double precision,
  l_extendedprice      double precision,
  l_discount           double precision,
  l_tax                double precision,
  l_returnflag         char(1),
  l_linestatus         char(1),
  l_shipdate           date,
  l_commitdate         date,
  l_receiptdate        date,
  l_shipinstruct       char(25),
  l_shipmode           char(10),
  l_comment            varchar(44)
 )
 partition by range(l_shipdate)
  (
   p0 values <= ('1991-12-31'),
   p1 values <= ('1992-12-31'),
   p2 values <= ('1993-12-31'),
   p3 values <= ('1994-12-31'),
   p4 values <= ('1995-12-31'),
   p5 values <= ('1996-12-31'),
   p6 values <= ('1997-12-31'),
   p7 values <= ('1998-12-31'),
   p8 values <= (MAX)
  );

CREATE DATE INDEX o_orderdate_date ON
orders(o_orderdate) ;
CREATE DATE INDEX l_shipdate_date ON
lineitem(l_shipdate) ;
CREATE DATE INDEX l_receiptdate_date ON
lineitem(l_receiptdate);

create HG index n_regionkey_hg on nation ( n_regionkey );
create HG index s_nationkey_hg on supplier ( s_nationkey );
create HG index c_nationkey_hg on customer ( c_nationkey );
create HG index o_custkey_hg on orders ( o_custkey );
create HG index l_partkey_hg on lineitem ( l_partkey );

create HG index ps_suppkey_hg on partsupp ( ps_suppkey );
create HG index ps_partkey_hg on partsupp ( ps_partkey );

create HG index l_suppkey_hg on lineitem ( l_suppkey );

create HG index l_orderkey_hg on lineitem ( l_orderkey );

commit;
```

B.9 drop_refresh_functions.sql

```
if ((select object_id('refresh_control')) is not NULL)
begin
        drop table refresh_control
end;
if ((select object_id('DBA.tpch_rf1')) is not NULL)
begin
drop procedure DBA.tpch_rf1
end;
if ((select object_id('DBA.tpch_rf2')) is not NULL)
begin
drop procedure DBA.tpch_rf2
end
```

B.10 drop_tables.sql

```
if ((select object_id('region')) is not NULL)
begin
        drop table region
end;
if ((select object_id('nation')) is not NULL)
begin
drop table nation
end;
if ((select object_id('customer')) is not NULL)
begin
drop table customer
end;
if ((select object_id('part')) is not NULL)
begin
    drop table part
end;
if ((select object_id('supplier')) is not NULL)
begin
drop table supplier
end;
if ((select object_id('partsupp')) is not NULL)
begin
drop table partsupp
end;
if ((select object_id('lineitem')) is not NULL)
begin
drop table lineitem
end;
if ((select object_id('orders')) is not NULL)
begin
drop table orders
end
```

B.11 gen_interfaces.sh

```
#Generate interfaces file
if [ -z "$HOSTNAME" ]
then
    HOSTNAME=`uname -n`
fi

USER_NAME=`whoami`

#Port Number shouldn't be in use already. Use last 4 digits of
your phone number for uniqueness.
#Server name on the machine should be unique.

PORT_NUMBER=5788
TEST_ENGINE_NAME=test_${VERSION}_${USER_NAM
E}
TPCH_ENGINE_NAME=tpch_${VERSION}_${USER_NA
ME}

if [ $VERSION = 12_7 ]
then
        PASSWORD=SQL
        DATABASE_NAME=asiqdemo
        DATABASE_FILE=asiqdemo.db
else
        PASSWORD=sql
        DATABASE_NAME=iqdemo
        DATABASE_FILE=iqdemo.db
fi

cat > $CONFIG_DIR/interfaces <<EOF
tpch_${VERSION}
    master tcp ether $HOSTNAME ${PORT_NUMBER}
    query tcp ether $HOSTNAME ${PORT_NUMBER}
EOF


#Generate odbc.ini file using template odbc.ini_template
if [ ! -f $CONFIG_DIR/.odbc.ini ]
then
sed -e 's/HOSTNAME/'$HOSTNAME'/'
$TEMPLATES/odbc.ini_template | \
        sed -e 's/PORT_NUMBER/'$PORT_NUMBER'/' | \
        sed -e
's/TEST_ENGINE_NAME/'$TEST_ENGINE_NAME'/' | \
        sed -e
's/TPCH_ENGINE_NAME/'$TPCH_ENGINE_NAME'/' | \
        sed -e 's/PASSWORD/'$PASSWORD'/' | \
        sed -e
's/DATABASE_NAME/'$DATABASE_NAME'/' | \
        sed -e 's/DATABASE_FILE/'$DATABASE_FILE'/' | \
        sed -e 's/VERSION/'$VERSION'/'
        > $CONFIG_DIR/.odbc.ini
fi

#Generate tpch.cfg
if [ ! -f $CONFIG_DIR/tpch.cfg ]
then
        sed -e
's/TPCH_ENGINE_NAME/'$TPCH_ENGINE_NAME'/'
$TEMPLATES/tpch.cfg.sf_${SCALE_FACTOR}| \
                sed -e
's/PORT_NUMBER/'$PORT_NUMBER'/'
        > $CONFIG_DIR/tpch.cfg
fi

#Generate test.cfg
if [ ! -f $CONFIG_DIR/test.cfg ]
then
        sed -e
's/TEST_ENGINE_NAME/'$TEST_ENGINE_NAME'/'
$TEMPLATES/test.cfg | \
                sed -e
's/PORT_NUMBER/'$PORT_NUMBER'/'          >
$CONFIG_DIR/test.cfg
fi
```

B.12 gen_tpch_wait.sh

```
#!/bin/ksh

if [ -z "$1" ]
then
        echo "Usage: gen_tpch_wait.sh <Scale-Factor>\n"
        exit
fi

SCALE_FACTOR=$1

cat > $TEMPFILES/tpch_wait_thresholds<< EOF
1,124000
10,1240000
30,3720000
100,12400000
300,37200000
1000,124000000
3000,372000000
10000,1240000000
30000,3720000000
100000,12400000000
EOF

threshold=$(grep "^$SCALE_FACTOR,"
$TEMPFILES/tpch_wait_thresholds | cut -d"," -f2)

echo '-- THIS IS A GENERATED FILE.'
>$GENERATED_FILES/tpch_wait.sql
echo '-- PLEASE CHANGE gen_tpch_wait.sh FOR
MODIFICATIONS' >>$GENERATED_FILES/tpch_wait.sql

sed -e 's/MINTHRESHOLDMAINFREE/'$threshold'/'
$TEMPLATES/tpch_wait_template.sql
>>$GENERATED_FILES/tpch_wait.sql
```

B.13 load_test

```
#!/bin/ksh
#Usage load_test [enable_monitoring]

if [ ! -z "$1" ]
then
        Enable_Monitoring=$1
else
        Enable_Monitoring=0
fi


mkdir $RESULTS/Load
echo Start load_test `format_date.sh` | tee -a
$RESULTS/Load/load_test.out
#create schema tables, RFs, tpch_wait function.
create_schema
dbisqlc -c "DSN=tpch_$VERSION" -q
$SQL/check_options.sql>>$RESULTS/ldeventlog

CURRENTLY_RUNNING=load; export
CURRENTLY_RUNNING

if [ $Enable_Monitoring -ne 0 ]
then
      start_monitor.sh
fi

# Load the database
load_db

SEED=`date '+%m%d%H%M%S'`;
#store SEED for future reportiing purpose
echo $SEED >$DEV_DIR/data/seed

if [ $Enable_Monitoring -ne 0 ]
then
      kill_stats.sh
fi

echo Finish load_test `format_date.sh` | tee -a
$RESULTS/Load/load_test.out
```

B.14 replace.sh

```
#!/bin/ksh

mydev_dir=`echo $DEV_DIR | sed -e 's|\/|\\\/|g'`
export mydev_dir
echo $mydev_dir

mycmd="sed -e s/\$DEV_DIR/$mydev_dir/g"
export mycmd
echo $mycmd

for i in `find . -name \*.sql`
do
        grep "DEV_DIR" $i
        if [ $? -eq 0 ]
        then
                cat $i | $mycmd > $TEMPFILES/tmp.txt
         mv -f $TEMPFILES/tmp.txt $i
        fi
done
```

Appendix C Acid Scripts

C.1 run_atomicity

```
#!/bin/ksh
cd $ACID_ROOT/atomicity
dbtest $ACID_ROOT/atomicity/acid_atomic_main.tst >
$ACID_RESULTS/acid_atomic_main.out

roll_back=1

rm -f $ACID_RESULTS/atomc $ACID_RESULTS/atomr

while read line
do
	if [ roll_back -eq 1 ]
	then
		commit_started=`echo $line |grep "Starting
atomicity test with commit"`
		if [ ! -z "$commit_started" ]
		then
			roll_back=0
			echo "$line" >
$ACID_RESULTS/atomc
		else
			echo "$line" >>
$ACID_RESULTS/atomr
		fi
	else
		echo "$line" >> $ACID_RESULTS/atomc
	fi
done < $ACID_RESULTS/acid_atomic_main.out
mv $ACID_RESULTS/atomr $ACID_RESULTS/atomr.`date
'+%y%m%d_%H%M%S'`
mv $ACID_RESULTS/atomc $ACID_RESULTS/atomc.`date
'+%y%m%d_%H%M%S'`
```

C.2 acid_atomic_main.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
% Created by: Masood Dirin
% Create Date:  April 21, 1999
%
% Purpose of this test is to run and verify the pass of the ACID
Atomicity
% test.

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

Test		"tpcd_acid_atomic_main.tst"
Description	"To run the ACID atomicity test"

stringconnect "dsn=qual_15_0;"

execute {select now(*)} into times
print 'Atomicity test start = ', times
print ' '

include 'acid_functions.tst'
commit

%
% Atomicity test with rollback
%
print ' '
print 'Starting atomicity test with rollback'
print ' '

%include 'acid_atomic_setup.tst'
run test 'acid_atomic_setup.tst'

stringconnect "dsn=qual_15_0;"
let counter=0

LOOP {
open cur2 {select ordr, line, delta from aa_whattodo where
seqnum=^}
	substitute counter
print 'counter = ',counter
fetch cur2 into ordr, line, delta
if ROWSTATUS != FOUND then { BREAK LOOP } endif
print 'Acid transaction for: o_key-',ordr,'  l_key-', line,'  delta-
',delta


print 'Initial values:'
execute {select o_totalprice, l_quantity, l_extendedprice
	from orders, lineitem
	where o_orderkey = l_orderkey and o_orderkey =^
and l_linenumber = ^}
	substitute ordr, line
	into o_total, l_quan, l_price
print	'o_totalprice = ',o_total,'     l_quantity = ',l_quan,
	'		l_extendedprice = ',l_price

execute {call acid_transaction(^, ^,  ^, rprice, quantity,
				tax, disc, extprice, ototal)
	} substitute ordr, line, delta
close cur2

execute {select count(*)
    from history
    where h_o_key =^ and h_l_key =^}
    substitute ordr, line
    into total_history_count
execute {select o_totalprice, l_quantity, l_extendedprice
    from orders, lineitem
    where o_orderkey = l_orderkey and o_orderkey =^ and
l_linenumber = ^}
    substitute ordr, line
```

```
      into o_total, l_quan, l_price
print      'Before Rolling back:'
print   'o_totalprice = ',o_total,'     l_quantity = ',l_quan,
      '       l_extendedprice = ',l_price

print   'Before Rollback History table
count=',total_history_count


let counter = counter+1

rollback
execute {select now(*)} into times
print 'rollback : ', times

execute {select o_totalprice, l_quantity, l_extendedprice
           from orders, lineitem
           where o_orderkey = l_orderkey and o_orderkey =^
and l_linenumber = ^}
           substitute ordr, line
           into o_total, l_quan, l_price

print      'After Rollback:'
print      'o_totalprice = ',o_total,'     l_quantity = ',l_quan,
           '            l_extendedprice = ',l_price
print ' '

execute {select count(*)
           from history
           where h_o_key =^ and h_l_key =^}
           substitute ordr, line
           into total_history_count

print      'After Rollback History table
count=',total_history_count

} ENDLOOP

commit

%
% Atomicity test with commit
%
stringconnect "dsn=qual_15_0;"
print ' '
print 'Starting atomicity test with commit '
print ' '
%include 'acid_atomic_setup.tst'
run test 'acid_atomic_setup.tst'

stringconnect "dsn=qual_15_0;"

open cur1 {select ordr, line, delta from aa_whattodo}
LOOP {
fetch cur1 into ordr, line, delta
if ROWSTATUS != FOUND then { BREAK LOOP } endif
print 'Acid transaction for: o_key-',ordr,'  l_key-', line,'  delta-
',delta
execute {select o_totalprice, l_quantity, l_extendedprice
           from orders, lineitem
           where o_orderkey = l_orderkey and o_orderkey =^
and l_linenumber = ^}
           substitute ordr, line
           into o_total, l_quan, l_price
print      'Initial values:'
print      'o_totalprice = ',o_total,'     l_quantity = ',l_quan,
           '            l_extendedprice = ',l_price
print      ''
print      ''
print   'Before Commit:'
print      '            l_extendedprice = ',l_price

execute {call acid_transaction(^, ^,  ^, rprice, quantity,
                                          tax, disc, extprice, ototal)
           } substitute ordr, line, delta


execute {select o_totalprice, l_quantity, l_extendedprice
      from orders, lineitem
      where o_orderkey = l_orderkey and o_orderkey =^ and
l_linenumber = ^}
      substitute ordr, line
      into o_total, l_quan, l_price

execute {select count(*)
      from history
      where h_o_key =^ and h_l_key =^}
      substitute ordr, line
      into total_history_count


print   'Before Commit:'
print   'o_totalprice = ',o_total,'     l_quantity = ',l_quan,
      '       l_extendedprice = ',l_price

print   'Before Commit History table
count=',total_history_count

commit
execute {select now(*)} into times
print 'commit : ', times

print      ''
execute {select o_totalprice, l_quantity, l_extendedprice
           from orders, lineitem
           where o_orderkey = l_orderkey and o_orderkey =^
and l_linenumber = ^}
           substitute ordr, line
           into o_total, l_quan, l_price

execute {select count(*)
      from history
      where h_o_key =^ and h_l_key =^}
      substitute ordr, line
      into total_history_count
```

```
print      'After Commit:'
print      'o_totalprice = ',o_total,'     l_quantity = ',l_quan,
           '              l_extendedprice = ',l_price
print   'After Commit History table count=',total_history_count

print ' '

} ENDLOOP

close cur1
commit

execute {select now(*)} into times
print 'Atomicity test end = ', times

End Test
```

C.3 acid_atomic_setup.tst

```
Test                "acid_setup.tst"
Description         "Creates aa_whattodo table"

stringconnect "dsn=qual_15_0;"

% Drop Table if found
print 'aa_whattodo!!'
allow error -141
execute { commit }
execute { drop table aa_whattodo }
allow no error

execute {
create table aa_whattodo (
              seqnum       int      not null,
              ordr         int      not null,
              line         int      null,
              delta        int      null)
}

print 'aa_whattodo CREATED!!'
execute {select now(*)} into times
print 'time = ', times

fetch {select count(*) from aa_whattodo } into ROWS
assert ROWS = 0

print 'Number of rows before load: ',ROWS

LOOP ({let counter = 0}; {counter < 5}; {let counter = counter
+ 1})
{
           execute {call generate_acid_values()}
                    into orderkey, linenumber,delta
           execute {insert into aa_whattodo values ( ^ , ^ , ^ , ^ )
}
                    substitute counter, orderkey, linenumber,
delta
 print  counter, ' ',orderkey, ' ',linenumber,' ', delta
}
ENDLOOP

commit

fetch {select count(*) from aa_whattodo } into ROWS
assert ROWS = 5

print 'Number of rows after load: ',ROWS

disconnect

End Test
```

C.4 acid_functions.tst

C.5 run_consistency

```
#!/bin/ksh
cd $ACID_ROOT/consistency/
initial_size=`cat $ACID_DEVICES/qual.iqmsg|wc -l`

dbtest $ACID_ROOT/consistency/acid_consistency_main.tst >
$ACID_RESULTS/consbe.`date '+%y%m%d_%H%M%S'`

final_size=`cat $ACID_DEVICES/qual.iqmsg|wc -l`

lines_during_test=`expr $final_size - $initial_size`

tail -$lines_during_test $ACID_DEVICES/qual.iqmsg |grep -i
chk> $ACID_RESULTS/consckpt.`date
'+%y%m%d_%H%M%S'`

mv *.ot $ACID_RESULTS
```

C.6 acid_consistency_main.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
%
% acid_consistency_main.tst
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%

Test                "tpch_acid_consistency_main.tst"
Description         "To run the ACID consistency test"

stringconnect "dsn=qual_15_0;"
```

```
execute {select now(*)} into times
print 'Consistency test start = ', times
print ' '

include 'acid_functions.tst'
include 'acid_consistency_setup.tst'

%run test 'acid_consistency_setup.tst'

execute {select now(*)} into times
print 'Consistency test time = ', times
print ' '

run test '-o' 'acid_consistency_q1.ot'
'acid_consistency_query.tst'
disconnect

let i = 1
LOOP {
      if i > 10 then { BREAK LOOP } endif
           let ot_file = "acid_consist_user", i, ".ot"
           let my_str = "stream=", i
           print ot_file, my_str
           start test '-o' ot_file my_str 'acid_consistency_txn.tst'
           sleep 900
           let i = i + 1
} ENDLOOP


synchronize 11

% let the log flush...
sleep 900
stringconnect "dsn=qual_15_0;"
%include 'acid_consistency_query.tst'
run test '-o' 'acid_consistency_q2.ot'
'acid_consistency_query.tst'

execute {select now(*)} into times
print 'Consistency test end = ', times
print ' '

End Test
```

C.7 acid_consistency_query.tst

```
Test 'tpch_acid_query'
Description 'perform the consistency check.'



stringconnect "dsn=qual_15_0;"

open cur1 {select stream, seqnum, ordr, line, delta from
acid_table
                  where seqnum > 10 order by seqnum}
print ' '

execute {select now(*)} into times
print 'Consistency verification start = ', times
print ' '

let n=1
LOOP {
  fetch cur1 into str, seq, ord, lin, delta

  fetch {select round(cast(o_totalprice as numeric(26,16)),2)
                  from orders where o_orderkey=^ }
          substitute ord into o_price

  if ROWSTATUS != FOUND then { BREAK LOOP } endif
  if n > 25 then { BREAK LOOP } endif

  execute { call acid_single_query (^) } substitute ord into
l_total

  fetch {select cast(^ as numeric(12,2)) } substitute o_price into
o_price
  fetch {select cast(^ as numeric(12,2)) } substitute l_total into
l_total

   print 'orderkey = ', ord, '
      o_totalprice = ', o_price,
      '     acid query = ' , l_total

   ASSERT (o_price = l_total)
      then { print 'Did not compare correctly' }
ENDASSERT
  let n=n+1
 } ENDLOOP

execute {select now(*)} into times
print 'Consistency verification end = ', times
print ' '

disconnect


END Test
```

C.8 acid_consistency_setup.tst

```
Test            "acid_consistency_setup.tst"
Description     "Creates acid_table table"

stringconnect "dsn=qual_15_0;"

execute { set option public.isolation_level=3 }
execute {set option public.query_plan='off'}
execute {set temporary option chained='on'}
execute {set option public.auto_commit=off}
```

```
% Drop Table if found
allow error -141
execute { drop table acid_table }
execute {drop table latest}
allow no error

execute {
create table acid_table (
                    stream   int        null,
            seqnum  int      null,
            ordr    int      null,
            line    int      null,
            delta   int      null)
on SYSTEM
}

fetch {select count(*) from acid_table } into ROWS
assert ROWS = 0
print 'Number of rows before load: ',ROWS
commit

print 'acid_table created'
execute {create table latest(stream int ,last int null) on
SYSTEM }
LOOP ({let j = 1}; {j <= 22}; {let j = j + 1})
   {
          execute { insert into latest(stream,last) values (^,0) }
                    substitute j
   } endloop
commit

print 'latest created'

LOOP ({let i = 1}; {i <= 22}; { let i = i + 1})
{
  lOOP ({let j = 1}; {j <= 100}; {let j = j + 1})
   {
          execute { call generate_acid_values()} into ordr, line,
delta
          execute { insert into acid_table values (^,^,^,^,^) }
                    substitute i,j,ordr,line,delta
   } endloop
  print (j-1)*i
} endloop

commit

fetch {select count(*) from acid_table } into ROWS
assert ROWS = 2200
print 'Number of rows after load: ',ROWS

End Test
```

C.9 acid_consistency_txn.tst

```
Test                "tpch_transaction.tst"
Description         "Run Acid Multiple Transactions"

stringconnect "dsn=qual_15_0;"

execute {set temporary option chained='on'}

execute {select now(*)} into times
print 'Consistancy test start = ', times
print ' '
print 'Stream:',stream
commit
LOOP ({let i = 1}; {i <= 100}; { let i = i + 1})
{
 fetch {select ordr, line, delta from acid_table
                    where stream=^ and seqnum=^ }
          substitute stream, i
 commit
 if ROWSTATUS != FOUND then { print 'not enough rows'
                BREAK LOOP }
 endif

 print 'User=',stream,' Acid Txn=',i,
          '   o_key=', ordr , '  l_key=', line , '  delta=' ,delta

 execute {call acid_transaction( ^, ^,  ^)
          } substitute ordr, line, delta into rprice,quantity,tax,
disc, extprice, ototal
print 'O_total = ',ototal,'quantity = ',quantity
 commit
print   ''
execute {select o_totalprice, l_quantity, l_extendedprice
      from orders, lineitem
      where o_orderkey = l_orderkey and o_orderkey =^ and
l_linenumber = ^}
      substitute ordr, line
      into o_total, l_quan, l_price

execute {select count(*)
      from history
      where h_o_key =^ and h_l_key =^}
      substitute ordr, line
      into total_history_count

print   'After Commit:'
print   'o_totalprice = ',o_total,'     l_quantity = ',l_quan,
      '      l_extendedprice = ',l_price
print   'After Commit History table count=',total_history_count

print ' '
 execute { update latest set last=^ where stream=^ } substitute
i,stream
 commit
 execute { set temporary option isolation_level=1 }
 execute { select max(last) from latest } into biggest
```

```
 execute { select min(last) from latest } into smallest
 commit
 execute { set temporary option isolation_level=3 }

 let num=120*(i-smallest)
 if i+4>=biggest
 then {let num=num+800}
 endif
--print 'user',stream,' = ',num
 sleep num
}
ENDLOOP
synchronize 11
disconnect

End Test
```

C.10 acid_durability_history.sql

```
select now(*)
;
select 'Transaction Count',
      count(*)
  from history
;
select h_p_key,
      h_s_key,
      h_o_key,
      h_l_key,
      h_delta,
      convert(varchar(30),h_date_t,109)
  from history
;
```

C.11 acid_durability_kill_and_continue.tst

```
Test            "tpch_kill_and_continue.tst"
Description     "kills the server, reboots it & checks for the
validity."

stringconnect "dsn=qual_15_0;"

            execute { set option isolation_level=1 }
LOOP ({let i = 1}; {i <= 1}; { let i = i + 0})
{
            execute { select min(last) from latest } into smallest
            commit

            if smallest>=100
            then
            {
                        BREAK LOOP
            }
            else
            {
                        sleep 1000
            }
            endif
}
ENDLOOP

run 'sh' './kill_and_continue.sh'
```

```
 run_durability
#!/bin/ksh
if [ -z "$1" ]
then
            echo "Usage:$0 $failure_type"
            exit 1
else
            failure_type=$1
fi

# The following block is used to capture system failure log.
# It requires read permission on /var/log/messages file.
if [ -z "$SERVER" ]
then
            wc -l /var/log/messages | cut -f 1 -d' ' >
$TEMPFILES/msgs_lc
else
            rsh $SERVER "wc -l /var/log/messages | cut -f 1 -d' '"
> $TEMPFILES/msgs_lc
fi

{ while true; do sync; sleep 1; done } &

cd $ACID_ROOT/durability
show_user_count $failure_type&
dbtest acid_durability_main.tst >
$ACID_RESULTS/acid_durability_main.out
```

C.12 acid_durability_main.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
%
% acid_durability_main.tst
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%

Test                    "tpch_acid_durability_main.tst"
Description             "To run the ACID durability test"

stringconnect "dsn=qual_15_0;"

execute {select now(*)} into times
print 'Durability test start = ', times
print ' '
```

```
include 'acid_functions.tst'
run test 'acid_durability_setup.tst'

execute {select now(*)} into times
print 'Durability test time = ', times
print ' '

run test  '-o' 'acid_durability_q1.ot' 'acid_durability_query.tst'

%start the fault to occur after 100 + transactions.
%start test '-o' 'kill.out' 'acid_durability_kill_and_continue.tst'

LOOP( { let i = 1 }; { i <= 10 }; { let i = i + 1 } )
{
        let ot_file = "acid_dura_user", i, ".ot"
        let my_str = "stream=", i

        start test '-o' ot_file my_str 'acid_durability_txn.tst'
        sleep 950
}
ENDLOOP

print 'Out of loop. Parent waiting for synch'
synchronize 11

execute {select now(*)} into times
print 'Durability test time = ', times
print ' '

run test  '-o' 'acid_durability_q2.ot' 'acid_durability_query.tst'

execute {select now(*)} into times
print 'Durability test end = ', times
print ' '

End Test
```

C.13 acid_durability_query.tst

```
Test 'tpch_acid_query'
Description 'perform the acid query.'

stringconnect "dsn=qual_15_0;"

open cur1 {select stream, seqnum, ordr, line, delta from
acid_table
                where seqnum > 5 order by seqnum}
print ' '

let n=1
LOOP {
  fetch cur1 into str, seq, ord, lin, delta

  fetch {select round(cast(o_totalprice as numeric(26,16)),2)
                from orders where o_orderkey=^ }
        substitute ord into o_price

  if ROWSTATUS != FOUND then { BREAK LOOP }
endif
  if n > 50 then { BREAK LOOP } endif

  execute { call acid_single_query (^) } substitute ord into
o_total

  fetch {select cast(^ as numeric(12,2)) } substitute o_price
into o_price
  fetch {select cast(^ as numeric(12,2)) } substitute o_total
into l_total

  print 'orderkey = ', ord, '          o_totalprice = ', o_price,
        '          acid query = ' , l_total

  ASSERT (o_price = l_total)
        then { print 'Did not compare correctly' }
ENDASSERT
  let n=n+1

 } ENDLOOP

disconnect

END Test
```

C.14 acid_durability_setup.tst

```
Test            "acid_durability_setup.tst"
Description     "Creates acid_table table"

stringconnect "dsn=qual_15_0;"

execute {set option public.query_plan='off'}
execute {set temporary option chained='on'}
execute {set option public.auto_commit=off}
execute { set option public.isolation_level=3 }

% Drop Table if found
allow error -141
execute { drop table acid_table }
allow no error

execute {
create table acid_table (
                stream  int     not null,
        seqnum  int     not null,
        ordr    int     null,
        line    int     null,
        delta   int     null)
on SYSTEM
}
```

```
fetch {select count(*) from acid_table } into ROWS
assert ROWS = 0
print 'Number of rows before load: ',ROWS
commit

print 'acid_table created'

allow error -141
execute { drop table latest }
allow no error

execute {create table latest(stream int ,last int null) on
SYSTEM }
LOOP ({let j = 1}; {j <= 10}; {let j = j + 1})
   {
          execute { insert into latest(stream,last) values (^,0) }
                substitute j
   } endloop
commit

print 'latest created'

LOOP ({let i = 1}; {i <= 10}; { let i = i + 1})
{
  LOOP ({let j = 1}; {j <= 200}; { let j = j + 1})
   {
          execute { call generate_acid_values()} into ordr, line,
delta
          execute { insert into acid_table values (^,^,^,^,^) }
                substitute i,j,ordr,line,delta
   } endloop
  print (j-1)*i
} endloop
commit

fetch {select count(*) from acid_table } into ROWS
print 'Number of rows after load: ',ROWS

End Test
```

C.15 acid_durability_txn.tst

```
Test            "tpcd_transaction1.tst"
Description     "Run Acid Multiple Transactions"

stringconnect "dsn=qual_15_0;"
execute {select now(*)} into times
print 'Durability test start = ', times
print ' '
print 'stream      trans.    o_key    l_key    p_key    s_key
          delta     date_t '
allow no error

let commit_delay=0
LOOP ({let i = 1}; {i <= 200}; { let i = i + 1})
{
 fetch {select ordr, line, delta from acid_table
          where stream=^ and seqnum=^ }
          substitute stream, i
 commit

 if ROWSTATUS != FOUND then { print 'not enough rows'
BREAK LOOP }
 endif

 if i=101 then {
    let smallest=0
    allow error -210
          commit
    execute { set temporary option isolation_level=1 }
    execute { select min(last) from latest } into smallest
    execute { set temporary option isolation_level=3 }
    commit
    LOOP{
          if smallest >= 100 then {
                print 'Stream ', stream,
                      ' Entering the Second
phase with delays'
                break loop}
          endif
          let sleep_time =10
          sleep sleep_time
              commit
              execute { set temporary option isolation_level=1 }
          execute { select min(last) from latest } into smallest
              execute { set temporary option isolation_level=3 }
              commit
    }
    ENDLOOP
    allow no error
    let commit_delay=20}
 endif


 %- Sometimes we have plans on, so just to make sure the
message file
 %- does not get huge...
 execute {set temporary option query_plan='off'}

% execute {select l_partkey, l_suppkey from lineitem
%          where l_orderkey=^ and l_linenumber=^}
%          substitute ordr, line
%          into p_key, s_key
execute{SELECT @@spid}into spid
allow error
LOOP {
  execute{begin transaction}
  execute{select Txnid from sp_iqtransaction() WHERE
ConnHandle=^ and state='ACTIVE'}substitute spid
    into newTxnId
 print 'New transactionid=',newTxnId
```

```
  execute {call acid_transaction( ^, ^, ^)
  } substitute ordr, line, delta
  into rprice, quantity, tax, disc,
extprice,ototal,TxnId,p_key,s_key

  if SQLCODE=0 then
  { break loop}
  else
  {
     execute{rollback}
     let sleeping_time=rand(625,6653)
     sleep sleeping_time
  }
  endif
}
ENDLOOP
allow no error

 print 'transaction=',TxnId
  print  'before commiting ',
            stream,'  ',
            'txn ',i, '  ',
        ordr,  '          ',
        line,  '          ',
        p_key, '          ',
        s_key, '          ',
        delta

  execute {call commit_acid_transaction(^)}substitute
commit_delay
  execute {select now(*)} into times
  print  'after commit', stream,'          ',
            'txn ',i, '  ',
        ordr,  '          ',
        line,  '          ',
        p_key, '          ',
        s_key, '          ',
        delta, '          ',
        times, '          '
commit
 execute { update latest set last=^ where stream=^ } substitute
i,stream
 commit
 execute { set temporary option isolation_level=1 }
 execute { select max(last) from latest } into biggest
 execute { select min(last) from latest } into smallest
 commit
 execute { set temporary option isolation_level=3 }

 let num=120*(i-smallest)
 if i+4>=biggest
 then {let num=num+800}
 endif
--print 'user',stream,' = ',num
 sleep num

}
ENDLOOP
print 'Out of loop. Child waiting for synch'
synchronize 11


End Test
```

C.16 run_isolation

```
#!/bin/ksh

cd $ACID_ROOT/isolation/isolation_1
dbtest acid_isolation_main1.tst >
$ACID_RESULTS/iso1.`date '+%y%m%d_%H%M%S'`
sleep 200

cd $ACID_ROOT/isolation/isolation_2
dbtest acid_isolation_main2.tst >
$ACID_RESULTS/iso2.`date '+%y%m%d_%H%M%S'`
sleep 200

cd $ACID_ROOT/isolation/isolation_3
dbtest acid_isolation_main3.tst >
$ACID_RESULTS/iso3.`date '+%y%m%d_%H%M%S'`
sleep 200

cd $ACID_ROOT/isolation/isolation_4
dbtest acid_isolation_main4.tst >
$ACID_RESULTS/iso4.`date '+%y%m%d_%H%M%S'`
sleep 200

cd $ACID_ROOT/isolation/isolation_5
dbtest
$ACID_ROOT/isolation/isolation_5/acid_isolation_main5.ts
t > $ACID_RESULTS/iso5.`date '+%y%m%d_%H%M%S'`
sleep 200

cd $ACID_ROOT/isolation/isolation_6
dbtest
$ACID_ROOT/isolation/isolation_6/acid_isolation_main6.ts
t > $ACID_RESULTS/iso6.`date '+%y%m%d_%H%M%S'`
sleep 200
```

C.17 acid_isolation_main1.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%  Created by: Masood Dirin
%  Created Date: 5/24/1999
%  Script name: tpcd_acid_isolation_main1.tst
%  ---------------------------------------------------------------------
%
%  Purpose of this test:
%  This test will run the first isolation test, which demonstrate
```

```
% isolation for the read-write conflict of a read-write transaction
% and a read-only transaction when the read-write transaction is commited.
% Run the test as follow:
%
% dbtest tpcd_acid_isolation_main1.tst > tpcd_acid_isolation_main1.ot
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Test		"tpch_acid_isolation_main1.tst"
Description	"To run the ACID isolation test1"

stringconnect "dsn=qual_15_0;"

execute {select now(*)} into times
print ' '
print ' '
print 'Isolation test 1'
print 'start = ', times
print ' '

include 'acid_functions.tst'
include 'acid_isolation_setup.tst'

start test  'acid_isolation_test1.tst'
start test  'acid_isolation_test1_query.tst'

End Test
```

C.18 acid_isolation_main2.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created by: Masood Dirin

% Created Date: 5/24/1999

% Script name: tpcd_acid_isolation_main2.tst
% ----------------------------------------------------------------------
% Purpose of this test:
% This test will run the second isolation test, which demonstrate
% isolation for the read-write conflict of a read-write transaction
% and a read-only transaction when the read-write transaction is
% rolled back.                                    %
% Run the test as follow:

%
% dbtest tpcd_acid_isolation_main2.tst > tpcd_acid_isolation_main2.ot
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Test		"tpcd_acid_isolation_main2.tst"
Description	"To run the ACID isolation test2"

stringconnect "dsn=qual_15_0;"

execute {select now(*)} into times
print ' '
print ' '
print 'Isolation test 2'
print 'start = ', times
print ' '


include 'acid_functions.tst'
include 'acid_isolation_setup.tst'

start test  'acid_isolation_test2.tst'
start test  'acid_isolation_test2_query.tst'

End Test
```

C.19 acid_isolation_main3.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created by: Masood Dirin
% Created Date: 5/24/1999
% Script name: tpcd_acid_isolation_main3.tst
% --------------------------------------------------------------------
% Purpose of this test:
% This test will run the third Acid isolation test, which
% demonstrate isolation for the write-write conflict of two
% update transactions when the first transaction is commited.
%
% Run the test as follow:
%
% dbtest tpcd_acid_isolation_main3.tst > tpcd_acid_isolation_main3.ot
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Test		"tpcd_acid_isolation_main3.tst"
Description	"To run the ACID isolation test3"

stringconnect "dsn=qual_15_0;"

execute {select now(*)} into times
print ' '
```

```
print ' '
print 'Isolation test 3'
print 'start = ', times
print ' '
print 'Isolation test start = ', times

include "acid_functions.tst"
include 'acid_isolation_setup.tst'

start test  'acid_isolation_test3_transaction1.tst'
start test  'acid_isolation_test3_transaction2.tst'

End Test
```

C.20 acid_isolation_main4.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
%  Created by: Masood Dirin
%  Created Date: 5/24/1999
%  Script name: tpcd_acid_isolation_main4.tst
%  ------------------------------------------------------------------
%  Purpose of this test:
%  This test will run the fourth Acid isolation test, which
%  demonstrate isolation for the write-write conflict of two
%  update transactions when the first transaction is rolled back.
%
%  Run the test as follow:
%
%  dbtest tpcd_acid_isolation_main4.tst >
tpcd_acid_isolation_main4.ot
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%

Test           "tpcd_acid_isolation_main4.tst"
Description     "To run the ACID isolation test4"

stringconnect "dsn=qual_15_0;"

execute {select now(*)} into times
print ' '
print ' '
print 'Isolation test 4'
print 'start = ', times
print ' '
print 'Isolation test start = ', times

include 'acid_functions.tst'
include 'acid_isolation_setup.tst'

start test  'acid_isolation_test4_transaction1.tst'
start test  'acid_isolation_test4_transaction2.tst'

End Test
```

C.21 acid_isolation_main5.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
%  Created by: Masood Dirin
%  Created Date: 5/27/1999
%  Script name: tpcd_acid_isolation_main5.tst
%  ------------------------------------------------------------------
%  Purpose of this test:
%  This test will run isolation test 5 and will demonstrate
%  the ability of read and write transactions affecting
different
%  database tables to make progress concurrently.
%
%  Run the test as follow:
%
%  dbtest tpcd_acid_isolation_main5.tst >
tpcd_acid_isolation_main5.ot
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%


Test           "tpcd_acid_isolation_main5.tst"
Description     "To run the ACID isolation test5."

stringconnect "dsn=qual_15_0;"

execute {select now(*)} into times
print ' '
print ' '
print 'Isolation test 5'
print 'start = ', times
print ' '

include 'acid_functions.tst'
include 'acid_isolation_setup.tst'

start test 'acid_isolation_test5_transaction1.tst'
start test 'acid_isolation_test5_query.tst'

End Test
```

C.22 acid_isolation_main6.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%  Created by: Masood Dirin
%  Created Date: 5/27/1999
%  Script name: tpcd_acid_isolation_main6.tst
```

```
% ------------------------------------------------------------------
-
%  Run the test as follow:
%
%  dbtest -u tpcd_acid_isolation_main6.tst >
tpcd_acid_isolation_main6.ot
%
%  Note:  -u switch will be used to archive the User1 query
result
%  in a file named queryresult.cfr. This switch needs to be
used each time
%  the test being run, since the query results will be different
as the
%  results of the updates on the lineitem tables.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%

Test            "tpcd_acid_isolation_main6.tst"
Description     "To run the ACID isolation test6."

stringconnect "dsn=qual_15_0;"

execute {select now(*)} into times
print ' '
print ' '
print 'Isolation test 6'
print 'start = ', times
print ' '

include 'acid_functions.tst'
include 'acid_isolation_setup.tst'

start test '-u' 'acid_isolation_test6_query.tst'
start test  'acid_isolation_test6_transaction1.tst'

End Test
```

C.23 acid_isolation_setup.tst

```
Test                "acid_isolation_setup.tst"
Description         "Creates acid_isolation_table table"

stringconnect "dsn=qual_15_0;"

% Drop Table if found

allow error -141
execute { commit }
execute { drop table acid_isolation_table }
allow no error

execute {
create table acid_isolation_table (
                ordr        int    not null,
                line        int    null,
                delta       int    null)
}

execute {checkpoint}

print 'acid_isolation_table CREATED!!'
execute {select now(*)} into times
print 'time = ', times

fetch {select count(*) from acid_isolation_table } into
ROWS
assert ROWS = 0

print 'Number of rows before load: ',ROWS

execute {call generate_acid_values()} into orderkey,
linenumber,delta
execute {insert into acid_isolation_table values ( ^ , ^ , ^ ) }
                        substitute orderkey, linenumber, delta
print  orderkey, ' ',linenumber,' ', delta

commit

fetch {select count(*) from acid_isolation_table } into
ROWS
assert ROWS = 1

print 'Number of rows after load: ',ROWS

disconnect

End Test
```

C.24 acid_isolation_test1_query.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Created by: Masood Dirin
%  Created Date: 5/24/1999
%  Script name: tpcd_acid_query_isolation_test1.tst
%  ----------------------------------------------------
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Test 'tpch_acid_query_isolation_test1'
Description 'perform the acid query for user2.'

stringconnect "dsn=qual_15_0;"

 synchronize 2
```

```
 print ' '
 execute {select now(*)} into times
 print 'User 2 start query = ', times

 execute {select ordr from acid_isolation_table}
       into ordr

 print 'user 2 ordr = ', ordr
 execute { call acid_single_query (^) } substitute ordr into
o_total
 print 'user 2 o_total=' , o_total
 print ' '

execute {select now(*)} into times
print 'User 2 completed query = ', times

disconnect

END Test
```

C.25 acid_isolation_test1.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%  Created by: Masood Dirin
%  Created Date: 5/24/1999
%  Script name: tpcd_acid_isolation_test1.tst

%
%  Part of tpcd_acid_isolation_main1.tst
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%

Test            "tpch_aci_isolation_test1.tst"
Description     "Run Acid isolation test 1"

stringconnect "dsn=qual_15_0;"

execute {select ordr, line, delta from acid_isolation_table}
       into ordr, line, delta

execute { select round(cast(o_totalprice as numeric(18,2)),2)
       from orders where o_orderkey = ^}
       substitute ordr into o_total
print 'User 1 old values: '
print 'user 1 ordr= ', ordr
print 'user 1 o_total= ', o_total
print ' '
 print 'The following are the data input values for the ACID
Transaction.'
 print '(user 1) o_key-',ordr, '  l_key-', line, '  delta-',delta
 execute {call acid_transaction( ^, ^,  ^)
            } substitute ordr, line, delta into rprice, quantity,
tax, disc, extprice, ototal

execute {select now(*)} into times
print 'User 1 waiting to commit = ', times
print ' '
synchronize 2
sleep 10000
execute {select now(*)} into times
print 'User 1 about to commit = ', times
commit
execute {select now(*)} into times
print 'User 1 has committed = ', times

execute { select round(cast(o_totalprice as numeric(18,2)),2)
            from orders where o_orderkey = ^}
            substitute ordr into o_total
print 'User 1 new values: '
print 'user 1 ordr= ', ordr
print 'user 1 o_total= ', o_total
print ' '

End Test
```

C.26 acid_isolation_test2_query.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Created by: Masood Dirin
%  Created Date: 5/24/1999
%  Script name: tpcd_acid_query_isolation_test1.tst
%  ----------------------------------------------------
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Test 'tpcd_acid_query_isolation_test1'
Description 'perform the acid query for user2.'

stringconnect "dsn=qual_15_0;"

 synchronize 2
 print ' '
 execute {select now(*)} into times
 print 'User 2 start query = ', times

 execute {select ordr from acid_isolation_table}
       into ordr

 print 'user 2 ordr = ', ordr
```

```
 execute { call acid_single_query (^) } substitute ordr into
o_total
 print 'user 2 o_total=' , o_total
 print ' '

execute {select now(*)} into times
print 'User 2 completed query = ', times

disconnect

END Test
```

C.27 acid_isolation_test2.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%  Created by: Masood Dirin
%  Created Date: 5/24/1999
%  Script name: tpcd_acid_isolation_test1.tst

%
%  Part of tpcd_acid_isolation_main1.tst
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%

Test                "tpcd_acid_isolation_test1.tst"
Description         "Run Acid isolation test 1"

stringconnect "dsn=qual_15_0;"

execute {select ordr, line, delta from acid_isolation_table}
       into ordr, line, delta
 print ''
 print 'The following are the data input values for the ACID
Transaction.'
 print '(user 1) o_key-',ordr, '  l_key-', line, '  delta-',delta

 execute { select o_totalprice from orders where o_orderkey
= ^}
         substitute ordr into o_total
 print 'Before user1 acid transaction o_total=',o_total
 print ''
 execute {call acid_transaction( ^, ^,  ^,
            rprice, quantity, tax, disc, extprice, ototal)
            } substitute ordr, line, delta

execute {select now(*)} into times
print 'User 1 waiting to roll back = ', times
print ' '
synchronize 2
sleep 10000
execute {select now(*)} into times
print 'User 1 about to roll back = ', times
rollback

execute { select round(cast(o_totalprice as numeric(18,2)),2)
            from orders where o_orderkey = ^}
            substitute ordr into o_total
print 'User 1 new values: '
print 'user 1 ordr= ', ordr
print 'user 1 o_total= ', o_total
print ' '

End Test
```

C.28 acid_isolation_test3_transaction1.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
%  Created by: Masood Dirin                                  %
%  Created Date: 5/25/1999                                   %
%  Script name: tpcd_acid_isolation_test3_transaction1.tst
%
%  -----------------------------------------------------------------
%
%                                                        %
%  This test could be run by itself, but it is recommended to
run it as   %
%  part of tpcd_acid_isolation_main3.tst file.
%
%                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

Test                "acid_isolation_test3_transaction1.tst"
Description         "Run Acid Transaction 1 for isolation test
3"

stringconnect "dsn=qual_15_0;"

execute {select now(*)} into times
print 'Isolation test 3 test start = ', times
print ' '

 execute {select ordr, line, delta from acid_isolation_table}
       into ordr, line, delta

 print 'User 1 --  The input data values for User 1 Acid
Transaction.'
 print 'User 1 -- o_key = ',ordr
 print 'User 1 -- l_key = ',line
 print 'User 1 -- delta1 = ',delta
```

```
 print ' '
 execute {select now(*)} into times
 print 'User 1 -- Starting the Acid Transaction: ', times

 execute {call acid_transaction( ^, ^,  ^ )}
            substitute ordr, line, delta
            into rprice, quantity, tax, disc, extprice, ototal

 print ' '
 execute {select now(*)} into times
 print 'User 1 -- Acid Transaction complete: ', times
 print '30 second timer started'
SYNCHRONIZE 2
 sleep 30000

 print ' '
 execute {select now(*)} into times
 print 'User 1 -- starting commit: ', times

 commit
 print ' '
 execute {select now(*)} into times
 print 'User 1 -- transaction commit complete: ', times

 print ' '
 print 'USER 1 -- original extendedprice = ', extprice
 print 'USER 1 -- original quantity = ', quantity

 fetch { select cast(^ as numeric(18,6))
                      + (cast(^ as numeric(18,6))*(cast (^ as
numeric(18,6))
                      /cast (^ as numeric(18,6)))) }
            substitute extprice, delta, extprice, quantity
            into result1
 % make it format nicely...
 execute { select cast(^ as numeric(18,2)) } substitute result1
into result2

 print ' '
 print 'User 1 -- result1 = '
 print '    txn1_extendedprice + (delta1 *
(txn1_extendedprice/txn1_quantity))'
 print 'User 1 -- result1= ', result2
 print ' '

 disconnect
 End Test
```

C.29 acid_isolation_test3_transaction2.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%  Created by: Masood Dirin                              %
%  Created Date: 5/25/1999                               %
%  Script name: tpcd_acid_isolation_test3_transaction2.tst
%
%  -----------------------------------------------------------------
%
%                                                        %
%  This test could be run by itself, but it is recommended to
run it as  %
%  part of tpcd_acid_isolation_main3.tst  file.
%
%                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%


Test                "acid_isolation_test3_transaction2.tst"
Description         "Run Acid Transaction 2 for isolation test
3"

stringconnect "dsn=qual_15_0;"

execute {select ordr, line, delta from acid_isolation_table}
       into ordr, line, delta
% generate a new set of values; we only use delta2
execute { call generate_acid_values()} into ordr2, line2,
delta2

 print ' '
 print 'User 2 - The input data values for the Acid
Transaction.'
 print 'User 2 -- o_key = ',ordr
 print 'User 2 -- l_key= ',line
 print 'User 2 -- delta2 = ',delta2

SYNCHRONIZE 2

 print ' '
 execute {select now(*)} into times
 print 'User 2 -- Starting the Acid Transaction: ', times

execute {call acid_transaction( ^, ^,  ^ ) }
            substitute ordr, line, delta2
            into rprice, quantity, tax, disc, extprice, ototal
execute {select round(cast(^ as numeric(20,6)),2) }
            substitute extprice into extprice2

print ' '
execute {select now(*)} into times
print 'User 2 -- About to commit: ', times
commit

execute {select now(*)} into times
print 'User 2 -- transaction commit complete: ', times
```

```
print ' '

print 'USER 2 -- original extendedprice = ', extprice2
print 'USER 2 -- original quantity = ', quantity
print ' '

 fetch { select cast(^ as numeric(18,6))
              + (cast(^ as numeric(18,6))*(cast (^ as
numeric(18,6))
              /cast (^ as numeric(18,6)))) }
       substitute extprice, delta, extprice, quantity
       into result1
 % make it format nicely...
 execute { select cast(^ as numeric(18,2)) } substitute result1
into result2

 print ' '
 print 'User 2 -- result1 = '
 print '    txn2_extendedprice + (delta2 *
(txn2_extendedprice/txn2_quantity))'
 print 'User 2 -- result1= ', result2
 print ' '

End Test
```

C.30 acid_isolation_test4_transaction1.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
%  Created by: Masood Dirin                                   %
%  Created Date: 5/25/1999                                    %
%  Script name: tpcd_acid_isolation_test3_transaction1.tst
%
%  ------------------------------------------------------------------
%
%                                                             %
%  This test could be run by itself, but it is recommended to
run it as   %
%  part of tpcd_acid_isolation_main3.tst file.
%
%                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

 Test                "acid_isolation_test4_transaction1.tst"
Description          "Transaction 1 for isolation test 4"

stringconnect "dsn=qual_15_0;"

execute {select now(*)} into times
print 'Isolation test 3 test start = ', times
print ' '

 execute {select ordr, line, delta from acid_isolation_table}
       into ordr, line, delta

 print 'User 1 --  The input data values for User 1 Acid
Transaction.'
 print 'User 1 -- o_key = ',ordr
 print 'User 1 -- l_key = ',line
 print 'User 1 -- delta1 = ',delta

 print ' '
 execute {select now(*)} into times
 print 'User 1 -- Starting the Acid Transaction: ', times

 execute {select l_extendedprice from lineitem where
l_linenumber=^ and l_orderkey=^}
            substitute line, ordr into extprice3

execute {select round(cast(^ as numeric(20,6)),2) }
       substitute extprice3 into extprice4
 print ' '
 print 'USER 1 -- extendedprice before acid transaction = ',
extprice4

 execute {call acid_transaction( ^, ^,  ^ )}
            substitute ordr, line, delta
            into rprice, quantity, tax, disc, extprice, ototal

 print ' '
 execute {select now(*)} into times
 print 'User 1 -- Acid Transaction complete: ', times
 print '30 second timer started'
SYNCHRONIZE
 sleep 30000

execute {select l_extendedprice from lineitem where
l_linenumber=^ and l_orderkey=^}
            substitute line, ordr into extprice3

execute {select round(cast(^ as numeric(20,6)),2) }
       substitute extprice3 into extprice4
 print ' '
 print 'USER 1 -- extendedprice before rooling back = ',
extprice4
 print ' '
 execute {select now(*)} into times
 print 'User 1 -- starting rollback: ', times

 rollback
 print ' '
 execute {select now(*)} into times
 print 'User 1 -- transaction rollback complete: ', times
```

```
 execute {select round(cast(^ as numeric(20,6)),2) }
             substitute extprice into extprice2
 print ' '
 print 'USER 1 -- original extendedprice = ', extprice2
 print 'USER 1 -- original quantity = ', quantity
 print ' '

 disconnect
 End Test
```

C.31 acid_isolation_test4_transaction2.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Created by: Masood Dirin                                    %
%  Created Date: 5/25/1999                                     %
%  Script name: tpcd_acid_isolation_test3_transaction2.tst
%
%  -------------------------------------------------------------------
%
%                                                        %
%  This test could be run by itself, but it is recommended to
run it as  %
%  part of tpcd_acid_isolation_main3.tst  file.
%
%                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Test            "acid_isolation_test4_transaction2.tst"
Description     "Transaction 2 for isolation test 4"

stringconnect "dsn=qual_15_0;"

execute {select ordr, line, delta from acid_isolation_table}
       into ordr, line, delta
% generate a new set of values; we only use delta2
execute { call generate_acid_values()} into ordr2, line2,
delta2

 print ' '
 print 'User 2 - The input data values for the Acid
Transaction.'
 print 'User 2 -- o_key = ',ordr
 print 'User 2 -- l_key= ',line
 print 'User 2 -- delta2 = ',delta2

SYNCHRONIZE 2

 print ' '
 execute {select now(*)} into times
 print 'User 2 -- Starting the Acid Transaction: ', times

execute {call acid_transaction( ^, ^, ^ ) }
            substitute ordr, line, delta2
            into rprice, quantity, tax, disc, extprice, ototal
execute {select round(cast(^ as numeric(20,6)),2) }
            substitute extprice into extprice2

print ' '
execute {select now(*)} into times
print 'User 2 -- About to commit: ', times
commit

execute {select now(*)} into times
print 'User 2 -- transaction commit complete: ', times
print ' '
print 'USER 2 -- original extendedprice = ', extprice2
print 'USER 2 -- original quantity = ', quantity
print ' '

 fetch { select cast(^ as numeric(18,6))
              + (cast(^ as numeric(18,6))*(cast (^ as
numeric(18,6))
              /cast (^ as numeric(18,6)))) }
       substitute extprice, delta2, extprice, quantity
       into result1
 % make it format nicely...
 execute { select cast(^ as numeric(18,2)) } substitute result1
into result2

 print ' '
 print 'User 2 -- result1 = '
 print '    txn2_extendedprice + (delta2 *
(txn2_extendedprice/txn2_quantity))'
 print 'User 2 -- result1= ', result2
 print ' '

End Test
```

C.32 acid_isolation_test5_query.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Created by: Masood Dirin
%  Created Date: 5/27/1999
%  Script name: tpcd_acid_isolation_query_test5.tst
%  -------------------------------------------------------------
%
%  This test could be run by itself, but it is recommended to
run
% it as part of tpcd_acid_isolation_main5.tst file.
%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%

Test                "tpcd_acid_isolation_query_test5.tst"
Description         "Run Acid isolation query for test 5"

stringconnect "dsn=qual_15_0;"

synchronize 2

execute { call generate_ps_values() } into ps_ptky, ps_spky
print ' '
print 'user 2 ps_partkey = ', ps_ptky
print 'user 2 ps_suppkey = ', ps_spky
print ' '

execute {select now(*)} into times
print 'User 2 beginning query = ', times
execute {select * from partsupp where ps_partkey=^ and ps_suppkey=^}
        substitute ps_ptky, ps_spky
        into ps_ptky, ps_spky, ps_aly, ps_spct, ps_ct

print ' '
print 'User2 gets all columns of the PARTSUPP table '
print ' for selected ps_partkey and ps_suppkey doing a query.'
print ' '
print 'ps_partkey = ', ps_ptky, '      ps_suppkey = ', ps_spky
print 'ps_availqty = ', ps_aly, '      ps_supplycost = ',ps_spct
print 'ps_comment = ', ps_ct
execute {select now(*)} into times
print 'User 2 query complete = ', times
print ' '

execute {select now(*)} into times
print 'User 2 about to commit = ', times
commit
execute {select now(*)} into times
print 'User 2 transaction commit complete = ', times

print ' '

End Test
```

C.33 acid_isolation_test5_transaction1.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
%  Created by: Masood Dirin                               %
%  Created Date: 5/27/1999                               %
%  Script name: tpcd_acid_isolation_test5_transaction1.tst
%
%  ------------------------------------------------------------------
%
%                                                        %
%  This test could be run by itself, but it is recommended to run it      %
%  as part of tpcd_acid_isolation_main5.tst file.
%
%                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

Test
        "tpcd_acid_isolation_test5_transaction1.tst"
Description       "Run Acid isolation for user1 on test5."

stringconnect "dsn=qual_15_0;"


execute {select ordr, line, delta from acid_isolation_table}
    into ordr, line, delta

 print ' '
 print 'The following are the input values for the users1 ACID Transaction.'
 print 'o_key = ',ordr,'      l_key = ',line,'     delta = ',delta
 print ''
execute {select now(*)} into times
print 'User 1 isolation test time = ', times
print ' '
print ' '
execute {select o_totalprice from orders where o_orderkey=^ }
        substitute ordr into o_tprice
execute {select l_extendedprice, l_quantity,l_partkey, l_suppkey
        from lineitem
        where l_orderkey=^ and l_linenumber=^}
        substitute ordr, line
        into l_price, l_quant, l_ptky, l_spky
print 'User 1 o_totalprice = ', o_tprice
print 'User 1 l_extendedprice = ', l_price,'  l_quantity = ', l_quant
print 'User 1 l_partkey      = ', l_ptky,'       l_suppkey = ', l_spky
print ' '

execute {select now(*)} into times
print 'User 1 starting acid transaction = ', times

execute {call acid_transaction( ^, ^,  ^, rprice, quantity, tax, disc,
        extprice, ototal) } substitute ordr, line, delta
```

```
execute {select now(*)} into times
print 'User 1 waiting to commit = ', times
print ' '
synchronize 2
sleep 10000
execute {select now(*)} into times
print 'User 1 about to commit = ', times
commit
execute {select now(*)} into times
print 'User 1 transaction commit complete = ', times

execute {select o_totalprice from orders where
o_orderkey=^ }
        substitute ordr into o_tprice
execute {select l_extendedprice, l_quantity
        from lineitem where l_orderkey=^ and
l_linenumber=^}
        substitute ordr, line
        into l_price, l_quant
print 'User 1 o_totalprice = ', o_tprice
print 'User 1 l_extendedprice = ', l_price,'  l_quantity = ',
l_quant
print 'User 1 l_partkey      = ', l_ptky,'      l_suppkey = ',
l_spky

print ' '
execute {select * from history where h_o_key=^
        and h_date_t=(select max(h_date_t) from history
where h_o_key=^)}
        substitute ordr, ordr
        into hpk, hsk, hok, hlk, hda, hdt

print 'User 1 history entry:'
print '  h_p_key = ', hpk
print '  h_s_key = ', hsk
print '  h_o_key = ', hok
print '  h_l_key = ', hlk
print '  h_delta = ', hda
print '  h_date_t = ', hdt

execute {select now(*)} into times
print 'User 1 isolation test time = ', times
print ''

End Test
```

C.34 acid_isolation_test6_query.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
% Created by: Masood Dirin                                   %
% Created Date: 5/27/1999                                   %
% Script name: tpcd_acid_isolation_test6_transaction1.tst
%
% -------------------------------------------------------------------
%
%                                                    %
% This test could be run by itself, but it is recommended to
run it      %
% as part of tpcd_acid_isolation_main6.tst file.
%
%                                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

Test
        "tpcd_acid_isolation_test6_transaction1.tst"
Description        "Run Acid isolation for user2 on test6."

stringconnect "dsn=qual_15_0;"

execute {select ordr, line, delta from acid_isolation_table}
        into ordr, line, delta

 execute {select now(*)} into qstart2
 print 'User2 acid Transaction = ', qstart2
 print 'o_key = ',ordr, '     l_key = ',line, '    delta = ',delta
 print ' '
execute {select o_totalprice from orders where
o_orderkey=^ }
        substitute ordr into o_tprice
execute {select l_extendedprice, l_quantity,l_partkey,
l_suppkey
        from lineitem where l_orderkey=^ and
l_linenumber=^}
        substitute ordr, line
        into l_price, l_quant, l_ptky, l_spky
print 'User 2 o_totalprice = ', o_tprice
print 'User 2 l_extendedprice = ', l_price,'    l_quantity = ',
l_quant
print 'User 2 l_partkey      = ', l_ptky,'     l_suppkey = ',
l_spky
print ' '

execute {select now(*)} into qstart2
print 'Start Time for User2 Transaction = ', qstart2
print ' '
execute {call acid_transaction( ^, ^,  ^, rprice, quantity,
                                        tax, disc, extprice, ototal)
}
        substitute ordr, line, delta

execute {select now(*)} into qstop2
print 'User 2 about to commit = ', qstop2
```

```
commit
execute {select now(*)} into qstop2
print 'User 2 transaction commit complete = ', qstop2
print ' '

execute {select o_totalprice from orders where
o_orderkey=^ }
            substitute ordr
            into o_tprice
execute {select l_extendedprice, l_quantity
            from lineitem where l_orderkey=^ and
l_linenumber=^}
            substitute ordr, line
            into l_price, l_quant
print 'User 2 o_totalprice = ', o_tprice
print 'User 2 l_extendedprice = ', l_price,'    l_quantity = ',
l_quant
print 'User 2 l_partkey       = ', l_ptky,'     l_suppkey = ',
l_spky
print ' '

print ' '
execute {select * from history
            where h_o_key=^
            and h_date_t=(select max(h_date_t) from history
where h_o_key=^)}
            substitute ordr, ordr
            into hpk, hsk, hok, hlk, hda, hdt

print 'User 2 history entry:'
print '  h_p_key = ', hpk
print '  h_s_key = ', hsk
print '  h_o_key = ', hok
print '  h_l_key = ', hlk
print '  h_delta = ', hda
print '  h_date_t = ', hdt

print ' '
execute {select now(*)} into times
print 'User 2 completed = ', times

End Test
```

C.35 acid_isolation_test6_transaction1.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
%  Created by: Masood Dirin                                    %
%  Created Date: 5/27/1999                                     %
%  Script name: tpcd_acid_isolation_test6_transaction1.tst
%
%  -----------------------------------------------------------------
%
%                                                              %
%  This test could be run by itself, but it is recommended to
run it      %
%  as part of tpcd_acid_isolation_main6.tst file.
%
%                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

Test
            "tpcd_acid_isolation_test6_transaction1.tst"
Description         "Run Acid isolation for user2 on test6."

stringconnect "dsn=qual_15_0;"

execute {select ordr, line, delta from acid_isolation_table}
            into ordr, line, delta

 execute {select now(*)} into qstart2
 print 'User2 acid Transaction = ', qstart2
 print 'o_key = ',ordr, '      l_key = ',line, '    delta = ',delta
 print ' '
execute {select o_totalprice from orders where
o_orderkey=^ }
            substitute ordr into o_tprice
execute {select l_extendedprice, l_quantity,l_partkey,
l_suppkey
            from lineitem where l_orderkey=^ and
l_linenumber=^}
            substitute ordr, line
            into l_price, l_quant, l_ptky, l_spky
print 'User 2 o_totalprice = ', o_tprice
print 'User 2 l_extendedprice = ', l_price,'    l_quantity = ',
l_quant
print 'User 2 l_partkey       = ', l_ptky,'     l_suppkey = ',
l_spky
print ' '

execute {select now(*)} into qstart2
print 'Start Time for User2 Transaction = ', qstart2
print ' '
execute {call acid_transaction( ^, ^,  ^, rprice, quantity,
                                        tax, disc, extprice, ototal)
}
            substitute ordr, line, delta

execute {select now(*)} into qstop2
print 'User 2 about to commit = ', qstop2
commit
execute {select now(*)} into qstop2
print 'User 2 transaction commit complete = ', qstop2
print ' '
```

```
execute {select o_totalprice from orders where
o_orderkey=^ }
        substitute ordr
        into o_tprice
execute {select l_extendedprice, l_quantity
        from lineitem where l_orderkey=^ and
l_linenumber=^}
        substitute ordr, line
        into l_price, l_quant
print 'User 2 o_totalprice = ', o_tprice
print 'User 2 l_extendedprice = ', l_price,'    l_quantity = ',
l_quant
print 'User 2 l_partkey       = ', l_ptky,'     l_suppkey = ',
l_spky
print ' '

print ' '
execute {select * from history
        where h_o_key=^
        and h_date_t=(select max(h_date_t) from history
where h_o_key=^)}
        substitute ordr, ordr
        into hpk, hsk, hok, hlk, hda, hdt

print 'User 2 history entry:'
print '  h_p_key = ', hpk
print '  h_s_key = ', hsk
print '  h_o_key = ', hok
print '  h_l_key = ', hlk
print '  h_delta = ', hda
print '  h_date_t = ', hdt

print ' '
execute {select now(*)} into times
print 'User 2 completed = ', times

End Test
```

Appendix D Query text and Output

::::::::::::::
qryqual01
::::::::::::::
Query Text

```
9> select
10>         l_returnflag,
11>         l_linestatus,
12>         sum(l_quantity) as sum_qty,
13>         sum(l_extendedprice) as sum_base_price,
14>         sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
15>         sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as
sum_charge,
16>         avg(l_quantity) as avg_qty,
17>         avg(l_extendedprice) as avg_price,
18>         avg(l_discount) as avg_disc,
19>         count(*) as count_order
20> from
21>         lineitem
22> where
23>         l_shipdate <= dateadd(day, -90, '1998-12-01')
24> group by
25>         l_returnflag,
26>         l_linestatus
27> order by
28>         l_returnflag,
29>         l_linestatus
30>
```

Query Result

```
l_returnflag l_linestatus sum_qty                 sum_base_price
sum_disc_price            sum_charge              avg_qty
avg_price                 avg_disc                count_order
------------ ------------ ------------------------ ------------------------ ----
------------------------- ------------------------ ------------------------ -----
------------------------- ------------------------ --------------------
A            F                37734107.000000
56586554400.729591          53758257134.869957
55909065222.827705                25.522006               38273.129735
0.049985               1478493
N            F                  991417.000000          1487504710.379997
1413082168.054108           1469649223.194361                  25.516472
38284.467761                0.050093                38854
N            O                74476040.000000
111701729697.740601         106118230307.603668
110367043872.498001               25.502227               38249.117989
0.049997               2920374
R            F                37719753.000000
56568041380.899521          53741292684.603989
55889619119.831291                25.505794               38250.854626
0.050009               1478870

(4 rows affected)
```

::::::::::::::
qryqual02
::::::::::::::
Query Text

```
3>
4> select top 100
5>         s_acctbal,
6>         s_name,
7>         n_name,
8>         p_partkey,
9>         p_mfgr,
10>         s_address,
11>         s_phone,
12>         s_comment
13> from
14>         part,
15>         supplier,
16>         partsupp,
17>         nation,
18>         region
19> where
20>         p_partkey = ps_partkey
21>         and s_suppkey = ps_suppkey
22>         and p_size = 15
23>         and p_type like '%BRASS'
24>         and s_nationkey = n_nationkey
25>         and n_regionkey = r_regionkey
26>         and r_name = 'EUROPE'
27>         and ps_supplycost = (
28>                 select
29>                         min(ps_supplycost)
30>                 from
31>                         partsupp,
32>                         supplier,
33>                         nation,
34>                         region
35>                 where
36>                         p_partkey = ps_partkey
37>                         and s_suppkey = ps_suppkey
38>                         and s_nationkey = n_nationkey
39>                         and n_regionkey = r_regionkey
40>                         and r_name = 'EUROPE'
41>         )
42> order by
43>         s_acctbal desc,
44>         n_name,
45>         s_name,
46>         p_partkey
```

Query Result

```
s_acctbal               s_name                  n_name
p_partkey  p_mfgr                s_address                      s_phone
s_comment
------------------------- ------------------------ ------------------------ ------
---------- ------------------------ ---------------------------------------- -------------
- ----------------------------------------------------------------------------
----------------
9938.530000 Supplier#000005359       UNITED KINGDOM
185358 Manufacturer#4          QKuHYh,vZGiwu2FWEJoLDx04
33-429-790-6131 uriously regular requests hag
9937.840000 Supplier#000005969       ROMANIA
108438 Manufacturer#1
ANDENSOSmk,miq23Xfb5RWt6dvUcvt6Qa       29-520-692-3537
efully express instructions. regular requests against the slyly fin
9936.220000 Supplier#000005250       UNITED KINGDOM
249 Manufacturer#4          B3rqp0xbSEim4Mpy2RH J
33-320-228-2957 etect about the furiously final accounts. slyly ironic
pinto beans sleep inside the furiously
```

```
9923.770000 Supplier#000002324       GERMANY
29821 Manufacturer#4          y3OD9UywSTOk                17-
779-299-1839 ackages boost blithely. blithely regular deposits c
9871.220000 Supplier#000006373       GERMANY
43868 Manufacturer#5          J8fcXWsTqM                  17-
813-485-8637 etect blithely bold asymptotes. fluffily ironic platelets
wake furiously; blit
9870.780000 Supplier#000001286       GERMANY
81285 Manufacturer#2
YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH    17-516-924-4574
regular accounts. furiously unusual courts above the fi
9870.780000 Supplier#000001286       GERMANY
181285 Manufacturer#4
YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH    17-516-924-4574
regular accounts. furiously unusual courts above the fi
9852.520000 Supplier#000008973       RUSSIA                18972
Manufacturer#2          t5L67YdBYYH6o,Vz24jpDyQ9          32-
188-594-7038 rns wake final foxes. carefully unusual depende
9847.830000 Supplier#000008097       RUSSIA                130557
Manufacturer#2          xMe97bpE69NzdwLoX                 32-375-
640-3593  the special excuses. silent sentiments serve carefully final
ac
9847.570000 Supplier#000006345       FRANCE
86344 Manufacturer#1
VSt3rzk3qG698u6ld8HhOByvrTcSTSvQlDQDag   16-886-766-7945
ges. slyly regular requests are. ruthless, express excuses cajole blithely
across the unu

<truncated>

7937.930000 Supplier#000009012       ROMANIA
83995 Manufacturer#2          iUiTziH,Ek3i4lwSgunXMgrcTzwdb
29-250-925-9690 to the blithely ironic deposits nag sly
7914.450000 Supplier#000001013       RUSSIA                125988
Manufacturer#2          riRcntps4KEDtYScjpMIWeYF6mNnR
32-194-698-3365  busily bold packages are dolphi
7912.910000 Supplier#000004211       GERMANY
159180 Manufacturer#5
2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG        17-266-947-7315
ay furiously regular platelets. cou
7912.910000 Supplier#000004211       GERMANY
184210 Manufacturer#4
2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG        17-266-947-7315
ay furiously regular platelets. cou
7894.560000 Supplier#000007981       GERMANY
85472 Manufacturer#4          NSJ96vMROAbeXP
17-963-404-3760 ic platelets affix after the furiously
7887.080000 Supplier#000009792       GERMANY
164759 Manufacturer#3          Y28ITVeYriT3kIGdV2K8fSZ
V2UqT5H1Otz      17-988-938-4296 ckly around the carefully fluffy
theodolites. slyly ironic pack
7871.500000 Supplier#000007206       RUSSIA                104695
Manufacturer#1          3w fNCnrVmvJjE95sgWZzvW           32-
432-452-7731 ironic requests. furiously final theodolites cajole. final,
express packages sleep. quickly reg
7852.450000 Supplier#000005864       RUSSIA                8363
Manufacturer#4          WCNfBPZeSXh3h,c                   32-454-
883-3821 usly unusual pinto beans. brave ideas sleep carefully quickly
ironi
7850.660000 Supplier#000001518       UNITED KINGDOM
86501 Manufacturer#1          ONda3YJiHKJOC               33-
730-383-3892 ifts haggle fluffily pending pai
7843.520000 Supplier#000006683       FRANCE
11680 Manufacturer#4          2Z0JGkiv01Y00oCFwUGfviIbhzCdy
16-464-517-8943  express, final pinto beans x-ray slyly asymptotes.
unusual, unusual

(100 rows affected)
::::::::::::::
qryqual03
::::::::::::::
Query Text
----------

3>
4> select top 10
5>         l_orderkey,
6>         sum(l_extendedprice * (1 - l_discount)) as revenue,
7>         dateformat (o_orderdate,'yyyy-mm-dd'),
8>         o_shippriority
9> from
10>        customer,
11>        orders,
12>        lineitem
13> where
14>        c_mktsegment = 'BUILDING'
15>        and c_custkey = o_custkey
16>        and l_orderkey = o_orderkey
17>        and o_orderdate < '1995-03-15'
18>        and l_shipdate > '1995-03-15'
19> group by
20>        l_orderkey,
21>        o_orderdate,
22>        o_shippriority
23> order by
24>        revenue desc,
25>        o_orderdate
26>
Query Result
------------

l_orderkey          revenue
dateformat(orders.o_orderdate,'yyyy-mm-dd') o_shippriority
-------------------- --------------------------- ------------------------------------
------ --------------
2456423              406181.011100 1995-03-05
0
3459808              405838.698900 1995-03-04
0
492164               390324.061000 1995-02-19
0
1188320              384537.935900 1995-03-09
0
2435712              378673.055800 1995-02-26
0
4878020              378376.795200 1995-03-12
0
5521732              375153.921500 1995-03-13
0
2628192              373133.309400 1995-02-22
0
993600               371407.459500 1995-03-05
0
2300070              367371.145200 1995-03-13
0

(10 rows affected)
::::::::::::::
```

```
qryqual04
::::::::::::::
Query Text
----------

2>
3> select
4>          o_orderpriority,
5>          count(*) as order_count
6> from
7>          orders
8> where
9>          o_orderdate >= '1993-07-01'
10>         and o_orderdate < dateadd(month, 3, '1993-07-01')
11>         and exists (
12>                 select *
13>                 from
14>                         lineitem
15>                 where
16>                         l_orderkey = o_orderkey
17>                         and l_commitdate < l_receiptdate
18>         )
19> group by
20>         o_orderpriority
21> order by
22>         o_orderpriority
Query Result
------------

o_orderpriority order_count
--------------- --------------------
1-URGENT                10594
2-HIGH                  10476
3-MEDIUM                10410
4-NOT SPECIFIED         10556
5-LOW                   10487

(5 rows affected)
::::::::::::::
qryqual05
::::::::::::::
Query Text
----------

2>
3> select
4>          n_name,
5>          sum(l_extendedprice * (1 - l_discount)) as revenue
6> from
7>          customer,
8>          orders,
9>          lineitem,
10>         supplier,
11>         nation,
12>         region
13> where
14>         c_custkey = o_custkey
15>         and l_orderkey = o_orderkey
16>         and l_suppkey = s_suppkey
17>         and c_nationkey = s_nationkey
18>         and s_nationkey = n_nationkey
19>         and n_regionkey = r_regionkey
20>         and r_name = 'ASIA'
21>         and o_orderdate >= '1994-01-01'
22>         and o_orderdate < dateadd(year, 1, '1994-01-01')
23> group by
24>         n_name
25> order by
26>         revenue desc
27>
Query Result
------------

n_name                  revenue
----------------------- --------------------------
INDONESIA                     55502041.169700
VIETNAM                       55295086.996700
CHINA                         53724494.256600
INDIA                         52035512.000200
JAPAN                         45410175.695400

(5 rows affected)
::::::::::::::
qryqual06
::::::::::::::
Query Text
----------

2>
3> select
4>          sum(l_extendedprice * l_discount) as revenue
5> from
6>          lineitem
7> where
8>          l_shipdate >= '1994-01-01'
9>          and l_shipdate < dateadd(year, 1, '1994-01-01')
10>         and l_discount between .06 - 0.01 and .06 + 0.01
11>         and l_quantity < 24
12>
Query Result
------------

revenue
---------------------------
123141078.228302

(1 row affected)
::::::::::::::
qryqual07
::::::::::::::
Query Text
----------

3>
4> select
5>          supp_nation,
6>          cust_nation,
7>          l_year,
8>          sum(volume) as revenue
9> from
10>         (
11>                 select
12>                         n1.n_name as supp_nation,
13>                         n2.n_name as cust_nation,
14>                         year(l_shipdate) as l_year,
15>                         l_extendedprice * (1 - l_discount) as
volume
16>                 from
```

```
17>                     supplier,
18>                     lineitem,
19>                     orders,
20>                     customer,
21>                     nation n1,
22>                     nation n2
23>             where
24>                     s_suppkey = l_suppkey
25>                     and o_orderkey = l_orderkey
26>                     and c_custkey = o_custkey
27>                     and s_nationkey = n1.n_nationkey
28>                     and c_nationkey = n2.n_nationkey
29>                     and (
30>                             (n1.n_name = 'FRANCE' and
n2.n_name = 'GERMANY')
31>                             or (n1.n_name = 'GERMANY'
and n2.n_name = 'FRANCE')
32>                     )
33>                     and l_shipdate between '1995-01-01' and
'1996-12-31'
34>     ) as shipping
35> group by
36>     supp_nation,
37>     cust_nation,
38>     l_year
39> order by
40>     supp_nation,
41>     cust_nation,
42>     l_year
43>
Query Result
------------

supp_nation              cust_nation              l_year      revenue
------------------------ ------------------------ ----------- ----------------------
-----
FRANCE                   GERMANY                  1995
54639732.733600
FRANCE                   GERMANY                  1996
54633083.307600
GERMANY                  FRANCE                   1995
52531746.669700
GERMANY                  FRANCE                   1996
52520549.022400

(4 rows affected)
::::::::::::::
qryqual08
::::::::::::::
Query Text
----------


4>
5> select
6>     o_year,
7>     sum(case
8>             when nation = 'BRAZIL' then volume
9>             else 0
10>    end) / sum(volume) as mkt_share
11> from
12>    (
13>             select
14>                     year(o_orderdate) as o_year,
15>                     l_extendedprice * (1 - l_discount) as
volume,
16>                     n2.n_name as nation
17>             from
18>                     part,
19>                     supplier,
20>                     lineitem,
21>                     orders,
22>                     customer,
23>                     nation n1,
24>                     nation n2,
25>                     region
26>             where
27>                     p_partkey = l_partkey
28>                     and s_suppkey = l_suppkey
29>                     and l_orderkey = o_orderkey
30>                     and o_custkey = c_custkey
31>                     and c_nationkey = n1.n_nationkey
32>                     and n1.n_regionkey = r_regionkey
33>                     and r_name = 'AMERICA'
34>                     and s_nationkey = n2.n_nationkey
35>                     and o_orderdate between '1995-01-01'
and '1996-12-31'
36>                     and p_type = 'ECONOMY ANODIZED
STEEL'
37>     ) as all_nations
38> group by
39>     o_year
40> order by
41>     o_year
42>
Query Result
------------

o_year      mkt_share
----------- ---------------------------
1995                 0.034436
1996                 0.041486

(2 rows affected)
::::::::::::::
qryqual09
::::::::::::::
Query Text
----------


3>
4> select
5>     nation,
6>     o_year,
7>     sum(amount) as sum_profit
8> from
9>     (
10>             select
11>                     n_name as nation,
12>                     year(o_orderdate) as o_year,
13>                     l_extendedprice * (1 - l_discount) -
ps_supplycost * l_quantity as amount
14>             from
15>                     part,
16>                     supplier,
17>                     lineitem,
18>                     partsupp,
19>                     orders,
```

```
20>                     nation
21>             where
22>                     s_suppkey = l_suppkey
23>                     and ps_suppkey = l_suppkey
24>                     and ps_partkey = l_partkey
25>                     and p_partkey = l_partkey
26>                     and o_orderkey = l_orderkey
27>                     and s_nationkey = n_nationkey
28>                     and p_name like '%green%'
29>     ) as profit
30> group by
31>     nation,
32>     o_year
33> order by
34>     nation,
35>     o_year desc
36>
Query Result
------------

nation                    o_year      sum_profit
------------------------- ----------- --------------------------
ALGERIA                   1998        31342867.234500
ALGERIA                   1997        57138193.023300
ALGERIA                   1996        56140140.133000
ALGERIA                   1995        53051469.653400
ALGERIA                   1994        53867582.128600
ALGERIA                   1993        54942718.132400
ALGERIA                   1992        54628034.712700
ARGENTINA                 1998        30211185.708100
ARGENTINA                 1997        50805741.752300
ARGENTINA                 1996        51923746.575500
ARGENTINA                 1995        49298625.766600
ARGENTINA                 1994        50835610.109500
ARGENTINA                 1993        51646079.177500
ARGENTINA                 1992        50410314.994800
BRAZIL                    1998        27217924.383200
BRAZIL                    1997        48378669.198900
BRAZIL                    1996        50482870.357200
BRAZIL                    1995        47623383.634900
BRAZIL                    1994        47840165.725600
BRAZIL                    1993        49054694.035100

<truncated>

UNITED KINGDOM            1997        49381810.898600
UNITED KINGDOM            1996        51386853.960400
UNITED KINGDOM            1995        51509586.788500
UNITED KINGDOM            1994        48086499.711500
UNITED KINGDOM            1993        49166827.223500
UNITED KINGDOM            1992        49349122.082500
UNITED STATES             1998        25126238.946100
UNITED STATES             1997        50077306.418600
UNITED STATES             1996        48048649.470300
UNITED STATES             1995        48809032.422600
UNITED STATES             1994        49296747.182700
UNITED STATES             1993        48029946.801400
UNITED STATES             1992        48671944.498300
VIETNAM                   1998        30442736.059400
VIETNAM                   1997        50309179.794200
VIETNAM                   1996        50488161.410000
VIETNAM                   1995        49658284.612500
VIETNAM                   1994        50596057.260700
VIETNAM                   1993        50953919.151900
VIETNAM                   1992        49613838.315100

(175 rows affected)
::::::::::::::
qryqual10
::::::::::::::
Query Text
----------

3>
4> select top 20
5>     c_custkey,
6>     c_name,
7>     sum(l_extendedprice * (1 - l_discount)) as revenue,
8>     c_acctbal,
9>     n_name,
10>    c_address,
11>    c_phone,
12>    c_comment
13> from
14>    customer,
15>    orders,
16>    lineitem,
17>    nation
18> where
19>    c_custkey = o_custkey
20>    and l_orderkey = o_orderkey
21>    and o_orderdate >= '1993-10-01'
22>    and o_orderdate < dateadd(month, 3, '1993-10-01')
23>    and l_returnflag = 'R'
24>    and c_nationkey = n_nationkey
25> group by
26>    c_custkey,
27>    c_name,
28>    c_acctbal,
29>    c_phone,
30>    n_name,
31>    c_address,
32>    c_comment
33> order by
34>    revenue desc
35>
Query Result
------------

c_custkey  c_name                  revenue                 c_acctbal
n_name                    c_address                                 c_phone
c_comment
---------- ------------------------ -------------------------- ----------------------
------ ------------------------- ---------------------------------------- -----------
--- ----------------------------------------------------------------------------
-----------------------------------
57040 Customer#000057040                  734235.245500
632.870000 JAPAN                  Eioyzjf4pp                              22-
895-641-3466 sits. slyly regular requests sleep alongside of the regular
inst
143347 Customer#000143347                 721002.694800
2557.470000 EGYPT                 1aReFYv,Kw4
14-742-935-3718 ggle carefully enticing requests. final deposits use
bold, bold pinto beans. ironic, idle re
60838 Customer#000060838                  679127.307700
2454.770000 BRAZIL
64EaJ5vMAHWJlBOxJklpNc2RJiWE              12-913-494-9813  need
to boost against the slyly regular account
```

```
101998 Customer#000101998                637029.566700
3790.890000 UNITED KINGDOM          01c9CILnNtfOQYmZj
33-593-865-6378 ress foxes wake slyly after the bold excuses. ironic
platelets are furiously carefully bold theodolites
125341 Customer#000125341                633508.086000
4983.510000 GERMANY
S29ODD6bceU8QSuuEJznkNaK            17-582-695-5962 arefully
even depths. blithely even excuses sleep furiously. foxes use except
the dependencies. ca
25501 Customer#000025501                 620269.784900
7725.040000 ETHIOPIA
W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ     15-874-808-6793
he pending instructions wake carefully at the pinto beans. regular, final
instructions along the slyly fina
115831 Customer#000115831                596423.867200
5098.100000 FRANCE                rFeBbEEyk dl
ne7zV5fDrmiq1oK09wV7pxqCgIc 16-715-386-3788 l somas sleep.
furiously final deposits wake blithely regular pinto b
84223 Customer#000084223                 594998.023900
528.650000 UNITED KINGDOM           nAVZCs6BaWap rrM27N
2qBnzc5WBauxbA       33-442-824-8191  slyly final deposits haggle
regular, pending dependencies. pending escapades wake
54289 Customer#000054289                 585603.391800
5583.020000 IRAN                  vXCxoCsU0Bad5JQI ,oobkZ
20-834-292-4707 ely special foxes are quickly finally ironic p
39922 Customer#000039922                 584878.113400
7321.110000 GERMANY
Zgy4s50l2GKN4pLDPBU8m342gIw6R          17-147-757-8036 y
final requests. furiously final foxes cajole blithely special platelets. f
6226 Customer#000006226                  576783.760600
2230.090000 UNITED KINGDOM
8gPu8,NPGkfyQQ0hcIYUGPIBWc,ybP5g,       33-657-701-3391
ending platelets along the express deposits cajole carefully final
922 Customer#000000922                   576767.533300
3869.250000 GERMANY
Az9RFaut7NkPnc5zSD2PwHgVwr4jRzq         17-945-916-9648
luffily fluffy deposits. packages c
147946 Customer#000147946                576455.132000
2030.130000 ALGERIA               iANyZHjqhyy7Ajah0pTrYyhJ
10-886-956-3143 ithely ironic deposits haggle blithely ironic requests.
quickly regu
115640 Customer#000115640                569341.193300
6436.100000 ARGENTINA             Vtgfia9qI 7EpHgecU1X
11-411-543-4901 ost slyly along the patterns; pinto be
73606 Customer#000073606                 568656.857800
1785.670000 JAPAN                 xuR0Tro5yChDfOCrjkd2ol
22-437-653-6966 he furiously regular ideas. slowly
110246 Customer#000110246                566842.981500
7763.350000 VIETNAM               7KzflgX MDOq7sOkI
31-943-426-9837 egular deposits serve blithely above the fl
142549 Customer#000142549                563537.236800
5085.990000 INDONESIA
ChqEoK43OysjdHbtKCp6dKqjNyvvi9         19-955-562-2398 sleep
pending courts. ironic deposits against the carefully unusual platelets
cajole carefully express accounts.
146149 Customer#000146149                557254.986500
1791.550000 ROMANIA               s87fvzFQpU
29-744-164-6487  of the slyly silent accounts. quickly final accounts
across the
52528 Customer#000052528                 556397.350900
551.790000 ARGENTINA              NFztyTOR10UOJ
11-208-192-3205  deposits hinder. blithely pending asymptotes breach
slyly regular re
23431 Customer#000023431                 554269.536000
3381.860000 ROMANIA               HgiV0phqhaIa9aydNoIlb
29-915-458-2654 nusual, even instructions: furiously stealthy n

(20 rows affected)
::::::::::::::
qryqual11
::::::::::::::
Query Text
----------


3>
4>
5> select
6>      ps_partkey,
7>      sum(ps_supplycost * ps_availqty) as value
8> from
9>      partsupp,
10>     supplier,
11>     nation
12> where
13>     ps_suppkey = s_suppkey
14>     and s_nationkey = n_nationkey
15>     and n_name = 'GERMANY'
16> group by
17>     ps_partkey having
18>             sum(ps_supplycost * ps_availqty) > (
19>                     select
20>                             sum(ps_supplycost *
ps_availqty) * 0.0001000000
21>                     from
22>                             partsupp,
23>                             supplier,
24>                             nation
25>                     where
26>                             ps_suppkey = s_suppkey
27>                             and s_nationkey =
n_nationkey
28>                             and n_name = 'GERMANY'
29>             )
30> order by
31>     value desc
32>
Query Result
------------

ps_partkey value
---------- --------------------------
129760          17538456.860000
166726          16503353.920000
191287          16474801.970000
161758          16101755.540000
34452           15983844.720000
139035          15907078.340000
9403           15451755.620000
154358          15212937.880000
38823           15064802.860000
85606           15053957.150000
33354           14408297.400000
154747          14407580.680000
82865           14235489.780000
76094           14094247.040000
222            13937777.740000
121271          13908336.000000
```

```
55221           13716120.470000
22819           13666434.280000
76281           13646853.680000
85298           13581154.930000
85158           13554904.000000
139684           13535538.720000
31034           13498025.250000
87305           13482847.040000
10181           13445148.750000

<truncated>

79753           7907933.880000
108765           7905338.980000
146530           7905336.600000
71475           7903367.580000
36289           7901946.500000
61739           7900794.000000
52338           7898638.080000
194299           7898421.240000
105235           7897829.940000
77207           7897752.720000
96712           7897575.270000
10157           7897046.250000
171154           7896814.500000
79373           7896186.000000
113808           7893353.880000
27901           7892952.000000
128820           7892882.720000
25891           7890511.200000
122819           7888881.020000
154731           7888301.330000
101674           7879324.600000
51968           7879102.210000
72073           7877736.110000
5182           7874521.730000

(1048 rows affected)
::::::::::::::
qryqual12
::::::::::::::
Query Text
----------

3>
4> select
5>          l_shipmode,
6>          sum(case
7>                    when o_orderpriority = '1-URGENT'
8>                              or o_orderpriority = '2-HIGH'
9>                              then 1
10>                   else 0
11>         end) as high_line_count,
12>         sum(case
13>                   when o_orderpriority <> '1-URGENT'
14>                             and o_orderpriority <> '2-HIGH'
15>                             then 1
16>                   else 0
17>         end) as low_line_count
18> from
19>         orders,
20>         lineitem
21> where
22>         o_orderkey = l_orderkey
23>         and l_shipmode in ('MAIL', 'SHIP')
24>         and l_commitdate < l_receiptdate
25>         and l_shipdate < l_commitdate
26>         and l_receiptdate >= '1994-01-01'
27>         and l_receiptdate < dateadd(year, 1, '1994-01-01')
28> group by
29>         l_shipmode
30> order by
31>         l_shipmode
32>
Query Result
------------

l_shipmode high_line_count      low_line_count
---------- -------------------- --------------------
MAIL                 6202                 9324
SHIP                 6200                 9262

(2 rows affected)
::::::::::::::
qryqual13
::::::::::::::
Query Text
----------

3>
4> select
5>          c_count,
6>          count(*) as custdist
7> from
8>          (
9>                    select
10>                             c_custkey,
11>                             count(o_orderkey)
12>                   from
13>                             customer left outer join orders on
14>                                       c_custkey = o_custkey
15>                                       and o_comment not like
'%special%requests%'
16>                   group by
17>                             c_custkey
18>         ) as c_orders (c_custkey, c_count)
19> group by
20>         c_count
21> order by
22>         custdist desc,
23>         c_count desc
24>
Query Result
------------

c_count             custdist
-------------------- --------------------
0                   50005
9                    6641
10                   6532
11                   6014
8                    5937
12                   5639
13                   5024
19                   4793
7                    4687
17                   4587
18                   4529
```

```
20              4516
15              4505
14              4446
16              4273
21              4190
22              3623
6               3265
23              3225
24              2742
25              2086
5               1948
26              1612
27              1179
4               1007
28               893
29               593
3                415
30               376
31               226
32               148
2                134
33                75
34                50
35                37
1                 17
36                14
38                 5
37                 5
40                 4
41                 2
39                 1

(42 rows affected)
::::::::::::::
qryqual14
::::::::::::::
Query Text
----------

3>
4> select
5>          100.00 * sum(case
6>                      when p_type like 'PROMO%'
7>                              then l_extendedprice * (1 - l_discount)
8>                      else 0
9>          end) / sum(l_extendedprice * (1 - l_discount)) as
promo_revenue
10> from
11>         lineitem,
12>         part
13> where
14>         l_partkey = p_partkey
15>         and l_shipdate >= '1995-09-01'
16>         and l_shipdate < dateadd(month, 1, '1995-09-01')
17>
Query Result
------------

promo_revenue
--------------------------
16.380779

(1 row affected)
::::::::::::::
qryqual15
::::::::::::::
Query Text
----------

3>
4> create view revenue0 (supplier_no, total_revenue) as
5>          select
6>                      l_suppkey,
7>                      sum(l_extendedprice * (1 - l_discount))
8>          from
9>                      lineitem
10>         where
11>                     l_shipdate >= '1996-01-01'
12>                     and l_shipdate < dateadd(month,3,'1996-01-01')
13>         group by
14>                     l_suppkey
1>
2> select
3>          s_suppkey,
4>          s_name,
5>          s_address,
6>          s_phone,
7>          total_revenue
8> from
9>          supplier,
10>         revenue0
11> where
12>         s_suppkey = supplier_no
13>         and total_revenue = (
14>                     select
15>                             max(total_revenue)
16>                     from
17>                             revenue0
18>         )
19> order by
20>         s_suppkey
21>
s_suppkey  s_name                   s_address                               s_phone
total_revenue
---------- ------------------------- --------------------------------------- -------
------- ---------------------------
8449 Supplier#000008449        Wp34zim9qYFbVctdW
20-469-856-8873             1772627.208700

(1 row affected)
1>
2> drop view revenue0
Query Result
------------

::::::::::::::
qryqual16
::::::::::::::
Query Text
----------

3>
4>
5> select
6>          p_brand,
7>          p_type,
8>          p_size,
9>          count(distinct ps_suppkey) as supplier_cnt
```

```
10> from
11>         partsupp,
12>         part
13> where
14>         p_partkey = ps_partkey
15>         and p_brand <> 'Brand#45'
16>         and p_type not like 'MEDIUM POLISHED%'
17>         and p_size in (49, 14, 23, 45, 19, 3, 36, 9)
18>         and ps_suppkey not in (
19>                 select
20>                         s_suppkey
21>                 from
22>                         supplier
23>                 where
24>                         s_comment like
'%Customer%Complaints%'
25>         )
26> group by
27>         p_brand,
28>         p_type,
29>         p_size
30> order by
31>         supplier_cnt desc,
32>         p_brand,
33>         p_type,
34>         p_size
35>
Query Result
------------

p_brand    p_type                    p_size      supplier_cnt
---------- ------------------------- ----------- --------------------
Brand#41   MEDIUM BRUSHED TIN               3               28
Brand#54   STANDARD BRUSHED COPPER          14
27
Brand#11   STANDARD BRUSHED TIN            23               24
Brand#11   STANDARD BURNISHED BRASS         36
24
Brand#15   MEDIUM ANODIZED NICKEL           3               24
Brand#15   SMALL ANODIZED BRASS            45               24
Brand#15   SMALL BURNISHED NICKEL          19               24
Brand#21   MEDIUM ANODIZED COPPER           3               24
Brand#22   SMALL BRUSHED NICKEL            3               24
Brand#22   SMALL BURNISHED BRASS           19               24
Brand#25   MEDIUM BURNISHED COPPER          36
24
Brand#31   PROMO POLISHED COPPER           36               24
Brand#33   LARGE POLISHED TIN              23               24
Brand#33   PROMO POLISHED STEEL            14               24
Brand#35   PROMO BRUSHED NICKEL            14               24
Brand#41   ECONOMY BRUSHED STEEL           9               24
Brand#41   ECONOMY POLISHED TIN            19               24
Brand#41   LARGE PLATED COPPER             36               24
Brand#42   ECONOMY PLATED BRASS            3               24
Brand#42   STANDARD POLISHED TIN           49               24

<truncated>

Brand#55   STANDARD POLISHED STEEL         14                4
Brand#55   STANDARD POLISHED STEEL         23                4
Brand#55   STANDARD POLISHED TIN           9                4
Brand#55   STANDARD POLISHED TIN           19                4
Brand#55   STANDARD POLISHED TIN           36                4
Brand#11   SMALL BRUSHED TIN               19               3
Brand#15   LARGE PLATED NICKEL             45               3
Brand#15   LARGE POLISHED NICKEL            9               3
Brand#21   PROMO BURNISHED STEEL            45               3
Brand#22   STANDARD PLATED STEEL            23               3
Brand#25   LARGE PLATED STEEL               19               3
Brand#32   STANDARD ANODIZED COPPER          23
3
Brand#33   SMALL ANODIZED BRASS             9               3
Brand#35   MEDIUM ANODIZED TIN              19               3
Brand#51   SMALL PLATED BRASS               23               3
Brand#52   MEDIUM BRUSHED BRASS             45               3
Brand#53   MEDIUM BRUSHED TIN               45               3
Brand#54   ECONOMY POLISHED BRASS           9               3
Brand#55   PROMO PLATED BRASS               19               3
Brand#55   STANDARD PLATED TIN              49               3

(18314 rows affected)
::::::::::::::
qryqual17
::::::::::::::
Query Text
----------

3>
4> select
5>         sum(l_extendedprice) / 7.0 as avg_yearly
6> from
7>         lineitem,
8>         part
9> where
10>         p_partkey = l_partkey
11>         and p_brand = 'Brand#23'
12>         and p_container = 'MED BOX'
13>         and l_quantity < (
14>                 select
15>                         0.2 * avg(l_quantity)
16>                 from
17>                         lineitem
18>                 where
19>                         l_partkey = p_partkey
20>         )
21>
Query Result
------------

avg_yearly
--------------------------
348406.054286

(1 row affected)
::::::::::::::
qryqual18
::::::::::::::
Query Text
----------

3>
4>
5> select top 100
6>         c_name,
7>         c_custkey,
8>         o_orderkey,
9>         dateformat (o_orderdate,'yyyy-mm-dd'),
10>         o_totalprice,
```

```
11>         sum(l_quantity)
12> from
13>         customer,
14>         orders,
15>         lineitem
16> where
17>         o_orderkey in (
18>                 select
19>                         l_orderkey
20>                 from
21>                         lineitem
22>                 group by
23>                         l_orderkey having
24>                                 sum(l_quantity) > 300
25>         )
26>         and c_custkey = o_custkey
27>         and o_orderkey = l_orderkey
28> group by
29>         c_name,
30>         c_custkey,
31>         o_orderkey,
32>         o_orderdate,
33>         o_totalprice
34> order by
35>         o_totalprice desc,
36>         o_orderdate
37>
Query Result
------------

c_name                  c_custkey  o_orderkey
dateformat(orders.o_orderdate,'yyyy-mm-dd') o_totalprice
sum(lineitem.l_quantity)
------------------------ ---------- -------------------- ----------------------------
--------------- -------------------------- --------------------------
Customer#000128120         128120          4722021 1994-04-07
544089.090000              323.000000
Customer#000144617         144617          3043270 1997-02-12
530604.440000              317.000000
Customer#000013940          13940          2232932 1997-04-13
522720.610000              304.000000
Customer#000066790          66790          2199712 1996-09-30
515531.820000              327.000000
Customer#000046435          46435          4745607 1997-07-03
508047.990000              309.000000
Customer#000015272          15272          3883783 1993-07-28
500241.330000              302.000000
Customer#000146608         146608          3342468 1994-06-12
499794.580000              303.000000
Customer#000096103          96103          5984582 1992-03-16
494398.790000              312.000000
Customer#000024341          24341          1474818 1992-11-15
491348.260000              302.000000
Customer#000137446         137446          5489475 1997-05-23
487763.250000              311.000000
Customer#000107590         107590          4267751 1994-11-04
485141.380000              301.000000
Customer#000050008          50008          2366755 1996-12-09
483891.260000              302.000000
Customer#000015619          15619          3767271 1996-08-07
480083.960000              318.000000
Customer#000077260          77260          1436544 1992-09-12
479499.430000              307.000000
Customer#000109379         109379          5746311 1996-10-10
478064.110000              302.000000
Customer#000054602          54602          5832321 1997-02-09
471220.080000              307.000000
Customer#000105995         105995          2096705 1994-07-03
469692.580000              307.000000
Customer#000148885         148885          2942469 1992-05-31
469630.440000              313.000000
Customer#000114586         114586           551136 1993-05-19
469605.590000              308.000000
Customer#000105260         105260          5296167 1996-09-06
469360.570000              303.000000

<truncated>

Customer#000066533          66533            29158 1995-10-21
443576.500000              305.000000
Customer#000037729          37729          4134341 1995-06-29
441082.970000              309.000000
Customer#000003566           3566          2329187 1998-01-04
439803.360000              304.000000
Customer#000045538          45538          4527553 1994-05-22
436275.310000              305.000000
Customer#000081581          81581          4739650 1995-11-04
435405.900000              305.000000
Customer#000119989         119989          1544643 1997-09-20
434568.250000              320.000000
Customer#000003680           3680          3861123 1998-07-03
433525.970000              301.000000
Customer#000113131         113131           967334 1995-12-15
432957.750000              301.000000
Customer#000141098         141098           565574 1995-09-24
430986.690000              301.000000
Customer#000093392          93392          5200102 1997-01-22
425487.510000              304.000000
Customer#000015631          15631          1845057 1994-05-12
419879.590000              302.000000
Customer#000112987         112987          4439686 1996-09-17
418161.490000              305.000000
Customer#000012599          12599          4259524 1998-02-12
415200.610000              304.000000
Customer#000105410         105410          4478371 1996-03-05
412754.510000              302.000000
Customer#000149842         149842          5156581 1994-05-30
411329.350000              302.000000
Customer#000010129          10129          5849444 1994-03-21
409129.850000              309.000000
Customer#000069904          69904          1742403 1996-10-19
408513.000000              305.000000
Customer#000017746          17746             6882 1997-04-09
408446.930000              303.000000
Customer#000013072          13072          1481925 1998-03-15
399195.470000              301.000000
Customer#000082441          82441           857959 1994-02-07
382579.740000              305.000000
Customer#000088703          88703          2995076 1994-01-30
363812.120000              302.000000

(57 rows affected)
::::::::::::::
qryqual19
::::::::::::::
Query Text
----------
```

```
3>
4>
5> select
6>          sum(l_extendedprice* (1 - l_discount)) as revenue
7> from
8>          lineitem,
9>          part
10> where
11>         (
12>                  p_partkey = l_partkey
13>                  and p_brand = 'Brand#12'
14>                  and p_container in ('SM CASE', 'SM BOX', 'SM
PACK', 'SM PKG')
15>                  and l_quantity >= 1 and l_quantity <= 1 + 10
16>                  and p_size between 1 and 5
17>                  and l_shipmode in ('AIR', 'AIR REG')
18>                  and l_shipinstruct = 'DELIVER IN PERSON'
19>         )
20>         or
21>         (
22>                  p_partkey = l_partkey
23>                  and p_brand = 'Brand#23'
24>                  and p_container in ('MED BAG', 'MED BOX',
'MED PKG', 'MED PACK')
25>                  and l_quantity >= 10 and l_quantity <= 10 + 10
26>                  and p_size between 1 and 10
27>                  and l_shipmode in ('AIR', 'AIR REG')
28>                  and l_shipinstruct = 'DELIVER IN PERSON'
29>         )
30>         or
31>         (
32>                  p_partkey = l_partkey
33>                  and p_brand = 'Brand#34'
34>                  and p_container in ('LG CASE', 'LG BOX', 'LG
PACK', 'LG PKG')
35>                  and l_quantity >= 20 and l_quantity <= 20 + 10
36>                  and p_size between 1 and 15
37>                  and l_shipmode in ('AIR', 'AIR REG')
38>                  and l_shipinstruct = 'DELIVER IN PERSON'
39>         )
40>
Query Result
------------

revenue
--------------------------
3083843.057800

(1 row affected)
::::::::::::::
qryqual20
::::::::::::::
Query Text
----------

3>
4> select
5>          s_name,
6>          s_address
7> from
8>          supplier,
9>          nation
10> where
11>         s_suppkey in (
12>                  select
13>                           ps_suppkey
14>                  from
15>                           partsupp
16>                  where
17>                           ps_partkey in (
18>                                    select
19>                                             p_partkey
20>                                    from
21>                                             part
22>                                    where
23>                                             p_name like
'forest%'
24>                           )
25>                           and ps_availqty > (
26>                                    select
27>                                             0.5 *
sum(l_quantity)
28>                                    from
29>                                             lineitem
30>                                    where
31>                                             l_partkey =
ps_partkey
32>                                             and l_suppkey =
ps_suppkey
33>                                             and l_shipdate >=
'1994-01-01'
34>                                             and l_shipdate <
dateadd(year,1,'1994-01-01')
35>                           )
36>         )
37>         and s_nationkey = n_nationkey
38>         and n_name = 'CANADA'
39> order by
40>         s_name
41>
Query Result
------------

s_name                    s_address
------------------------- ----------------------------------------
Supplier#000000020        iybAE,RmTymrZVYaFZva2SH,j
Supplier#000000091
YV45D7TkfdQanOOZ7q9QxkyGUapU1oOWU6q3
Supplier#000000197        YC2Acon6kjY3zj3Fbxs2k4Vdf7X0cd2F
Supplier#000000226        83qOdU2EYRdPQAQhEtn GRZEd
Supplier#000000285        Br7e1nnt1yxrw6ImgpJ7YdhFDjuBf
Supplier#000000378        FfbhyCxWvcPrO8ltp9
Supplier#000000402        i9Sw4DoyMhzhKXCH9By,AYSgmD
Supplier#000000530        0qwCMwobKY OcmLyfRXlagA8ukENJv,
Supplier#000000688        D
fw5ocppmZpYBBIPI718hCihLDZ5KhKX
Supplier#000000710        f19YPvOyb QoYwjKC,oPycpGfieBAcwKJo
Supplier#000000736
l6i2nMwVuovfKnuVgaSGK2rDy65DlAFLegiL7
Supplier#000000761
zlSLelQUj2XrvTTFnv7WAcYZGvvMTx882d4
Supplier#000000884        bmhEShejaS
Supplier#000000887        urEaTejH5POADP2ARrf
Supplier#000000935        ij98czM 2KzWe7dDTOxB8sq0UfCdvrX
Supplier#000000975        ,AC e,tBpNwKb5xMUzeohxlRn,
hdZJo73gFQF8y
Supplier#000001263        rQWr6nf8ZhB2TAiIDIvo5Io
```

```
Supplier#000001399      LmrocnIMSyYOWuANx7
Supplier#000001446      lch9HMNU1R7a0LIybsUodVknk6
Supplier#000001454      TOpimgu2TVXIjhiL93h,
Supplier#000001500      wDmF5xLxtQch9ctVu,

<truncated>

Supplier#000008972      w2vF6 D5YZO3visPXsqVfLADTK
Supplier#000009032      qK,trB6Sdy4Dz1BRUFNy
Supplier#000009147      rOAuryHxpZ9eOvx
Supplier#000009252      F7cZaPUHwh1 ZKyj3xmAVWC1XdP
ue1p5m,i
Supplier#000009278      RqYTzgxj93CLX 0mcYfCENOefD
Supplier#000009327      uoqMdf7e7Gj9dbQ53
Supplier#000009430      igRqmneFt
Supplier#000009567      r4Wfx4c3xsEAjcGj71HHZByornl
D9vrztXlv4
Supplier#000009601      51m637bO,Rw5DnHWFUvLacRx9
Supplier#000009709      rRnCbHYgDgl9PZYnyWKVYSUW0vKg
Supplier#000009753      wLhVEcRmd7PkJF4FBnGK7Z
Supplier#000009796      z,y4Idmr15DOvPUqYG
Supplier#000009799       4wNjXGa4OKWl
Supplier#000009811      E3iuyq7UnZxU7oPZIe2Gu6
Supplier#000009812
APFRMy3lCbgFga53n5t9DxzFPQPgnjrGt32
Supplier#000009862      rJzweWeN58
Supplier#000009868      ROjGgx5gvtkmnUUoeyy7v
Supplier#000009869
ucLqxzrpBTRMewGSM29t0rNTM30g1Tu3Xgg3mKag
Supplier#000009899      7XdpAHrzr1t,UQFZE
Supplier#000009974      7wJ,J5DKcxSU4Kp1cQLpbcAvB5AsvKT

(204 rows affected)
::::::::::::::
qryqual21
::::::::::::::
Query Text
----------

4>
5> select
6>      top 100 s_name,
7>      count(*) as numwait
8> from
9>      supplier,
10>       lineitem l1,
11>       orders,
12>       nation
13> where
14>       s_suppkey = l1.l_suppkey
15>       and o_orderkey = l1.l_orderkey
16>       and o_orderstatus = 'F'
17>       and l1.l_receiptdate > l1.l_commitdate
18>       and exists (
19>            select
20>                 *
21>            from
22>                 lineitem l2
23>            where
24>                 l2.l_orderkey = l1.l_orderkey
25>                 and l2.l_suppkey <> l1.l_suppkey
26>       )
27>       and not exists (
28>            select
29>                 *
30>            from
31>                 lineitem l3
32>            where
33>                 l3.l_orderkey = l1.l_orderkey
34>                 and l3.l_suppkey <> l1.l_suppkey
35>                 and l3.l_receiptdate > l3.l_commitdate
36>       )
37>       and s_nationkey = n_nationkey
38>       and n_name = 'SAUDI ARABIA'
39> group by
40>       s_name
41> order by
42>       numwait desc,
43>       s_name
44>
Query Result
------------

s_name                    numwait
------------------------- --------------------
Supplier#000002829                  20
Supplier#000005808                  18
Supplier#000000262                  17
Supplier#000000496                  17
Supplier#000002160                  17
Supplier#000002301                  17
Supplier#000002540                  17
Supplier#000003063                  17
Supplier#000005178                  17
Supplier#000008331                  17
Supplier#000002005                  16
Supplier#000002095                  16
Supplier#000005799                  16
Supplier#000005842                  16
Supplier#000006450                  16
Supplier#000006939                  16
Supplier#000009200                  16
Supplier#000009727                  16
Supplier#000000486                  15

<truncated>

Supplier#000007417                  13
Supplier#000007497                  13
Supplier#000007602                  13
Supplier#000008134                  13
Supplier#000008234                  13
Supplier#000009435                  13
Supplier#000009436                  13
Supplier#000009564                  13
Supplier#000009896                  13
Supplier#000000379                  12
Supplier#000000673                  12
Supplier#000000762                  12
Supplier#000000811                  12
Supplier#000000821                  12
Supplier#000001337                  12
Supplier#000001916                  12
Supplier#000001925                  12
Supplier#000002039                  12
Supplier#000002357                  12
Supplier#000002483                  12
```

(100 rows affected)
::::::::::::::
qryqual22
::::::::::::::
Query Text

```
3>
4> select
5>      cntrycode,
6>      count(*) as numcust,
7>      sum(c_acctbal) as totacctbal
8> from
9>      (
10>          select
11>              substr(c_phone,1,2) as cntrycode,
12>              c_acctbal
13>          from
14>              customer
15>          where
16>              substr(c_phone,1,2) in
17>                  ('13', '31', '23', '29', '30', '18', '17')
18>              and c_acctbal > (
19>                  select
20>                      avg(c_acctbal)
21>                  from
22>                      customer
23>                  where
24>                      c_acctbal > 0.00
25>                      and substr(c_phone,1,2) in
26>                          ('13', '31', '23', '29', '30', '18', '17')
27>              )
28>              and not exists (
29>                  select
30>                      *
31>                  from
32>                      orders
33>                  where
34>                      o_custkey = c_custkey
35>              )
36>      ) as custsale
37> group by
38>      cntrycode
39> order by
40>      cntrycode
41>
```

Query Result
----

cntrycode	numcust	totacctbal
13	888	6737713.990000
17	861	6460573.720000
18	964	7236687.400000
23	892	6701457.950000
29	948	7158866.630000
30	909	6808436.130000
31	922	6806670.180000

(7 rows affected)

Appendix E Seed and Input Parameters

E.1 stream_seeds

stream0 122133321
stream1 122133322
stream2 122133323
stream3 122133324
stream4 122133325
stream5 122133326
stream6 122133327
stream7 122133328

E.2 opts.0

14 1993-03-01
2 19 STEEL AFRICA
9 lime
20 maroon 1993-01-01 INDONESIA
6 1994-01-01 0.05 25
17 Brand#14 MED BAG
18 315
8 UNITED STATES AMERICA MEDIUM BRUSHED COPPER
21 CANADA
13 unusual packages
3 HOUSEHOLD 1995-03-01
22 10 12 15 25 13 32 17
16 Brand#11 ECONOMY PLATED 33 28 24 20 16 17 44 21
4 1993-03-01
11 JAPAN 0.0000001000
15 1996-08-01
1 115
10 1993-10-01
19 Brand#33 Brand#12 Brand#13 6 15 25
5 ASIA 1994-01-01
7 INDONESIA UNITED STATES
12 RAIL MAIL 1993-01-01

E.3 opts.1

21 SAUDI ARABIA
3 BUILDING 1995-03-18
18 312
5 MIDDLE EAST 1994-01-01
11 ALGERIA 0.0000001000
7 ARGENTINA MOZAMBIQUE
6 1994-01-01 0.02 25
20 turquoise 1997-01-01 UNITED STATES
17 Brand#11 MED PACK
12 AIR MAIL 1993-01-01
16 Brand#51 STANDARD BRUSHED 5 48 41 7 26 12 20 31
15 1994-05-01
13 unusual packages
10 1994-07-01
2 7 BRASS ASIA
8 MOZAMBIQUE AFRICA MEDIUM PLATED COPPER
14 1993-06-01
19 Brand#35 Brand#55 Brand#12 1 16 22
9 khaki
22 11 13 18 12 15 25 20
1 62
4 1995-10-01

E.4 opts.2

6 1994-01-01 0.07 24
17 Brand#13 MED DRUM
14 1993-09-01
16 Brand#31 LARGE ANODIZED 22 35 11 5 48 20 1 21
19 Brand#32 Brand#33 Brand#51 6 17 29
10 1993-04-01
9 green
2 44 TIN AFRICA
15 1996-11-01
8 INDIA ASIA MEDIUM ANODIZED COPPER
5 AFRICA 1994-01-01
22 23 34 26 16 27 25 20
12 REG AIR FOB 1993-01-01
7 CHINA INDIA
13 unusual packages
18 314
1 70
4 1993-07-01
20 green 1995-01-01 KENYA
3 HOUSEHOLD 1995-03-03
11 JORDAN 0.0000001000
21 JAPAN

E.5 opts.3

8 ALGERIA AFRICA SMALL BRUSHED TIN
5 AMERICA 1994-01-01
4 1996-02-01
6 1994-01-01 0.05 24
17 Brand#15 JUMBO BAG
7 IRAN ALGERIA
1 78
18 315
22 13 17 26 19 32 16 28
14 1993-12-01
9 floral
10 1994-01-01
15 1994-08-01
11 ARGENTINA 0.0000001000
20 royal 1993-01-01 CANADA
2 32 COPPER EUROPE
21 EGYPT
19 Brand#44 Brand#11 Brand#55 2 18 25
13 express packages
16 Brand#11 PROMO PLATED 33 40 49 8 39 36 13 2
12 FOB AIR 1994-01-01
3 AUTOMOBILE 1995-03-20

E.6 opts.4

5 ASIA 1995-01-01
21 VIETNAM
14 1994-03-01
19 Brand#42 Brand#54 Brand#55 7 19 21
15 1997-03-01
17 Brand#12 JUMBO PACK
12 MAIL FOB 1994-01-01
6 1995-01-01 0.02 25
4 1993-11-01

9 dark
8 PERU AMERICA SMALL PLATED TIN
16 Brand#51 SMALL POLISHED 47 4 17
37 20 43 49 9
11 KENYA 0.0000001000
2 20 STEEL AFRICA
10 1994-10-01
18 313
1 86
13 express packages
7 BRAZIL PERU
22 21 30 16 15 32 26
13
3 HOUSEHOLD 1995-03-05
20 cornsilk 1997-01-01 CHINA

E.7 opts.5

21 KENYA
15 1994-11-01
4 1996-05-01
6 1995-01-01 0.08 24
7 ROMANIA INDONESIA
16 Brand#31 ECONOMY ANODIZED 40 14
36 10 20 8 9 6
19 Brand#44 Brand#32 Brand#44 2 20 28
18 314
14 1994-07-01
22 20 16 19 14 34 30
28
11 BRAZIL 0.0000001000
13 express requests
3 AUTOMOBILE 1995-03-22
1 94
2 8 BRASS EUROPE
5 EUROPE 1995-01-01
8 INDONESIA ASIA SMALL ANODIZED TIN
20 navy 1995-01-01 INDIA
12 TRUCK FOB 1994-01-01
17 Brand#14 JUMBO DRUM
10 1993-08-01
9 chocolate

E.8 opts.6

10 1994-05-01
3 FURNITURE 1995-03-07
15 1997-06-01
13 express requests
6 1995-01-01 0.05 24
8 ARGENTINA AMERICA STANDARD
POLISHED TIN
9 blush
7 IRAQ ARGENTINA
4 1994-02-01
11 MOROCCO 0.0000001000
22 30 18 24 10 17 25
22
18 312
12 RAIL SHIP 1994-01-01
1 102
5 MIDDLE EAST 1995-01-01
16 Brand#11 STANDARD BURNISHED 43 7
47 20 18 27 41 46
2 45 NICKEL AMERICA
14 1994-10-01
19 Brand#51 Brand#25 Brand#43 7 10 24
20 azure 1994-01-01 RUSSIA
17 Brand#11 WRAP BAG
21 FRANCE

E.9 opts.7

18 313
8 CHINA ASIA STANDARD BURNISHED TIN
20 lavender 1997-01-01 JAPAN
21 UNITED KINGDOM
2 33 COPPER EUROPE
4 1996-09-01
22 21 26 13 15 12 25
28
17 Brand#13 WRAP PACK
1 110
11 CANADA 0.0000001000
9 azure
19 Brand#53 Brand#53 Brand#32 3 11 21
3 MACHINERY 1995-03-24
13 express requests
5 AFRICA 1995-01-01
7 CANADA CHINA
10 1993-02-01
16 Brand#51 MEDIUM POLISHED 19 15
48 6 30 38 12 9
6 1995-01-01 0.03 25
14 1995-01-01
15 1995-03-01
12 AIR SHIP 1995-01-01

Appendix F Benchmark Scripts

F.1 run_tpch

```
#!/bin/ksh

if (( $# < 1 ))
then
        echo
        echo "Usage: run_tpch scope scale_factor (plus
additional args depending upon scope)"
        echo
        echo "scope values:"
        echo ""
        echo "          - load          load the tpch test
database"
        echo ""
        echo "          - power  do a single power run "
        echo "                                  (with or without
refresh streams)"
        echo
        echo "          - throughput    do a single
throughput run"
        echo "                                  (with or without
refresh streams)"
        echo
        echo "          - refresh       run refresh pair(s)"
        echo
        echo "          - perf          run a single
power-throughput cycle (without a load)"
        echo
        echo "          - all           do a full-audit run
(load + 2 power-throughput cycles)"
        echo
        echo "          -audit_only     Runs only
audit_scripts."
        echo
        echo "To get usage help on individual scope options,
type"
        echo
        echo "                  run_tpch scope help"
        echo
        exit 1
fi
unset STREAM_COUNT
nParams=$#
i=2

if [ ! -z "$2" ] && [ $2 != help ]
then
    while [ $i -le $nParams ]
    do
        eval j=\$$i
        junk=`expr $j + 1` 2>/dev/null
        if [ $? != 0 ] || [ -z "$junk" ]
        then
            echo "input parameter $i supplied is not correct"
            echo " Please run ./run_tpch help to get usage
information."
            exit
        fi
        i=`expr $i + 1`
    done
fi

SCOPE=$1

if [ $SCOPE = audit_only ]
then
        if [ $# -lt 2 ] || [ $2 = help ]
        then

                echo
                echo "Usage: run_tpch $SCOPE
scale_factor"
                echo ""
                echo "    -- This runs the audit scripts and
these need to be run before and after perf test."
                echo
                exit
        fi
        SCALE_FACTOR=$2
        . ./bm_setup $SCALE_FACTOR
        echo
        echo "run_tpch SCOPE=$SCOPE"

elif [ $SCOPE = load ]
then
        if [ $# -lt 2 ] || [ $2 = help ]
        then
                echo
                echo "Usage: run_tpch $SCOPE scale_factor
[enable_monitoring] [stream_count] "
                echo ""
                echo "                -- Default value for "
                echo "
        enable_monitoring: 0 (no monitoring)"
                echo "                                stream_count: will
be set to minimum required "
                echo "
        stream count for the supplied "
                echo "
        Scale Factor        "
                echo
                echo '                -- This generates
$stream_count number of query streams'
                echo
                exit
        else
                SCALE_FACTOR=$2

                if [ -z "$3" ]
                then
                        Enable_Monitoring=0; export
Enable_Monitoring
```

```
                else
                        Enable_Monitoring=$3; export
Enable_Monitoring
                fi

                if [ ! -z "$4" ]
                then
                        STREAM_COUNT=$4; export
STREAM_COUNT
                fi

                . ./bm_setup $SCALE_FACTOR
#bm_setup call shouldn't be before $# variables assignment.

                echo
                echo
                echo "run_tpch SCOPE=$SCOPE
SCALE_FACTOR=$SCALE_FACTOR
ENABLE_MONITORING=$Enable_Monitoring
STREAM_COUNT=$STREAM_COUNT"
                echo
                echo
        fi
elif [ $SCOPE = power ]
then
        if [ $# -lt 2 ] || [ $2 = help ]
        then
                echo
                echo "Usage: run_tpch $SCOPE scale_factor
[with_rf] [enable_monitoring]"
                echo ""
                echo "               -- Default value for "
                echo "                       a) with_rf option:
1 (run with refresh streams)"
                echo "                       b)
enable_monitoring: 0 (no monitoring)"
                echo
                exit
        else
         SCALE_FACTOR=$2

                if [ -z "$3" ]
                then
                        WITH_RF=1
                else
                        WITH_RF=$3
                fi

                if [ -z "$4" ]
                then
                        Enable_Monitoring=0; export
Enable_Monitoring
                else
                        Enable_Monitoring=$4; export
Enable_Monitoring
                fi

                . ./bm_setup $SCALE_FACTOR
                echo
                echo
                echo "run_tpch SCOPE=$SCOPE
SCALE_FACTOR=$SCALE_FACTOR
WITH_RF=$WITH_RF
ENABLE_MONITORING=$Enable_Monitoring "
                echo
                echo
        fi

elif [ $SCOPE = throughput ] || [ $SCOPE = all ]
then
        if [ $# -lt 2 ] || [ $2 = help ]
        then
                echo
                echo "Usage: run_tpch $SCOPE scale_factor
[with_rf] [enable_monitoring]"
                echo "
        [stream-count]"
                echo "               -- Default value for "
                echo "                       a) with_rf option:
1 (run with refresh streams)"
                echo "                       b)
enable_monitoring: 0 (no monitoring)"
                echo "                       c) stream_count:
will be set to minimum required"
                echo "
stream count for the respective"
                echo "
scale factor."
                echo ""
                echo
                exit
        else
         SCALE_FACTOR=$2

                if [ -z "$3" ]
                then
                        WITH_RF=1
                else
                        WITH_RF=$3
                fi

                if [ -z "$4" ]
                then
                        Enable_Monitoring=0; export
Enable_Monitoring
                else
                        Enable_Monitoring=$4; export
Enable_Monitoring
                fi

                if [ ! -z "$5" ]
                then
                        STREAM_COUNT=$5; export
STREAM_COUNT
                fi
```

```
		. ./bm_setup $SCALE_FACTOR

		echo
		echo
		echo "run_tpch SCOPE=$SCOPE
SCALE_FACTOR=$SCALE_FACTOR
WITH_RF=$WITH_RF
ENABLE_MONITORING=$Enable_Monitoring
STREAM_COUNT=$STREAM_COUNT"
		echo
		echo
	fi

elif [ $SCOPE = perf ]
then
	if [ $# -lt 3 ] || [ $2 = help ]
	then
		echo
		echo "Usage: run_tpch $SCOPE scale_factor
run_number [with_rf] [enable_monitoring]"
		echo "
	[stream-count]"
		echo "		-- Default value for "
		echo "			a) with_rf option:
1 (run with refresh streams)"
		echo "			b)
enable_monitoring: 0 (no monitoring)"
		echo "			c) stream_count:
will be set to minimum required"
		echo "
stream count for the respective"
		echo "
scale factor."
		echo ""
		echo
		exit
	else
	 SCALE_FACTOR=$2
		RUN_NUMBER=$3
		if [ -z "$4" ]
		then
			WITH_RF=1
		else
			WITH_RF=$4
		fi

		if [ -z "$5" ]
		then
			Enable_Monitoring=0; export
Enable_Monitoring
		else
			Enable_Monitoring=$5; export
Enable_Monitoring
		fi

		if [ ! -z "$6" ]
		then
			STREAM_COUNT=$6; export
STREAM_COUNT
		fi

		. ./bm_setup $SCALE_FACTOR

		echo
		echo
		echo "run_tpch SCOPE=$SCOPE
SCALE_FACTOR=$SCALE_FACTOR
RUN_NUMBER=$RUN_NUMBER WITH_RF=$WITH_RF
ENABLE_MONITORING=$Enable_Monitoring
STREAM_COUNT=$STREAM_COUNT"
		echo
		echo
	fi

elif [ $SCOPE = refresh ]
then
	if [ "$2" = help ]
	then
		echo
		echo "Usage: run_tpch $SCOPE
$SCALE_FACTOR [stream_count] [enable_monitoring]"
		echo ""
		echo "		-- Default value for "
		echo "			a) stream_count: 1
(runs one refresh pair)"
		echo "			b)
enable_monitoring: 0 (no monitoring)"
		echo ""
		exit
	else
		SCALE_FACTOR=$2
		if [ -z "$3" ]
		then
			STREAM_COUNT=1; export
STREAM_COUNT
		else
			STREAM_COUNT=$3; export
STREAM_COUNT
		fi

		if [ -z "$4" ]
		then
			Enable_Monitoring=0; export
Enable_Monitoring
		else
			Enable_Monitoring=$4; export
Enable_Monitoring
		fi

		. ./bm_setup $SCALE_FACTOR

		echo "run_tpch SCOPE=$SCOPE
SCALE_FACTOR=$SCALE_FACTOR
STREAM_COUNT=$STREAM_COUNT
ENABLE_MONITORING=$Enable_Monitoring"
```

```
        fi
else
        echo "Please supply valid name for scope"
        exit 1
fi

if [ $SCOPE = all ]
then
        echo " Audit run started `date` "
        bld_system $SCALE_FACTOR
        RET_STATUS=$?
        if [ RET_STATUS -eq 1 ]; then
                echo "The bld_system step reported failure.
Aborting full-audit run ..."
                exit 1
        elif [ RET_STATUS -eq 2 ]; then
                exit
        fi
fi

if [ -z "$RUN_NUMBER" ]
then
    RUN_NUMBER=0
fi

#Set Results directory with seq_number
SEQ_NUMBER=`cat ${DEV_DIR}/data/seq_number`
RESULTS=${RESULTS}/run_${SEQ_NUMBER};export
RESULTS
mkdir -p $RESULTS > /dev/null 2>&1
SEQ_NUMBER=`expr $SEQ_NUMBER + 1 `
echo ${SEQ_NUMBER}>${DEV_DIR}/data/seq_number

$STOP_SERVER -stop all
rm $TPCH_ROOT/plans/* > /dev/null 2>&1

os=`uname`
if [ $os = AIX ]
then
        rm -f $RESULTS/sysunused
        echo "The following disks are assigned to the
indicated volume groups">$RESULTS/sysunused
        lspv>>$RESULTS/sysunused
        echo "The following volume groups are currently
online">>$RESULTS/sysunused
        echo `date` >>$RESULTS/sysunused
        lsvg -o >>$RESULTS/sysunused
fi

#Flat file reference data check.
if [ $SCOPE = 'all' ]
then
        reference_data_check.sh $SCALE_FACTOR
        sleep 100
fi

echo Restart IQ with TPCH database: `date`
$START_SERVER @$CONFIG_DIR/tpch.cfg
$DEV_DIR/tpch.db
echo IQ restarted: `date`
echo "  "


if [ $SCOPE = 'all'  -o $SCOPE = 'load' ]
then
        # Load the tpch test database
        load_test $Enable_Monitoring
fi

if [ $SCOPE = 'all' ]
then
        start_log_details=`ls -l $DEV_DIR/tpch.log`
        echo "$start_log_details" > $RESULTS/mlog
fi

#Generate streams & set the Debug mode.
cwd=`pwd`
cd $QUERYGEN
SEED=`cat $DEV_DIR/data/seed`
. ./querygen.sh $SEED $SCALE_FACTOR
${STREAM_COUNT}
echo "
cd $cwd

if [ $SCOPE = 'refresh' ]
then
        run_refresh.sh $STREAM_COUNT
$Enable_Monitoring
fi

echo
echo
# After the load Completes run the Audit SQL if SCOPE=all
if [ $SCOPE = 'all' -o $SCOPE = 'audit_only' -o $SCOPE =
'perf' -a $RUN_NUMBER = '1' ]
then
        echo "Running the Audit Script   `date` "
        dbisqlc -c "DSN=tpch_$VERSION" -q
$SQL/dbtables-syb.sql > $RESULTS/rdbtablest
        dbisqlc -c "DSN=tpch_$VERSION" -q
$SQL/dew_cat1.sql > $RESULTS/dew_cat1_start.out
        dbisqlc -c "DSN=tpch_$VERSION" -q
$SQL/dew_cat2.sql > $RESULTS/dew_cat2_start.out
        dbisqlc -c "DSN=tpch_$VERSION" -q
$SQL/dew_cat3.sql > $RESULTS/dew_cat3_start.out
        dbisqlc -c "DSN=tpch_$VERSION" -q
$SQL/referential_integrity.sql >
$RESULTS/referential_integrity.out
fi

if [ $SCOPE = 'all' -o $SCOPE = 'power' -o $SCOPE = 'perf' -o
$SCOPE = 'throughput' ]
then
        SEED=`cat $DEV_DIR/data/seed`;export SEED
fi
```

```
# reboot server after load
if [ $SCOPE = 'all' ]
then
        $STOP_SERVER_WITH_CHECKPOINT -stop all
        sleep 200
        get_db_qgen_versions.sh
        echo Restart IQ with TPCH database: `date`
        $START_SERVER @$CONFIG_DIR/tpch.cfg
$DEV_DIR/tpch.db
        echo IQ restarted: `date`
        echo "  "
fi


if [ $SCOPE = 'all' -o $SCOPE = 'perf' ]
then
        ################## FIRST RUN
#################
        #Run power test.
        powertest.sh Run $SCALE_FACTOR $WITH_RF
$Enable_Monitoring

        #Run throughput test.
        throughputtest.sh Run $SCALE_FACTOR
$WITH_RF $Enable_Monitoring $STREAM_COUNT

        echo "  "
        if [ $SCOPE = 'all' ]
        then
                tpch_report.sh $SCOPE $SCALE_FACTOR
$WITH_RF 0 0 1 $SEED $STREAM_COUNT 0  0 0 >
$RESULTS/mrun1_report.out
        fi

        if [ $SCOPE = 'perf' ]
        then
                tpch_report.sh $SCOPE $SCALE_FACTOR
$WITH_RF 0 0 $RUN_NUMBER $SEED
$STREAM_COUNT > $RESULTS/perf.out
        fi
fi

if [ $SCOPE = 'all' ]
then
        #Move the $TPCH_ROOT/plans to $RESULTS/plans.
        plan_count=$(ls -1 $TPCH_ROOT/plans/|wc -l)

        if [ $plan_count -ne 0 ]
        then
        mkdir $RESULTS/plans > /dev/null 2>&1
        mv $TPCH_ROOT/plans/*.html $RESULTS/plans/ >
/dev/null 2>&1
        fi

        move_out_files 1 $WITH_RF > /dev/null 2>&1
fi

if [ $SCOPE = 'all' ]
then
        # reboot server after run1
    $STOP_SERVER_WITH_CHECKPOINT -stop all
    sleep 100
    echo Restart IQ with TPCH database: `date`
    $START_SERVER @$CONFIG_DIR/tpch.cfg
$DEV_DIR/tpch.db
    echo IQ restarted: `date`
    echo "  "

        ################SECOND RUN
####################
        #Run power test.
        powertest.sh Run $SCALE_FACTOR $WITH_RF
$Enable_Monitoring

        #Run throughput test.
        throughputtest.sh Run $SCALE_FACTOR
$WITH_RF $Enable_Monitoring $STREAM_COUNT

        echo "  "
        tpch_report.sh all $SCALE_FACTOR $WITH_RF 0 0
2 $SEED $STREAM_COUNT 0 0 0 >
$RESULTS/mrun2_report.out
        ##########################################
####
fi
if [ $SCOPE = 'all' -o $SCOPE = 'perf' -a $RUN_NUMBER =
'2' ]
then
        dbisqlc -c "DSN=tpch_$VERSION" -q
$SQL/dew_cat1.sql > $RESULTS/dew_cat1_end.out
        dbisqlc -c "DSN=tpch_$VERSION" -q
$SQL/dew_cat2.sql > $RESULTS/dew_cat2_end.out
        dbisqlc -c "DSN=tpch_$VERSION" -q
$SQL/dew_cat3.sql > $RESULTS/dew_cat3_end.out
fi

if [ $SCOPE = 'all' -o $SCOPE = 'load' ]
then
        cat $RESULTS/ldeventlog |tr -s ' ' |tr -s '\n'|tr -s
'='>$RESULTS/ldeventlog.`date '+%y%m%d_%H%M%S'`
        rm $RESULTS/ldeventlog
        if [ $SCOPE = 'all' ]
        then
                move_out_files 2 $WITH_RF > /dev/null
2>&1
        fi
fi
if [ $SCOPE = 'all' -o $SCOPE = 'perf' ]
then
        generate_mlog.sh
        show_rf_impl
        aixmachine=`uname |grep -i aix`
        if [ ! -z "$aixmachine" ]
        then
        aix_sys_conf.sh $RESULTS/sysconfig
```

```
                machinename=`uname -n`
                cp $RESULTS/sysconfig/SoftwareConfig*
$RESULTS/syssw.`date
'+%y%m%d_%H%M%S'`.$machinename
                cp $RESULTS/sysconfig/Tunables*
$RESULTS/sysostune.`date
'+%y%m%d_%H%M%S'`.$machinename

                echo "Device Config info" >
$RESULTS/syshw
                echo "=================" >>
$RESULTS/syshw
                cat $RESULTS/sysconfig/DeviceConfig* >>
$RESULTS/syshw

                echo "File System  info" >>
$RESULTS/syshw
                echo "=================" >>
$RESULTS/syshw
                cat
$RESULTS/sysconfig/FilesystemConfig*>>
$RESULTS/syshw

                echo "Hardware confi info" >>
$RESULTS/syshw
                echo "===================" >>
$RESULTS/syshw
                cat $RESULTS/sysconfig/HardwareConfig*
>> $RESULTS/syshw

                echo "Physical volume List info" >>
$RESULTS/syshw
                echo "=========================" >>
$RESULTS/syshw
                cat
$RESULTS/sysconfig/Physical_Volume_List* >>
$RESULTS/syshw


                echo "Volume Detail info" >>
$RESULTS/syshw
                echo "==================" >>
$RESULTS/syshw
                cat $RESULTS/sysconfig/Volume_Detail*
>> $RESULTS/syshw

                echo "Volume_Group_List info" >>
$RESULTS/syshw
                echo "======================" >>
$RESULTS/syshw
                cat
$RESULTS/sysconfig/Volume_Group_List* >>
$RESULTS/syshw

                echo "Volume_Group_Member_list info" >>
$RESULTS/syshw
                echo
"============================" >>
$RESULTS/syshw
                cat
$RESULTS/sysconfig/Volume_Group_Member_list* >>
$RESULTS/syshw

                echo "Volume_Group_Pdisk_list info" >>
$RESULTS/syshw
                echo "=========================="
>> $RESULTS/syshw
                cat
$RESULTS/sysconfig/Volume_Group_Pdisk_list* >>
$RESULTS/syshw
                mv $RESULTS/syshw
$RESULTS/syshw.`date
'+%y%m%d_%H%M%S'`.$machinename
        fi

        linuxmachine=`uname |grep -i linux`
    if [ ! -z "$linuxmachine" ]
        then
                linux_sys_conf.sh $RESULTS
                machinename=`uname -n`
                mv $RESULTS/syssw
$RESULTS/syssw.`date
'+%y%m%d_%H%M%S'`.$machinename
                mv $RESULTS/sysostune
$RESULTS/sysostune.`date
'+%y%m%d_%H%M%S'`.$machinename
                mv $RESULTS/syshw
$RESULTS/syshw.`date
'+%y%m%d_%H%M%S'`.$machinename
                mv $RESULTS/sysdbtune
$RESULTS/sysdbtune.`date
'+%y%m%d_%H%M%S'`.$machinename
        fi
fi

if [ $SCOPE = 'power' ]
then
        #Run power test.
        powertest.sh All $SCALE_FACTOR $WITH_RF
$Enable_Monitoring
fi

if [ $SCOPE = 'throughput' ]
then
        #Run throughput test.
        throughputtest.sh All $SCALE_FACTOR $WITH_RF
$Enable_Monitoring $STREAM_COUNT
fi

#Clean up
rm $TEMPFILES/* > /dev/null 2>&1

#Move the $TPCH_ROOT/plans to $RESULTS/plans.
plan_count=$(ls -1 $TPCH_ROOT/plans/|wc -l)
```

```
if [ $plan_count -ne 0 ]
then
        mkdir $RESULTS/plans > /dev/null 2>&1
        mv $TPCH_ROOT/plans/*.html $RESULTS/plans/ >
/dev/null 2>&1
fi
if [ $os = AIX ]
then
        mv $RESULTS/sysunused
$RESULTS/sysunused.`date '+%y%m%d_%H%M%S'`
fi

#unset STREAM_COUNT. Otherwise same value will be
remain in consecutive runs.
unset STREAM_COUNT

if [ $SCOPE = all ]
then
        echo " Audit run ended `date` "
fi
```

F.2 powertest.sh

```
if [ -z "$2" ]
then
      echo 'Usage: powertest.sh <SCOPE>
<SCALE_FACTOR> [WITH_RF] [Enable_Monitoring]'
        echo 'Allowed values for scope Report, Run, All'
      exit 1
fi

SCOPE=$1
SCALE_FACTOR=$2
if [ -z "$3" ]
then
        WITH_RF=1
else
        WITH_RF=$3
fi

if [ -z "$4" ]
then
     Enable_Monitoring=0
else
     Enable_Monitoring=$4
fi

CURRENTLY_RUNNING=power; export
CURRENTLY_RUNNING

if [ $Enable_Monitoring -ne 0 ] && [ $DEBUG -eq 0 ]
then
      start_monitor.sh
fi

if [ $SCOPE = 'Run' -o $SCOPE = 'All' ];
then
        ##  Check power test is allowed or not
                ##
        ##  It is allowed unconditionally if executed
without_RFs             ##
        ##  It is not allowed if RFs are executed more than or
equal to  ##
        ##  MAX_TIMES RFs are allowed to execute before
RELOAD                  ##

        if [ $WITH_RF -ne 0 ];
        then
                MAX_STREAMS=`ls
${DEV_DIR}/data/lineitem.tbl.u*| wc -l`
                LAST_STREAM=`cat
${DEV_DIR}/data/last_stream`

                if [ $LAST_STREAM -ge
$MAX_STREAMS ];
                        then
                        echo "Please reload the data as total
STREAMS executed is exceeded the
MAX_ALLOWED_STREAMS"
                        exit 1
                fi
                gen_update_power.sh $LAST_STREAM
        fi

        echo "  "
        echo "Start the Power Test   `date` "
        echo "  "

        if [ $WITH_RF -ne 0 ];
        then
                #
                # Touch the lock files so that the RF will
pause after RF1 and wait for the query stream to complete
                #
                touch $TPCH_ROOT/rf1.lock
                touch $TPCH_ROOT/rf2.lock
                #
                # Start the RF Stream in the Background
                #
                $ISQL -i
$GENERATED_FILES/update_power.sql >
$RESULTS/update_power.out &
                rfspid=$!
                #
                # Wait for RF1 to Complete before Starting
the Query Stream
                #
                # the rf1.lock file will be removed after RF1
completes
                # RF will then wait for rf2.lock to be
removed before continuing
                #
                while [ -f $TPCH_ROOT/rf1.lock ]
```

```
                do
                        # Sleep while Trigger file exists
                        # echo "Waiting until trigger file is
removed"
                        sleep 10
                done
                #
                # Continue RF1 has completed and RF2 is
waiting on the Query Stream to Complete
                #
                echo "  "
        fi

        #
        # Start the Query Stream
        #
        $ISQL -i $STREAMS/stream0.sql -e >
$RESULTS/stream0.out


        if [ $WITH_RF -ne 0 ];
        then
                #
                # Remove the lock file so that the RF will
continue with RF2
                #
                `\rm -f $TPCH_ROOT/rf2.lock`
                #
                # Now wait for the RF stream 0 to complete
                #

                wait $rfspid
        fi

        echo End Power `date`
        if [ $WITH_RF -ne 0 ];
        then
                LAST_STREAM=`expr $LAST_STREAM +
1`
                echo $LAST_STREAM >
$DEV_DIR/data/last_stream
        fi
fi

if [ $Enable_Monitoring -ne 0 ] && [ $DEBUG -eq 0 ]
then
      kill_stats.sh
fi

if [ $SCOPE = 'Report' -o $SCOPE = 'All' ];
then
        tpch_report.sh power $SCALE_FACTOR $WITH_RF
0  0 power $SEED > $RESULTS/powertest.out
fi

os=$(uname)
if [ $os = AIX ]
then
        echo "After completion of the Power Test" >>
$RESULTS/sysunused
        echo "The following volume groups are currently
online" >> $RESULTS/sysunused
        echo `date` >> $RESULTS/sysunused
        echo>> $RESULTS/sysunused
        lsvg -o >> $RESULTS/sysunused
fi
```

F.3 throughputtest.sh

```
if [ -z "$2" ]
then
      echo 'Usage: throughtest.sh <SCOPE>
<SCALE_FACTOR> [WITH_RF] [Enable_Monitoring]
[STREAM_COUNT]'
      echo 'Allowed values for scope Report, Run, All'
      exit 1
fi

SCOPE=$1
SCALE_FACTOR=$2
if [ -z "$3" ]
then
        WITH_RF=1
else
        WITH_RF=$3
fi

if [ -z "$4" ]
then
     Enable_Monitoring=0
else
     Enable_Monitoring=$4
fi

if [ ! -z "$5" ]
then
        STREAM_COUNT=$5
fi

CURRENTLY_RUNNING=throughput; export
CURRENTLY_RUNNING

if [ $Enable_Monitoring -ne 0 ]
then
        start_monitor.sh
fi

if [ $SCOPE = 'Run' -o $SCOPE = 'All' ];
then
        ##  Check THROUGHPUT test is allowed or not.
##
        ##  It is allowed unconditionally if the run is wihtout
RFs.      ##
```

```
        ##  It is not allowed if
NO_OF_TIMES_RFs_EXECUTED + STREAM_COUNT >
##
        ##  MAX_TIMES RFs are allowed to execute before
RELOAD          ##

        if [ $WITH_RF -ne 0 ];
        then
                MAX_STREAMS=`ls
$DEV_DIR/data/lineitem.tbl.u*|wc -l`
                LAST_STREAM=`cat
$DEV_DIR/data/last_stream`
                REMAINING_STREAMS=`expr
$MAX_STREAMS - $LAST_STREAM`
                if [ $STREAM_COUNT -gt
$REMAINING_STREAMS ];
                then
                        echo "Please reload the data as total
STREAMS executed is exceeded the
MAX_ALLOWED_STREAMS"
                        exit 1
                fi
                gen_update_throughput.sh
$STREAM_COUNT $LAST_STREAM
        fi



        echo "  "
        echo Start Throughput `date`
        echo "  "
        echo Running Query Streams
        for i in `seq 1 $STREAM_COUNT`
        do
                $ISQL -i $STREAMS/stream${i}.sql -e >
$RESULTS/stream${i}.out &
                export qspid${i}=$!
        done

        sleep 60
        if [ $WITH_RF -ne 0 ];
        then
                echo Running refresh streams
                $ISQL -i
$GENERATED_FILES/update_throughput.sql >
$RESULTS/update_throughput.out &
                rf0pid=$!

                # Wait for the refresh streams to complete.
                wait $rf0pid
        fi
        # Wait for the query streams to complete.
        for i in `seq 1 $STREAM_COUNT`
        do
                eval temp=\$qspid$i
                wait $temp
        done

        echo End Throughput test: `date`
        echo ""

        #Don't update LAST_STREAM if RF is not run.
        if [ $WITH_RF -ne 0 ];
        then
                LAST_STREAM=`expr $LAST_STREAM +
$STREAM_COUNT`
                echo $LAST_STREAM
>$DEV_DIR/data/last_stream
        fi
fi

if [ $Enable_Monitoring -ne 0 ]
then
      kill_stats.sh
fi



if [ $SCOPE = 'Report' -o $SCOPE = 'All' ];
then
        tpch_report.sh throughput $SCALE_FACTOR
$WITH_RF 0 0 tput $SEED $STREAM_COUNT >
$RESULTS/throughput_report.out
fi

os=$(uname)
if [ $os = AIX ]
then
        echo "After completion of the Throughput Test" >>
$RESULTS/sysunused
        echo "The following volume groups are currently
online" >> $RESULTS/sysunused
        echo `date` >> $RESULTS/sysunused
        echo>> $RESULTS/sysunused
        lsvg -o >> $RESULTS/sysunused
fi
```

F.4 dbtables-syb.sql

```
--
==========================================
====================+
--  FILENAME
--      DBTABLES.SQL
--  DESCRIPTION
--      CHECK ROW COUNT AND ROW
STRUCTURE/CONTENT FOR EACH TABLE
--      IN THE TPC-H DATABASE.
--
--
==========================================
====================
--
-- GET TIMESTAMP
SELECT 'START TIME', CONVERT(CHAR(30),
GETDATE(), 120);
go
```

```
--
================================================================
--          TABLE: LINEITEM
--
================================================================
SELECT COUNT(*) FROM LINEITEM;
go
SELECT l_orderkey, l_partkey, l_suppkey, l_linenumber,
cast(l_quantity as NUMERIC(30,2)), cast(l_extendedprice as
NUMERIC(30,2)), cast(l_discount as NUMERIC(30,2)),
cast(l_tax as NUMERIC(30,2)), l_returnflag, l_linestatus,
l_shipdate, l_commitdate, l_receiptdate, l_shipinstruct,
l_shipmode, l_comment FROM LINEITEM
WHERE L_ORDERKEY IN
    ( 4, 26598,  148577,  3871431, 46556704, 1925174742,
6000000000)
    AND L_LINENUMBER = 1
ORDER BY L_ORDERKEY;
go
--
================================================================
--          TABLE: ORDERS
--
================================================================
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM ORDERS;
go
SELECT o_orderkey, o_custkey, o_orderstatus,
cast(o_totalprice as NUMERIC(30,2)), o_orderdate,
o_orderpriority, o_clerk, o_shippriority, o_comment FROM
ORDERS
WHERE O_ORDERKEY IN ( 7, 44065, 287590, 4111111,
24483876, 2901689942 )
ORDER BY O_ORDERKEY;
go
--
================================================================
--          TABLE: PART
--
================================================================
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM PART;
go
SELECT p_partkey, p_name, p_mfgr, p_brand, p_type, p_size,
p_container, cast(p_retailprice as NUMERIC(30,2)),
p_comment FROM PART
WHERE P_PARTKEY IN
(1,9804,86743,976028,173876,16777899,500000000)
ORDER BY P_PARTKEY;
go
--
================================================================
--          TABLE: PARTSUPP
--
================================================================
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM PARTSUPP;
go
SELECT ps_partkey, ps_suppkey, ps_availqty,
cast(ps_supplycost as NUMERIC(30,2)), ps_comment FROM
PARTSUPP
  WHERE PS_PARTKEY = 3398
  AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
    FROM PARTSUPP WHERE PS_PARTKEY = 3398);
go
SELECT ps_partkey, ps_suppkey, ps_availqty,
cast(ps_supplycost as NUMERIC(30,2)), ps_comment  FROM
PARTSUPP
  WHERE PS_PARTKEY =1587553
  AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
    FROM PARTSUPP WHERE PS_PARTKEY = 1587553);
go
SELECT ps_partkey, ps_suppkey, ps_availqty,
cast(ps_supplycost as NUMERIC(30,2)), ps_comment  FROM
PARTSUPP
  WHERE PS_PARTKEY = 1139412
  AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
    FROM PARTSUPP WHERE PS_PARTKEY = 1139412);
go
SELECT ps_partkey, ps_suppkey, ps_availqty,
cast(ps_supplycost as NUMERIC(30,2)), ps_comment  FROM
PARTSUPP
  WHERE PS_PARTKEY = 67436543
  AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
    FROM PARTSUPP WHERE PS_PARTKEY = 67436543);
go
SELECT ps_partkey, ps_suppkey, ps_availqty,
cast(ps_supplycost as NUMERIC(30,2)), ps_comment FROM
PARTSUPP
  WHERE PS_PARTKEY = 197637894
  AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
    FROM PARTSUPP WHERE PS_PARTKEY
=197637894);
go
--
================================================================
--          TABLE: SUPPLIER
--
================================================================
-- GET TIMESTAMP
```

```
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM SUPPLIER;
go
SELECT s_suppkey, s_name, s_address, s_nationkey, s_phone,
cast(s_acctbal as NUMERIC(30,2)), s_comment FROM
SUPPLIER
WHERE  S_SUPPKEY IN
(83,265,5492,78454,905871,3000000)
ORDER BY S_SUPPKEY;
go
--
================================================
====================
--        TABLE: CUSTOMER
--
================================================
====================
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM CUSTOMER;
go
SELECT c_custkey, c_name, c_address, c_nationkey, c_phone,
cast(c_acctbal as NUMERIC(30,2)), c_mktsegment,
c_comment FROM CUSTOMER
WHERE  C_CUSTKEY IN
(832,2653,4924,7845,92016,108070)
ORDER BY C_CUSTKEY;
go
--
================================================
====================
--        TABLE: NATION & REGION
--
================================================
====================
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT * FROM REGION;
go
SELECT COUNT(*) FROM NATION;
go
SELECT * FROM NATION
WHERE N_NATIONKEY IN (3,10,14,20)
ORDER BY N_NATIONKEY;
go
--
================================================
====================
--        CHECK KEY VALUES
--
================================================
====================
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
if exists (select name from sysobjects where name='MINMAX')
  drop table MINMAX
go
CREATE TABLE MINMAX
(TNAME CHAR(15),
KEYMIN bigint,
KEYMAX bigint);
go
INSERT INTO MINMAX
SELECT
'LINEITEM_ORD',MIN(L_ORDERKEY),MAX(L_ORDERK
EY)
FROM LINEITEM;
go
INSERT INTO MINMAX
SELECT
'LINEITEM_NBR',MIN(L_LINENUMBER),MAX(L_LINEN
UMBER)
FROM LINEITEM;
go
INSERT INTO MINMAX
SELECT
'ORDERS',MIN(O_ORDERKEY),MAX(O_ORDERKEY)
FROM ORDERS;
go
INSERT INTO MINMAX
SELECT
'CUSTOMER',MIN(C_CUSTKEY),MAX(C_CUSTKEY)
FROM CUSTOMER;
go
INSERT INTO MINMAX
SELECT 'PART',MIN(P_PARTKEY),MAX(P_PARTKEY)
FROM PART;
go
INSERT INTO MINMAX
SELECT
'SUPPLIER',MIN(S_SUPPKEY),MAX(S_SUPPKEY)
FROM SUPPLIER;
go
INSERT INTO MINMAX
SELECT
'PARTSUPP_PART',MIN(PS_PARTKEY),MAX(PS_PARTK
EY)
FROM PARTSUPP;
go
INSERT INTO MINMAX
SELECT
'PARTSUPP_SUPP',MIN(PS_SUPPKEY),MAX(PS_SUPPKE
Y)
FROM PARTSUPP;
go
INSERT INTO MINMAX
SELECT
'NATION',MIN(N_NATIONKEY),MAX(N_NATIONKEY)
FROM NATION;
go
INSERT INTO MINMAX
```

```
SELECT
'REGION',MIN(R_REGIONKEY),MAX(R_REGIONKEY)
FROM REGION;
go
SELECT * FROM MINMAX;
go
if exists (select name from sysobjects where name='MINMAX')
  drop table MINMAX
go
SELECT 'END TIME', CONVERT(CHAR(30), GETDATE(),
120);
go
```

F.5 dew_cat1.sql

```
SELECT  st.table_name,
                st.table_type,
                su.user_name,
                st.server_type
           from SYS.SYSTABLE st, SYS.SYSUSERPERMS
su
           where creator = user_id
order by 4,1,3
;
```

F.6 dew_cat2.sql

```
          select T.table_name    ,
                   T.table_type    ,
               C.column_name   ,
               C.column_id
          From   SYS.SYSTABLE T, SYS.SYSCOLUMN C,
SYS.SYSDOMAIN D,
               SYS.SYSUSERPERMS SU
          where T.creator = SU.user_id
            and T.table_id = C.table_id
        and C.domain_id = D.domain_id
      order by 1,2;
```

F.7 dew_cat3.sql

```
SELECT  index_name,T.table_name  ,
                column_name
                index_type
           from   SYS.SYSTABLE T, SYS.SYSCOLUMN C,
SYS.SYSINDEX I,
                SYS.SYSUSERPERMS UP, SYS.SYSFILE F,
SYS.SYSIXCOL IC
           where T.table_id = C.table_id
             and C.table_id = I.table_id
             and T.file_id = F.file_id
             and I.table_id = IC.table_id
             AND I.index_id = IC.index_id
                AND IC.column_id = C.column_id
                and T.creator = UP.user_id
;
```

F.8 get_db_qgen_versions.sh

```
#!/bin/ksh
dbgen -h >$TEMPFILES/dbgen 2>&1
qgen -h >$TEMPFILES/qgen 2>&1
head -2 $TEMPFILES/dbgen
>$RESULTS/dbgen_qgen_versions
head -2 $TEMPFILES/qgen
>>$RESULTS/dbgen_qgen_versions
```

F.9 querygen.sh

```
#!/bin/ksh

if [ -z "$1" -o -z "$2" ]
then
        echo 'usage querygen <random_seed> <scale_factor>
[stream_count]'
        exit 1
fi
RANDOM_SEED=$1
SCALE_FACTOR=$2
if [ ! -z "$3" ]
then
        STREAM_COUNT=$3
elif [ ! -z "$STREAM_COUNT" ]
then
        echo "
        echo '$STREAM_COUNT not defined.'
        echo 'Pass STREAM_COUNT parameter or run
bm_setup'
        echo "
        exit 1
fi
DEBUG=0;export DEBUG

mkdir -p $STREAMS > /dev/null 2>&1

\rm -f $STREAMS/opts.*
\rm -f $STREAMS/stream*.sql
\rm -f $STREAMS/stream_seeds

for i in `seq 0 $STREAM_COUNT`
do
        SEED=`expr $RANDOM_SEED + $i`;
        qgen -r $SEED  -p $i -c -i $QUERYGEN/begin.sql -t
$QUERYGEN/end.sql -s $SCALE_FACTOR \
                -l $QUERYGEN/opts.$i >
$STREAMS/stream$i.sql

        echo "stream$i $SEED">>$STREAMS/stream_seeds
```

```
        mv -f $QUERYGEN/opts.$i $STREAMS
done
```

Appendix G Price Quotes

Sharada Bose
Performance Manager BCS
Hewlett-Packard
19447 Pruneridge Avenue, MS4220
Cupertino, CA 95014
February 1, 2010

Description	Part Number		Reference Price	Qty	Extended Price	3 yr Maint Price
Server Hardware						
HP ProLiant DL785 G6 Rack CTO Chassis	AM437A	1	9,999	1	9,999	
HP O8439SE 2.8GHz DL785 4p FIO Kit	575261-L21	1	19,499	1	19,499	
HP O8439SE 2.8GHz DL785 4p Kit	575261-B21		19,499	1	19,499	
HP 64GB Reg PC2-5300 8x8GB	495605-B21	1	7,199	8	57,592	
HP 146GB 10k 2.5 Dual Port HP SAS Drive	418367-B21	1	269	8	2,152	
HP 3year 4h 24x7 Proliant DL785 HW Support	HA104A3 Opt.6	1	3,208	1		3,208
HP SA P400 FIO 256MB Cache	405139-B21	1	99	1	99	
HP TFT7600 Rackmount Keyboard 17in Monitor	AG052A	1	1,679	1	1,679	
				Subtotal	**110,519**	**3,208**
Server Software						
RHELAP Ultd SKT 24x7 3Year RHN Nm SW	445214-B21	1	6,747	1	6,747	Included
				Subtotal	**6,747**	**0**
Storage						
HP 5642 Unassembled Rack	358254-B21	1	865	1	865	
HP 82Q 8Gb Dual Port PCI-e FC HBA	AJ764A	1	2,250	8	18,000	
HP 2324fc DC Modular Smart Array	AJ797A	1	8,900	4	35,600	
MSA2000 Array Support	HA104A3 Opt. 9	1	1,513	4		6,052
HP 2m Multi-mode OM2 LC/LC FC Cable	221692-B21	1	75	16	1,200	
HP 146GB 3G SAS 15K 2.5in DP ENT HDD	504062-B21	1	529	96	50,784	
				Subtotal	**106,449**	**6,052**
				Total	**223,715**	**9,260**
16.0 % Large Configuration Discount and Support Prepayment*					(35,779)	(1,482)
				Grand Total	**187,936**	**7,778**
				3-yr Cost of Ownership:		**195,715**

*All discounts are based on US list prices and for similar quantities and configurations

For:

Quotation for Software and Support

Company Hewlett-Packard
Contact Sharada Bose
Phone 408-447-6715
Fax
Address 4220 19447 Pruneridge Avenue
Cupertino, California
USA 95014

SYBASE Sales Rep: Anne Belt
Phone: 925-236-4108
Fax: 925-236-6178
Sybase Inc. 1 Sybase Drive, Dublin, CA 94568

	Catalogue Number	Product Description	License Type	Machine	P/S	List Price Per Unit	Quantity	Price	Discount	Extended Price
1	13545	Sybase IQ Single App Svr, per cpu core	CP	Linux x86-64	P	2,595	48	124,560	18,684	105,876.00
3	98477	3 yr support Single App Svr, per cpu core				1,713	48	82,224	12,334	69,890.40
4		Discounts:								
5		15.00%								
6										
7										
8										
9										
10										
11										
12										

Quote Date: ***1/25/10***
Valid thru: ***2/25/10***

Total 175,766.40

Licence + 3 year support

Payment terms : Net 30 Days