TPC Benchmark™ H
Full Disclosure Report

Sun Microsystems Sun Fire™ X4540 Server Using ParAccel Analytic Database™

Submitted for Review
Report Date: June 21, 2009

TPC Benchmark H Full Disclosure Report

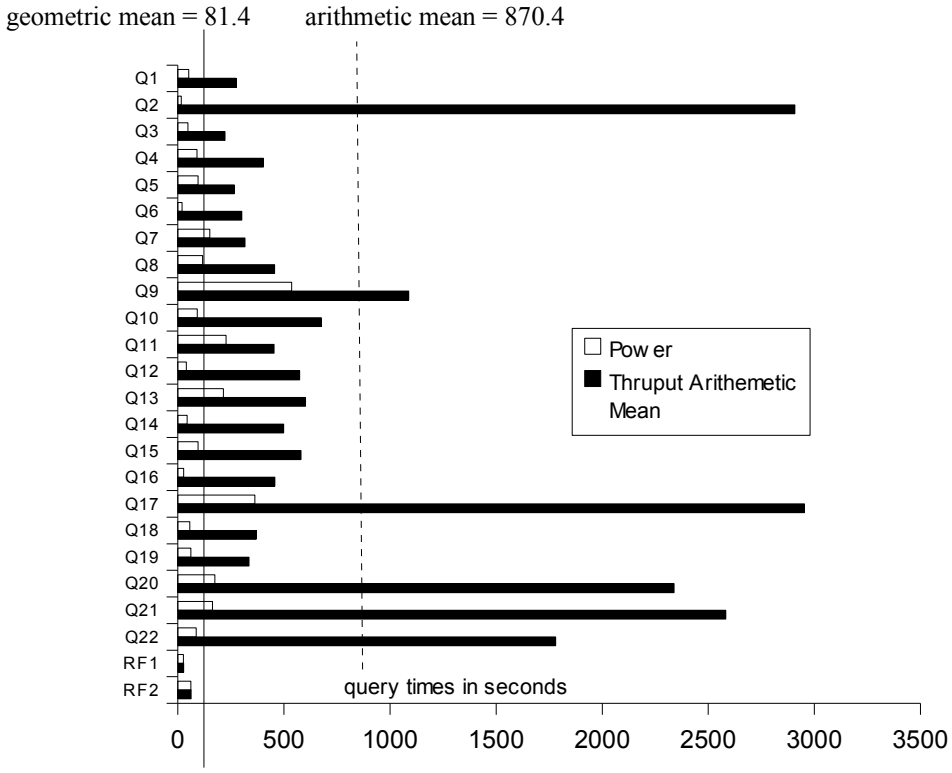
First Printing

901 San Antonio Road, Palo Alto, California 94303 U.S.A.

Sun microsystems / PARACCEL	**Sun Fire™ X4540 Server with ParAccel Analytic Database™**	**TPC-H Rev. 2.8.0** **Report Date: June 21, 2009**

Total System Cost	Composite Query per Hour Metric	Price/Performance
$3,006,861.06	**1,050,556.2** QphH@30000GB	**$2.86** per QphH@30000GB

Database Size	Database Manager	Operating System	Other Software	Availability Date
30000GB	**ParAccel Analytic Database™**	**OpenSolaris2009**	**SVM**	**June 21, 2009**



Database Load Time = 3H28M44Sec	Load Includes Backup: N	Total Storage/Database Size=32.04

RAID (Base tables): Y	RAID (Base tables and auxiliary data structures): Y	RAID (All): Y

System Configuration:

43 x SunFire X4540 Server, each server with
2 x AMD Opteron model 2356 2.3GHz processors (each processor is 1 chip, 4 cores, 4 threads)
64 GB memory
48 x 500,000,000,000 bytes (7.5K RPM) internal SATA disks

Total Storage: 961,124.9 GB

(in this calculation 1 GB is defined as 1024 * 1024 * 1024) bytes

Sun Fire™ X4540 Server with ParAccel Analytic Database™

TPC-H Rev. 2.8.0

Report Date: June 21, 2009

Description	Part Number	Source	Unit Price	Qty	Ext. Price	3 Yr. maint
Server Hardware						
SunFire X4540 M2 x64 Server 2 X AMD Opteron Model 2220 (2.8GHz) processor 64 X 2GB DDR2-667 Memory, 48 X 500 GB HDD	B24-FSZ2-64G-HMH	2	41,995.00	43	1,805,785.00	
Server Discount (18%)		2			-325,041.30	
Sun GOLD Support. 24x7 hardware only support with 4 hour response	B24-FSZ2-64G-HMH	2	2,766.60	43		118,963.80
Cisco Catalyst 3750-24TS 48-port switch	CIS-WS-C3750G-24T-S	3	8,450.00	3	25,350.00	
Cisco Catalyst 3750-24TS 24-port switch	CIS-WS-C3750G-24T-S	3	3,615.00	4	14,460.00	
Belkin Cat5E 7' cable	BEL-A3L781-07BL-CDW	3	1.25	172	215.00	
Cisco premium support for 48 port switch	CD1-CON-OSP-3750G48T	3	1,750.00	9		15,750.00
Cisco premium support for 24 port switch	CD1-CON-OSP-3750G24TS	3	825.00	12		9,900.00
Minuteman Pro1500	CWD-682296	3	266.22	43		11,447.46
Minuteman Pro1500 (10% spares)	CWD-682296	3	266.22	5		1,331.10
Server Hardware Subtotal					1,520,768.70	157,392.36
External Storage None						
Server Software						
ParAccel Analytic Database, per TB data	PAR-ADB	1	100,000.00	30	3,000,000.00	
ParAccel Support, per TB data per year	PAR-SUP-B	1	1,000.00	90		90,000.00
ParAccel Discount 57%		1			-1,710,000.00	-51,300.00
Server Software Subtotal					1,290,000.00	38,700.00
			Total		**2,810,768.70**	**196,092.36**
			3 Yr. Cost		**3,006,861.06**	
			QphH @30000GB		1,050,556	
			$/QphH @30000GB		$2.86	

Service for Sun products from Sun Microsystems, Inc.
Service for ParAccel products is from ParAccel, Inc.
Service for APC Smart-UPS is from cdw.com

Sources
1. ParAccel, Inc.
2. Agilysys
3. cdw.com

Audited by: Francois Raab, InfoSizing, Inc. (www.sizing.com)

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

Sun Fire™ X4540 Server with ParAccel Analytic Database™

Numerical Quantities

TPC-H Rev. 2.8.0

Report Date: June 21, 2009

Measurement Results:

Database Scale Factor	= 30000GB
Total Data Storage / Database Size	= 32.04
Start of database load time	= 16:53:55
End of database load time	= 20:22:39
Database Load Time	= 3H28M44Sec
Query Streams for Throughput Test	= 10
TPC-H Power	= 1,326,910.4
TPC-H Throughput	= 831,758.0
TPC-H Composite Query-per-Hour Rating (QphH@30000GB)	= 1,050,556.2
Total System Price Over 3 Years	= $3,006,861.06
TPC-H Price/Performance Metric ($/QphH@30000GB)	= $2.86

Measurement Intervals:

Measurement Interval in Throughput Test (Ts)	= 28,566 seconds

Duration of Stream Execution:

Stream ID	Seed	Start Date	Start Time	End Date	End Time	Duration (secs)
Stream 0	606202239	6/7/2009	5:08:41 AM	6/7/2009	5:56:23 AM	2774.1
Stream 1	606202240	6/7/2009	5:56:24 AM	6/7/2009	12:44:47 PM	24503.2
Stream 2	606202241	6/7/2009	5:56:24 AM	6/7/2009	1:03:47 PM	25643.5
Stream 3	606202242	6/7/2009	5:56:24 AM	6/7/2009	12:21:25 PM	23101.3
Stream 4	606202243	6/7/2009	5:56:24 AM	6/7/2009	1:24:30 PM	26886.3
Stream 5	606202244	6/7/2009	5:56:24 AM	6/7/2009	11:25:21 AM	19736.9
Stream6	606202245	6/7/2009	5:56:24 AM	6/7/2009	12:29:02 PM	23558.2
Stream7	606202246	6/7/2009	5:56:24 AM	6/7/2009	1:37:15 PM	27651.2
Stream8	606202247	6/7/2009	5:56:24 AM	6/7/2009	11:36:23 AM	20399.2
Stream9	606202248	6/7/2009	5:56:24 AM	6/7/2009	1:12:20 PM	26156.7
Stream10	606202249	6/7/2009	5:56:24 AM	6/7/2009	11:26:56 AM	19832.7

PARACCEL

Sun Fire™ X4540 Server with ParAccel Analytic Database™

TPC-H Rev. 2.8.0

Report Date: June 21, 2009

TPC-H Timing Intervals (in seconds)

Stream ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
00	51.4	16.7	47.6	90.2	95.4	20.0	151.4	117.1	537.2	91.9	227.5	40.2
01	246.2	7,011.0	566.5	163.9	111.4	930.9	515.4	401.0	992.1	891.3	448.9	768.1
02	132.0	3,974.6	120.2	158.0	222.1	20.1	213.8	260.8	2,316.3	145.8	304.1	1,165.4
03	613.8	1,701.9	235.6	528.7	606.1	329.4	228.2	654.0	604.0	301.3	542.8	554.2
04	569.0	2,376.5	287.2	913.1	249.7	489.1	237.4	596.0	1,201.2	1,912.8	858.4	740.3
05	39.3	2,371.2	79.2	568.2	312.3	21.1	552.7	705.1	881.0	709.5	334.9	178.8
06	38.8	3,675.7	762.6	235.8	103.2	25.6	458.8	248.9	2,343.8	92.4	571.4	431.8
07	81.2	4,115.0	48.5	227.8	95.5	19.9	151.6	1,298.0	625.1	71.5	322.9	46.8
08	536.7	1,967.6	391.2	340.9	435.2	882.8	210.6	338.0	823.1	200.9	390.2	176.2
09	130.1	5,164.5	56.5	768.1	132.7	1,042.2	468.4	544.7	524.3	295.0	479.9	106.3
10	439.4	2,161.9	106.6	266.7	853.3	711.1	395.0	199.1	658.2	138.7	368.0	122.2
Minimum	39.3	16.7	47.6	90.2	95.4	20.0	151.4	117.1	537.2	91.9	227.5	40.2
Average	275.3	2,908.6	222.7	403.7	266.2	301.8	316.5	455.7	1,088.6	675.4	452.8	574.5
Maximum	613.8	7,011.0	566.5	913.1	606.1	930.9	552.7	705.1	2,316.3	1,912.8	858.4	1,165.4

Stream ID	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
00	214.5	43.8	95.8	27.3	363.5	56.6	61.5	174.1	163.8	86.6	26.8	61.8
01	392.2	128.7	638.5	826.2	2,070.0	61.5	101.1	3,634.7	2,335.4	1,268.3	31.5	61.2
02	452.7	1,054.5	1,302.7	922.6	6,397.8	139.6	366.8	2,622.6	1,552.1	1,798.7	30.6	61.9
03	693.5	83.5	691.2	123.7	4,285.8	439.9	604.3	3,262.0	2,299.8	3,717.6	30.8	61.5
04	1,546.2	789.9	436.0	28.3	3,270.2	507.7	664.3	1,809.5	6,167.6	1,235.7	11.2	62.5
05	314.3	890.4	319.4	817.3	1,334.9	1,007.5	216.4	2,531.2	2,976.3	2,576.0	31.5	64.0
06	384.5	541.0	1,091.6	357.1	2,754.6	356.8	716.0	2,876.2	2,943.0	2,548.7	30.8	61.6
07	315.1	44.6	94.6	28.3	1,854.4	1,416.7	61.7	5,495.3	8,295.4	2,941.4	30.9	61.9
08	783.6	89.8	665.4	246.4	4,195.4	171.3	154.1	2,021.9	2,672.4	2,705.5	30.7	62.3
09	616.3	48.1	310.3	335.7	4,743.2	331.7	246.9	4,997.1	3,455.1	1,359.5	30.3	62.0
10	333.2	1,065.8	453.3	362.1	5,312.8	685.1	203.6	1,407.9	1,590.5	1,998.0	30.0	67.4
Minimum	214.5	43.8	95.8	27.3	363.5	56.6	61.5	174.1	163.8	86.6	11.2	61.2
Average	602.2	498.5	580.6	457.6	2,953.7	368.8	335.7	2,339.0	2,582.5	1,780.5	27.1	62.1
Maximum	1,546.2	1,054.5	1,302.7	922.6	6,397.8	1,007.5	664.3	3,634.7	6,167.6	3,717.6	31.5	64.0

Table of Contents

1. General Items....12
   1.1. Benchmark Sponsor....12
   1.2. Parameter Settings....12
   1.3. Configuration Diagram....13
2. Clause 1 Logical Database Design....15
   2.1. Database Definition Statements....15
   2.2. Physical Organization....15
   2.3. Horizontal Partitioning....15
   2.4. Replication....15
3. Clause 2 Queries and Refresh Functions....16
   3.1. Query Language....16
   3.2. Verifying Method for Random Number Generation....16
   3.3. Generating Values for Substitution Parameters....16
   3.4. Query Text and Output Data from Qualification Database....16
   3.5. Query Substitution Parameters and Seeds Used....16
   3.6. Query Isolation Level....16
   3.7. Source Code of Refresh Functions....17
4. Clause 3 Database System Properties....18
   4.1. ACID Properties....18
   4.2. Atomicity....18
      4.2.1 Completed Transaction....18
      4.2.2 Aborted Transaction....18
   4.3. Consistency....18
      4.3.1 Consistency Test....19
   4.4. Isolation....19
      4.4.1 Read-Write Conflict with Commit....19
      4.4.2 Read-Write Conflict with Rollback....19
      4.4.3 Write-Write Conflict with Commit....19
      4.4.4 Write-Write Conflict with Rollback....20
      4.4.5 Concurrent Progress of Read and Write Transactions....20
      4.4.6 Read-Only Query Conflict with Update Transaction....20
   4.5. Durability....20
      4.5.1 Failure of a Durable Medium....21
      4.5.2 System Crash....21
      4.5.3 Memory Failure....21
5. Clause 4 Scaling and Database Population....22
   5.1. Ending Cardinality of Tables....22
   5.2. Distribution of Tables and Logs Across Media....22
   5.3. Database partition/replication mapping....23
   5.4. RAID Feature....24
   5.5. Modifications to the DBGEN....24
   5.6. Database Load Time....24
   5.7. Data Storage Ratio....24
   5.8. Database Load Mechanism Details and Illustration....25
6. Clause 5 Performance Metrics and Execution Rules....26

6.1. System Activity Between Load and Performance Tests....26
6.2. Steps in the Power Test....26
6.3. Timing Intervals for Each Query and Refresh Functions....26
6.4. Number of Streams for the Throughput Test....26
6.5. Start and End Date/Times for Each Query Stream....26
6.6. Total Elapsed Time of the Measurement Interval....27
6.7. Refresh Function Start Date/Time and Finish Date/Time....27
6.8. Timing Intervals for Each Query and Each Refresh Function for Each Stream....28
6.9. Performance Metrics....28
6.10. The Performance Metric and Numerical Quantities from Both Runs....28
6.11. System Activity Between Performance Tests....28
7. Clause 6 SUT and Driver Implementation....29
7.1. Driver 29
7.2. Implementation-Specific Layer....29
7.3. Profile-Directed Optimization....29
8. Clause 7 Pricing....30
8.1. Hardware and Software Used....30
8.2. Total Three Year Price 30
8.3. Availability Date....30
9. Auditor's Information and Attestation Letter....31
Appendix A. ParAccel and OpenSolaris Parameters....32
Appendix B. Programs and Scripts....32
Appendix C. Query Text and Query Output....58
Appendix D. Seed and Query Substitution Parameters....70
Appendix E. Implementation-Specific Layer/Driver Code....72
Appendix F. Misc database scripts....73
Appendix G. Pricing information....78

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

June 19, 2009

Benchmark Sponsors:

Brad Carlile
Director, Enterprise Benchmarking
Sun Microsystems, Inc.
8305 S. W. Creekside Place
Beaverton, OR 97008

Barry Zane
Chief Technology Officer, ParAccel, Inc.
ParAccel, Inc.
9920 Pacific Heights Blvd, Suite 450
San Diego, CA 92121

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

Platform: **Sun Fire X4540 Server, 43-node cluster**

Database Manager: **ParAccel Analytic Database**

Operating System: **OpenSolaris2009**

The results were:

CPU (Speed)	**Memory**	**Disks**	**QphH@30000GB**
Sun Fire X4540 Server**, 43-node cluster (each node with)**			
2 x AMD Opteron 2356 Quad Core (2.3 GHz)	64 GB Main	48 x 500GB SATA 7.5K RPM int.	1050556.2

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:

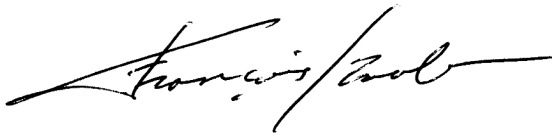
- The database records were defined with the proper layout and size
- The database population was generated using DBGEN
- The database was properly scaled to 30,000GB and populated accordingly
- The compliance of the database auxiliary data structures was verified

- 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
- 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 10 query streams
- The ratio between the longest and the shortest query was such that no query timing was 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

TPC Benchmark H Overview

The TPC BenchmarkTM H (TPC-H) is a Decision Support benchmark. It is a suite of business-oriented ad-hoc queries and concurrent 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
- 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

1. General Items

1.1. Benchmark Sponsor

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

Sun Microsystems, Inc. and ParAccel, Inc. are the sponsors of this TPC-H benchmark.

1.2. Parameter Settings

Settings must be provided for all customer-tunable parameters and options that 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 OpenSolaris and ParAccel Analytic Database™ parameters used in this benchmark.

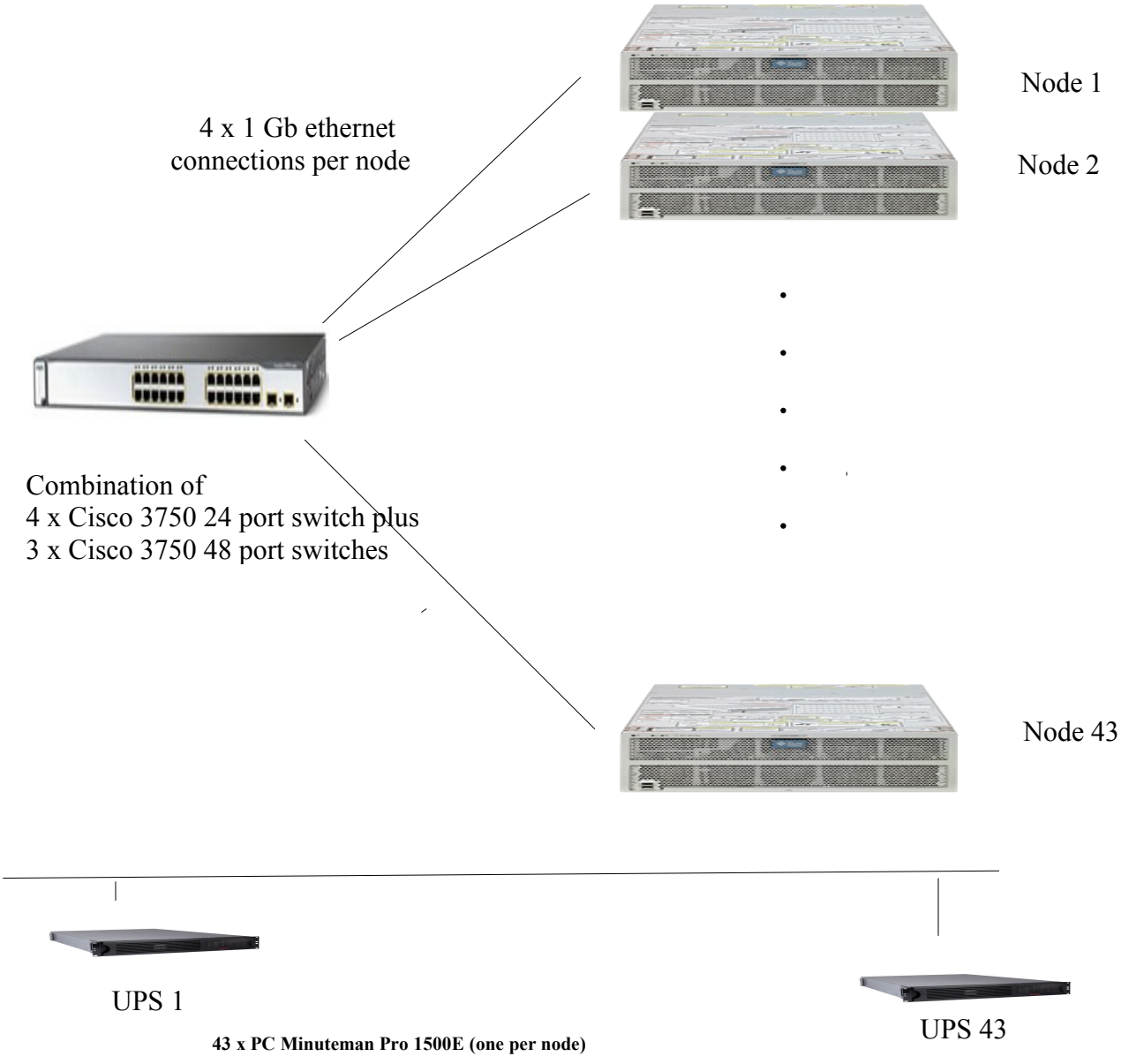
1.3. Configuration Diagram

Provide diagrams of both the measured and priced configurations, accompanied by a description of the differences.

The priced configuration is as follows:

43 SunFire X4540 Servers (each with): **Plus** **4 Cisco 3750 24 Port Switches and 3 Cisco 48 port switches**

- **2 X 2.3 GHz AMD Opteron Processors** **and** **43 Minuteman Pro 1500 UPSs**
- **64 GB Memory**
- **48 X 500 GB Internal disks (1GB = 1,000,000,000 bytes)**



43 x PC Minuteman Pro 1500E (one per node)

The measured configuration is exactly the same as the priced configuration, except that it did not include the 43 UPSs (plus 10% spares).

2. Clause 1 Logical Database Design

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 that create and analyze the tables for the TPC-H 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.

2.3. Horizontal Partitioning

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

Data is horizontally distributed across all nodes based on hashing the primary key (or the first column if no primary key is specified) for each table. No additional data partitioning was used. See Appendix B for details.

2.4. Replication

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

No form of data 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 2.8.0 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.

The supplied QGEN version 2.8.0 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 definitions 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 C contains the query text and query output. The standard queries were used throughout with the following modifications:

- In Q2, Q3, Q10, Q18 and Q21, the "limit" function is used to restrict the number of output rows.
- The semicolon (;) is used as a command delimiter.

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 D 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 isolation level 3.

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).

Appendix B contains the source code for the refresh functions.

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 the ACID tests is included in Appendix B.

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.

4.2.1 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

1. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for a randomly selected order key.
2. The ACID Transaction was performed using the order key from step 1.
3. The ACID Transaction committed.
4. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for the same order key. It was verified that the appropriate rows had been changed.

4.2.2 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.

1. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for a randomly selected order key.
2. The ACID Transaction was performed using the order key from step 1. The transaction was stopped prior to the commit.
3. The ACID Transaction was ROLLED BACK.
4. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for the same order key. It was verified that the appropriate rows had not been changed.

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.

4.3.1 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 by each of eleven 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 the proper order.

4.4.1 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.

1. An ACID Transaction was started for a randomly selected O_KEY, L_KEY, and DELTA. The ACID Transaction was suspended prior to COMMIT.
2. An ACID Query was started for the same O_KEY used in step 1.
3. The ACID Transaction was resumed and COMMITTED.
4. The ACID Query completed. It returned the data as committed by the ACID Transaction.

4.4.2 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.

1. An ACID Transaction was started for a randomly selected O_KEY, L_KEY, and DELTA. The ACID Transaction was suspended prior to ROLLBACK.
2. An ACID Query was started for the same O_KEY used in step 1. The ACID Query did not see the uncommitted changes made by the ACID Transaction.
3. The ACID Transaction was ROLLED BACK.
4. The ACID Query completed.

4.4.3 Write-Write Conflict with Commit

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

1. An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. T1 was suspended prior to COMMIT.
2. Another ACID Transaction, T2, was started using the same O_KEY and L_KEY and a randomly selected DELTA.
3. T2 waited.
4. T1 was allowed to COMMIT and T2 completed.

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

4.4.4 Write-Write Conflict with Rollback

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

1. An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. T1 was suspended prior to COMMIT.
2. Another ACID Transaction, T2, was started using the same O_KEY and L_KEY and a randomly selected DELTA.
3. T2 waited.
4. T1 was allowed to ROLLBACK and T2 completed.
5. It was verified that T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE.

4.4.5 Concurrent Progress of Read and Write Transactions

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

1. An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. T1 was suspended prior to ROLLBACK.
2. Another Transaction, T2, was started which did the following:

   For random values of PS_PARTKEY and PS_SUPPKEY, all columns of the PARTSUPP table for which PS_PARTKEY and PS_SUPPKEY are equal, are returned.
3. T2 completed.
4. T1 was allowed to COMMIT.
5. It was verified that appropriate rows in ORDERS, LINEITEM and HISTORY tables were changed.

4.4.6 Read-Only Query Conflict with Update Transaction

Demonstrate 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.

1. A Transaction, T1, executing iso6-Query against the qualification database, was started using a randomly selected O_KEY.
2. An ACID Transaction T2, was started for a randomly selected O_KEY, L_KEY and DELTA.
3. T2 completed and appropriate rows in the ORDERS, LINEITEM and HISTORY tables had been changed.
4. Transaction T1 completed executing iso6-Query.

4.5. Durability

The SUT 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

4.5.1 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.

All disks containing TPC-H tables are housed on mirrored (RAID1) volumes. A permanent irrecoverable failure of a single durable medium was simulated by removing one side of a mirror from one of the X4540 servers. The system continued to process transactions without any noticeable impact, as is to be expected when only half a mirror is no longer functioning.

4.5.2 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.

There were two independent tests conducted to simulate system crash.

1. Power was cut off to a compute node of the X4540 Servers . When power was restored to that compute node, that node automatically rebooted. The database was manually restarted. The durability success file and the HISTORY table were compared successfully.

2. A system crash was produced by cutting off power to the leader node of the X4540 Servers. When power was restored to the leader node, the leader node automatically rebooted and the database was manually restarted. The durability success file and the HISTORY table were compared successfully.

4.5.3 Memory Failure

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

In the priced configuration, each cluster node was protected by a UPS, which results in limiting the impact of a loss all external power to a single node. The failure was tested as described in 4.5.2. Note, enough UPSs were provided so as to be able to keep all nodes alive after a power failure to the entire cluster.

5. Clause 4 Scaling and Database Population

5.1. Ending Cardinality of Tables

The cardinality (i.e., 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	*Rows*
Lineitem	179999978268
Orders	45000000000
Partsupp	24000000000
Part	6000000000
Customer	4500000000
Supplier	300000000
Nation	25
Region	5

5.2. Distribution of Tables and Logs Across Media

The distribution of tables and logs across all media must be explicitly described.

- All tables were stored on 24 RAID1 volumes per server. The volume was constructed using Solaris Volume Manager.

The following table shows all the disk slices for each disk on each server. All sizes below are pre-RAID1 sizes. Effective RAID1 size is one half the listed number.

Partition	Use	Size
s0	ParAccel Table Space – 24 RAID1 Partitions	50GB
s1	Landing zone for DBGEN – 24 RAID1 Partitions	50GB
s2	Not used (overlay of full disk)	500GB
s3	Swap - RAID0	2GB
s4	2 disks = ZFS for "/" 46 disks = RAID1+0 landing zone	50GB
s7	Solaris Volume Manager Metadata	16MB

5.3. Database partition/replication mapping

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

The rows of each table are horizontally distributed across all nodes based on hashing the primary key for each table. No additional data partitioning was used. In addition, no form of data replication was used.

5.4. RAID Feature

Implementations 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.

RAID 1 was used for all base tables and auxiliary data structures.

5.5. Modifications to the DBGEN

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 2.8.0 was used 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 =3H28M44Sec

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.

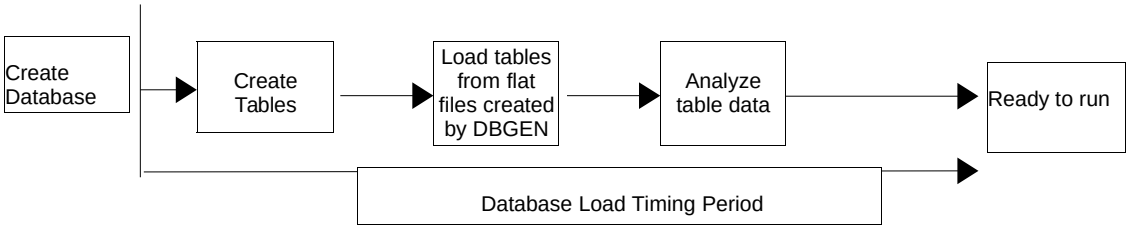
The data storage ratio is computed from the following information:

# of Disks	GB* per disk	Total Disk Space in GB**
2064	500	961124.9
	Total Space	961124.9
	Data Storage Ratio	32.04

** Disk manufacturer definition of one GB is 10^9 bytes*
***In this calculation one GB is defined as 2^30 bytes*

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 test database was loaded using flat files. All load scripts are included in Appendix B. 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 to create and load the data with only the necessary adjustments for size differences.

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.

1. The DBMS on the SUT was restarted between the conclusion of the load test and the beginning of the performance test
2. No other activity on the SUT occurred between the conclusion of the load test and the beginning of the performance test

All scripts and queries used are included in Appendix F

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 and for both refresh functions must be reported for the power test.

The timing intervals for each query and both update functions are reported on the page titled “Numerical Quantities”, contained in the beginning of this document and replicated in the Executive Summary document.

6.4. Number of Streams for the Throughput Test

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

10 query streams and one refresh stream were used for the throughput test.

6.5. Start and End Date/Times for Each Query Stream

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

The start times and finish times for each query stream in the throughput test are reported on the page titled “Numerical Quantities”, contained in the beginning of this document and replicated in the Executive Summary 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 reported on the page titled "Numerical Quantities", contained in the beginning of this document and replicated in the Executive Summary document.

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

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

The start and finish times for each refresh function:

Stream ID	Refresh Function	Start Date	Start Time	End Date	End Time
Stream01	RF1	6/7/2009	1:37:15 PM	6/7/2009	1:37:47 PM
Stream01	RF2	6/7/2009	1:37:47 PM	6/7/2009	1:38:48 PM
Stream02	RF1	6/7/2009	1:38:48 PM	6/7/2009	1:39:19 PM
Stream02	RF2	6/7/2009	1:39:19 PM	6/7/2009	1:40:21 PM
Stream03	RF1	6/7/2009	1:40:21 PM	6/7/2009	1:40:51 PM
Stream03	RF2	6/7/2009	1:40:51 PM	6/7/2009	1:41:53 PM
Stream04	RF1	6/7/2009	1:41:53 PM	6/7/2009	1:42:04 PM
Stream04	RF2	6/7/2009	1:42:04 PM	6/7/2009	1:43:06 PM
Stream05	RF1	6/7/2009	1:43:06 PM	6/7/2009	1:43:38 PM
Stream05	RF2	6/7/2009	1:43:38 PM	6/7/2009	1:44:42 PM
Stream06	RF1	6/7/2009	1:44:42 PM	6/7/2009	1:45:13 PM
Stream06	RF2	6/7/2009	1:45:13 PM	6/7/2009	1:46:14 PM
Stream07	RF1	6/7/2009	1:46:14 PM	6/7/2009	1:46:45 PM
Stream07	RF2	6/7/2009	1:46:45 PM	6/7/2009	1:47:47 PM
Stream08	RF1	6/7/2009	1:47:47 PM	6/7/2009	1:48:18 PM
Stream08	RF2	6/7/2009	1:48:18 PM	6/7/2009	1:49:20 PM
Stream09	RF1	6/7/2009	1:49:20 PM	6/7/2009	1:49:50 PM
Stream09	RF2	6/7/2009	1:49:50 PM	6/7/2009	1:50:52 PM
Stream10	RF1	6/7/2009	1:50:52 PM	6/7/2009	1:51:22 PM
Stream10	RF2	6/7/2009	1:51:22 PM	6/7/2009	1:52:30 PM

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

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

The timing intervals for each query and each refresh function for the throughput test are reported on the page titled "Numerical Quantities", contained in the beginning of this document and replicated in the Executive Summary 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, are reported on the page titled "Numerical Quantities", contained in the beginning of this document and replicated in the Executive Summary 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 three metrics:

Run ID	Qpp	Qth	Qph
Run 1	1,341,952.40	851,735.00	1,069,106.10
Run 2	1,326,910.40	831,758.00	1,050,556.20
% Difference	-1.13%	-2.40%	-1.77%

6.11. System Activity Between Performance Tests

Any activity on the SUT that takes place between the conclusion of Run1 and the beginning of Run2 must be disclosed.

No activity occurred between Run 1 and Run 2.

7. Clause 6 SUT and Driver Implementation

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.

The entire test is run by executing the do30tb.sh shell script.

The text of do30tb.sh is reproduced in Appendix E and the texts of the subscripts invoked by do30tb.sh are reproduced in Appendix B.

7.2. Implementation-Specific Layer

If an implementation-specific layer is used, then a detailed description of how it performs its functions must be supplied, including any related source code or scripts. This description should allow an independent reconstruction of the implementation-specific layer.

All database configuration was done through scripts disclosed in Appendix B.

The performance tests are performed using pSql. It is a ParAccel -provided utility that allows SQL statements to be executed against the ParAccel Analytic Database™. The psql utility is invoked from the command-line on the SUT. It reads input from files containing SQL statements and sends results to stdout. The performance test scripts utilizing psql can be found in Appendix E.

The ACID tests are performed using pSQL. All the ACID test scripts are reproduced in Appendix B.

7.3. Profile-Directed Optimization

If profile-directed optimization as described in Clause 5.2.9 is used, such use must be disclosed.

Profile-directed optimization was not used.

8. Clause 7 Pricing

8.1. Hardware and Software Used

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.

Refer to the Executive Summary for the pricing spreadsheet and Appendix G for the actual price quotes used to create the spreadsheet.

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.

The total 3-year price of the configuration is $3,006,861.06

. For details of pricing, see the second page of the Executive Summary.

Discounts were taken from actual price quotes,available to any buyer with like conditions, provided by Agilysys Inc. and ParAccel Inc. The respective price quotes are included in Appendix G of this document.

8.3. Availability Date

The committed delivery date for general availability of products used in the price calculations must be reported. When the priced system includes products with different availability dates, the reported availability date for the priced system must be the date at which all components are committed to be available.

All hardware and software components used in the measured configuration are generally available as of June 21, 2009.

9. Auditor's Information and Attestation Letter

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.

The auditor's attestation letter follows the table of contents.

Appendix A. ParAccel and OpenSolaris Parameters

This Appendix contains OpenSolaris kernel parameters and environment variables and ParAccel system parameters.

```
================================================
```

ParAccel Server Configuration Parameters
(altered from default)

```
user_queues|1|Query tuning|2|querytuning.html||int|0|
Maximum number of queries that may be in flight at
once--any more wait in line
```

/etc/system settings (on each node)
(altered from default)

```
================================================
set lotsfree = 4096
set bufhwm = 10000

set shmsys:shminfo_shmmax=0xffffffff
set shmsys:shminfo_shmmin=10
set shmsys:shminfo_shmmni=1024
set shmsys:shminfo_shmseg=256
set semsys:seminfo_semmni=4000
set semsys:seminfo_semmsl=300
set semsys:seminfo_semmns=5000
set semsys:seminfo_semopm=150
set rlim_fd_max=1500000
set rlim_fd_cur=1500000
set sata:sata_func_enable=0x5
```

Created rc3.d/S99paraccel (on each node)

```
========================================


udp_recv_hiwat = 4000000
udp_xmit_hiwat = 4000000
udp_do_checksum = 0
eprom  nge_intpt_bind_cpus=0,1,4,5
```

And on the leader node only, added

```
export dn=`ramdiskadm -a rd 128m`
mkfs -F ufs /dev/rramdisk/rd -o
131018,601,1,8192,1024,16,10,120,2048,t,0,-1,8,1,n
mkdir -p /mnt/ramdisk
mount $dn /mnt/ramdisk
chown -R paraccel:paraccel /mnt/ramdisk
```

Appendix B. Programs and Scripts

```
====================================
calculate_load_time.bash
====================================
#!/bin/bash

# calculates load time from the start and end times in
the files
#   start_load.out
#   end_load.out

# called from tpch_report.bash

# final version


if (( $# < 2 ))
then
  echo "usage:  $0  end_load_time  start_load_time"
  echo ""
  echo "           - returns load_time"
  echo "             calls;"
  echo "                sec_from_epoch.bash"
  echo "                my_calc.pl"
  exit
fi


elt=$1     # of the form: yyyy-mm-dd hh:mm:ss
# echo "end_load_time=$elt   [debug
calculate_load_time.bash]"

end_secs_from_epoch=`secs_from_epoch.pl "$elt"`
# echo "end_secs_from_epoch=$end_secs_from_epoch
[debug   calculate_load_time.bash]"

slt=$2     # of the form: yyyy-mm-dd hh:mm:ss
# echo "start_load_time=$slt   [debug
calculate_load_time.bash]"

start_secs_from_epoch=`secs_from_epoch.pl "$slt"`
# echo "start_secs_from_epoch=$start_secs_from_epoch
[debug   tpch_throughput_interval.bash]"

((lt=end_secs_from_epoch-start_secs_from_epoch))

lt_dhms=`s2dhms.bash $lt`

echo $lt_dhms


====================================
clean.sh
====================================
:
# delete output files

rm -f *.out
rm -f *.trunc
rm -f *.ooo
rm -f *.params
rm -f stream*.sql
rm -f *.log
rm -f *.sqlx
rm -f *.params
rm -f update*.sql


====================================
create_tables.sql
====================================
BEGIN;

create table part (
  p_partkey int8 not null encode delta32k distkey
sortkey,
  p_name varchar(55) not null,
  p_mfgr char(25) not null encode bytedict,
  p_brand char(10) not null encode bytedict,
  p_type varchar(25) not null,
  p_size int4 not null encode bytedict,
```

```
  p_container char(10) not null encode bytedict,
  p_retailprice numeric(12,2) not null,
  p_comment varchar(23) not null encode text255,
  PRIMARY KEY (p_partkey)
);

create table region (
  r_regionkey int4 not null distkey sortkey,
  r_name char(25) not null,
  r_comment varchar(152) not null,
  PRIMARY KEY (r_regionkey)
);

create table nation (
  n_nationkey int4 not null distkey sortkey,
  n_name char(25) not null,
  n_regionkey int4 not null,
  n_comment varchar(152) not null,
  PRIMARY KEY (n_nationkey),
  FOREIGN KEY (n_regionkey) REFERENCES region
(r_regionkey)
);

create table supplier (
  s_suppkey int4 not null distkey sortkey,
  s_name char(25) not null,
  s_address varchar(40) not null,
  s_nationkey int4 not null encode always8,
  s_phone char(15) not null,
  s_acctbal numeric(12,2) not null,
  s_comment varchar(101) not null encode text255,
  PRIMARY KEY (s_suppkey),
  FOREIGN KEY (s_nationkey) REFERENCES nation
(n_nationkey)
);


create table partsupp (
  ps_partkey int8 not null encode runlength distkey
sortkey,
  ps_suppkey int4 not null,
  ps_availqty int4 not null always16,
  ps_supplycost numeric(12,2) not null,
  ps_comment varchar(199) not null text255,
  PRIMARY KEY (ps_partkey, ps_suppkey),
  FOREIGN KEY (ps_partkey) REFERENCES part
(p_partkey),
  FOREIGN KEY (ps_suppkey) REFERENCES supplier
(s_suppkey)
);

create table customer (
  c_custkey int8 not null encode delta32k distkey
sortkey,
  c_name varchar(25) not null,
  c_address varchar(40) not null,
  c_nationkey int4 not null encode bytedict,
  c_phone char(15) not null,
  c_acctbal numeric(12,2) not null encode always32,
  c_mktsegment char(10) not null,
  c_comment varchar(117) not null encode text255,
  PRIMARY KEY (c_custkey),
  FOREIGN KEY (c_nationkey) REFERENCES nation
(n_nationkey)
);

create table orders (
  o_orderkey int8 not null encode delta32k distkey
sortkey,
  o_custkey int8 not null,
  o_orderstatus char(1) not null,
  o_totalprice numeric(12,2) not null encode always32,
  o_orderdate date not null encode delta32k,
  o_orderpriority char(15) not null encode bytedict,
  o_clerk char(15) not null,
  o_shippriority int4 not null encode runlength,
  o_comment varchar(79) not null text255,
  PRIMARY KEY (o_orderkey),
  FOREIGN KEY (o_custkey) REFERENCES customer
(c_custkey)
);

create table lineitem (
  l_orderkey int8 not null encode delta32k distkey
sortkey,
  l_partkey int8 not null,
  l_suppkey int4 not null,
  l_linenumber int4 not null encode always8,
  l_quantity numeric(12,2) not null encode bytedict,
  l_extendedprice numeric(12,2) not null encode
always32,
  l_discount numeric(12,2) not null encode always8,
  l_tax numeric(12,2) not null encode always8,
  l_returnflag char(1) not null,
  l_linestatus char(1) not null,
  l_shipdate date not null encode delta,
  l_commitdate date not null encode delta,
  l_receiptdate date not null encode delta,
  l_shipinstruct char(25) not null encode bytedict,
  l_shipmode char(10) not null encode bytedict,
  l_comment varchar(44) not null encode text255,
  FOREIGN KEY (l_orderkey) REFERENCES orders
(o_orderkey),
  FOREIGN KEY (l_partkey,l_suppkey) REFERENCES
partsupp (ps_partkey,ps_suppkey)
);

create table delete_ids (
  d_orderkey int8 not null distkey sortkey
);

COMMIT;
```

```
====================================
dbgen.sh
====================================
amake -o all
cqi allx rm -rf /export/home/paraccel/xen/data/log/\*
cqi allx rm -rf /export/home/paraccel/xen/data/exec/\*
cqi allx rm -rf /export/home/paraccel/xen/data/disks/\
*/tpch/\*
cqi allx rm -rf /export/home/paraccel/xen/data/disks/\
*/dev
cd
cqi setworking
cp $HOME/xen/rel/etc/xenpostgresql.conf $HOME/xen/rel/
etc/xenpostgresql.conf.real
cp $HOME/xen/rel/etc/xenpostgresql.conf.dbgen
$HOME/xen/rel/etc/xenpostgresql.conf
sh initdb.sh
cp $HOME/xen/rel/etc/xenpostgresql.conf.real
$HOME/xen/rel/etc/xenpostgresql.conf
cd $HOME/run/scripts
psql dev -c "create database tpch"
psql tpch -f create_tables.sql
psql tpch -c "xpx 'dbgen 30000'"
cd /tpch_data
rm *
cp $HOME/xen/rel/share/tpc/dists.dss .
$HOME/xen/rel/share/tpc/dbgen -v -C 30 -U 22 -s 30000
cd $HOME/run/scripts
sh localize_updates.sh
cd
sh initdb.sh


====================================
dbtables-xen.sql
====================================
\d lineitem;
\d region;
\d nation;
\d customer;
\d part;
```

```
\d supplier;
\d partsupp;
\d orders;


==================================
end_throughput_interval.bash
==================================
#!/bin/bash

if (( $# < 1 ))
then
  echo
  echo "usage:  $0  num_streams  "
  echo
  echo "           - returns ending time for the last
throughput stream to finish"
  echo
  exit
fi

ns=$1

# set string to "stream1.out stream2.out ...
stream$ns.out"

for i in stream[1-9]*.out;   # gets every streamN.out
except stream0.out
do
    num_plus_out=${i#stream}
    num=${num_plus_out%.out}
    #echo $num
    if ((num<=ns))
    then
         string=$string" "$i
    fi
done

# create a file of completion times of the query
streams
grep 'Stream completion' $string  | cut -d'|' -f 2 |
sed "s/^\s//" > /tmp/completion_times

# append the largest completion time of the refresh
streams to the file
max_pre_refresh=$(grep STOP update_throughput.out |
\
                                    cut -d'|' -f 2 |
sed "s/^\s//")
max_refresh=${max_pre_refresh%.*}
echo $max_refresh >> /tmp/completion_times

# return the latest time
sort /tmp/completion_times > /tmp/ascending_times

max=`tail -1 /tmp/ascending_times`     # get the
highest time
echo $max



==================================
gen_streams.sh
==================================
#$SHELL
echo " Gen_Streams got called"

if (( $# < 2 ))
then
     echo "usage:  $0   seed1 seed2  scale_factor
num_streams"
     exit
fi

DSS_QUERY=$HOME/xen/rel/share/tpc
SCRIPTS_DIR=$HOME/run/scripts

seed1=$1
seed2=$2
sf=$3
ns=$4

i=0

# cd $DSS_QUERY
cd $SCRIPTS_DIR

while ((i<=ns))
do

   #Create params file for stream $i
   echo "Creating param files for stream $i"
   echo
   qn=1
   $DSS_QUERY/qgen -p $i -l
$SCRIPTS_DIR/stream$i.params -r $seed1$seed2 -s $sf
> /dev/null
   echo "Params files created"
   echo

   $DSS_QUERY/qgen -a -p $i -r $seed1$seed2 -i
$SCRIPTS_DIR/begin.sql -t $SCRIPTS_DIR/end.sql -s
$sf \
   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
21 22 > $SCRIPTS_DIR/stream${i}.sql
   ((seed2=seed2+1))

if (($i == 0))
then
      echo "Stream $i (for power test) created."
fi
if (($i > 0))
then
      echo "Stream $i created."
fi
((i=i+1))
done

((seed2=seed2-1))
echo $seed1$seed2


==================================
get_cardinal_info.sh
==================================
#!/usr/bin/bash
date > cardinal_info.out
echo "Cardinal info for the current data set" >>
cardinal_info.out
echo "------------------" >> cardinal_info.out
for table in lineitem customer orders part partsupp
supplier nation region; do echo $table count: >>
cardinal_info.out; echo "-----------" >>
cardinal_info.out; psql tpch -c "select count(*) from
$table" >> cardinal_info.out; echo "-----------" >>
cardinal_info.out;done


==================================
go
==================================
y
y
y
y
y


==================================
localize_updates.sh
==================================
psql tpch -c "xpx 'localize delete1
/tpch_data/delete.1'"
```

```
psql tpch -c "xpx 'localize delete2
/tpch_data/delete.2'"
psql tpch -c "xpx 'localize delete3
/tpch_data/delete.3'"
psql tpch -c "xpx 'localize delete4
/tpch_data/delete.4'"
psql tpch -c "xpx 'localize delete5
/tpch_data/delete.5'"
psql tpch -c "xpx 'localize delete6
/tpch_data/delete.6'"
psql tpch -c "xpx 'localize delete7
/tpch_data/delete.7'"
psql tpch -c "xpx 'localize delete8
/tpch_data/delete.8'"
psql tpch -c "xpx 'localize delete9
/tpch_data/delete.9'"
psql tpch -c "xpx 'localize delete10
/tpch_data/delete.10'"
psql tpch -c "xpx 'localize delete11
/tpch_data/delete.11'"
psql tpch -c "xpx 'localize delete12
/tpch_data/delete.12'"
psql tpch -c "xpx 'localize delete13
/tpch_data/delete.13'"
psql tpch -c "xpx 'localize delete14
/tpch_data/delete.14'"
psql tpch -c "xpx 'localize delete15
/tpch_data/delete.15'"
psql tpch -c "xpx 'localize delete16
/tpch_data/delete.16'"
psql tpch -c "xpx 'localize delete17
/tpch_data/delete.17'"
psql tpch -c "xpx 'localize delete18
/tpch_data/delete.18'"
psql tpch -c "xpx 'localize delete19
/tpch_data/delete.19'"
psql tpch -c "xpx 'localize delete20
/tpch_data/delete.20'"
psql tpch -c "xpx 'localize delete21
/tpch_data/delete.21'"
psql tpch -c "xpx 'localize delete22
/tpch_data/delete.22'"
psql tpch -c "xpx 'localize lineitem_u1
/tpch_data/lineitem.tbl.u1'"
psql tpch -c "xpx 'localize lineitem_u2
/tpch_data/lineitem.tbl.u2'"
psql tpch -c "xpx 'localize lineitem_u3
/tpch_data/lineitem.tbl.u3'"
psql tpch -c "xpx 'localize lineitem_u4
/tpch_data/lineitem.tbl.u4'"
psql tpch -c "xpx 'localize lineitem_u5
/tpch_data/lineitem.tbl.u5'"
psql tpch -c "xpx 'localize lineitem_u6
/tpch_data/lineitem.tbl.u6'"
psql tpch -c "xpx 'localize lineitem_u7
/tpch_data/lineitem.tbl.u7'"
psql tpch -c "xpx 'localize lineitem_u8
/tpch_data/lineitem.tbl.u8'"
psql tpch -c "xpx 'localize lineitem_u9
/tpch_data/lineitem.tbl.u9'"
psql tpch -c "xpx 'localize lineitem_u10
/tpch_data/lineitem.tbl.u10'"
psql tpch -c "xpx 'localize lineitem_u11
/tpch_data/lineitem.tbl.u11'"
psql tpch -c "xpx 'localize lineitem_u12
/tpch_data/lineitem.tbl.u12'"
psql tpch -c "xpx 'localize lineitem_u13
/tpch_data/lineitem.tbl.u13'"
psql tpch -c "xpx 'localize lineitem_u14
/tpch_data/lineitem.tbl.u14'"
psql tpch -c "xpx 'localize lineitem_u15
/tpch_data/lineitem.tbl.u15'"
psql tpch -c "xpx 'localize lineitem_u16
/tpch_data/lineitem.tbl.u16'"
psql tpch -c "xpx 'localize lineitem_u17
/tpch_data/lineitem.tbl.u17'"
psql tpch -c "xpx 'localize lineitem_u18
/tpch_data/lineitem.tbl.u18'"
psql tpch -c "xpx 'localize lineitem_u19
/tpch_data/lineitem.tbl.u19'"
psql tpch -c "xpx 'localize lineitem_u20
/tpch_data/lineitem.tbl.u20'"
psql tpch -c "xpx 'localize lineitem_u21
/tpch_data/lineitem.tbl.u21'"
psql tpch -c "xpx 'localize lineitem_u22
/tpch_data/lineitem.tbl.u22'"
psql tpch -c "xpx 'localize orders_u1
/tpch_data/orders.tbl.u1'"
psql tpch -c "xpx 'localize orders_u2
/tpch_data/orders.tbl.u2'"
psql tpch -c "xpx 'localize orders_u3
/tpch_data/orders.tbl.u3'"
psql tpch -c "xpx 'localize orders_u4
/tpch_data/orders.tbl.u4'"
psql tpch -c "xpx 'localize orders_u5
/tpch_data/orders.tbl.u5'"
psql tpch -c "xpx 'localize orders_u6
/tpch_data/orders.tbl.u6'"
psql tpch -c "xpx 'localize orders_u7
/tpch_data/orders.tbl.u7'"
psql tpch -c "xpx 'localize orders_u8
/tpch_data/orders.tbl.u8'"
psql tpch -c "xpx 'localize orders_u9
/tpch_data/orders.tbl.u9'"
psql tpch -c "xpx 'localize orders_u10
/tpch_data/orders.tbl.u10'"
psql tpch -c "xpx 'localize orders_u11
/tpch_data/orders.tbl.u11'"
psql tpch -c "xpx 'localize orders_u12
/tpch_data/orders.tbl.u12'"
psql tpch -c "xpx 'localize orders_u13
/tpch_data/orders.tbl.u13'"
psql tpch -c "xpx 'localize orders_u14
/tpch_data/orders.tbl.u14'"
psql tpch -c "xpx 'localize orders_u15
/tpch_data/orders.tbl.u15'"
psql tpch -c "xpx 'localize orders_u16
/tpch_data/orders.tbl.u16'"
psql tpch -c "xpx 'localize orders_u17
/tpch_data/orders.tbl.u17'"
psql tpch -c "xpx 'localize orders_u18
/tpch_data/orders.tbl.u18'"
psql tpch -c "xpx 'localize orders_u19
/tpch_data/orders.tbl.u19'"
psql tpch -c "xpx 'localize orders_u20
/tpch_data/orders.tbl.u20'"
psql tpch -c "xpx 'localize orders_u21
/tpch_data/orders.tbl.u21'"
psql tpch -c "xpx 'localize orders_u22
/tpch_data/orders.tbl.u22'"


====================================
looper
====================================
#! /bin/csh

set i = 0
while 1
    echo "Running [$i] $* ..."
    $*

    @ i = ($i + 1)
end


====================================
my_calc.pl
====================================
#!/usr/bin/perl

# -w removed from above to surpress runtime warnings
on garbadge expressions i.e.
#                       function keys, escape
sequence, etc. as well a certain
#                       string expressions eval
```

```
doesn't understand

# final version:  dec 21 2002

# evaluates all kinds of expressions, except as noted
below
#
#   we can do everything except exp! when exp contains
parentheses
#   but  factorial(exp) works for arbitrary exp
#
#    todo: maybe lift this restriction



our $ARGC = @ARGV ;

my $show_commas;      # initially turned off
my $show_trace;       # initially turned off

if ( $ARGC >= 1 )   # these are the batch cases
  {
     if ($ARGC == 2)
       {
         $show_commas = $ARGV[1];
       }
     else
       {
         $show_commas = '';
       }

     if ($ARGV[0] eq "help")
       {
          print "\nusage:  $0 \n";
          print "              interactive case \n";
          print "   or\n\n";
          print "usage:  $0  help\n";
          print "              displays usage info
(i.e. this output)\n";
          print "   or\n\n";
          print "usage:  $0  expression  [any 2nd arg
means commify]\n";
          print "              - displays evaluated
expression and exits\n";
          print "              - place expression in
quotes if it contains spaces\n\n";
          exit;
       }

     # $ARGV[0] is the first actual argument, not the
script name as in C or the shell

     my $result = evaluate_perl($ARGV[0]);

     # print $ARGV[0], " = ", $result, "\n";
     # just show the result in the batch case, so it
can be used in a pipeline

     print  $result, "\n";

     exit;
  }

# interactive case
#    need to commify this

my $input;

my $mode = 'perl';

for (;;)
  {
    print "enter arithmetic or string expression
(q=exit): ";
    $input = <STDIN>;
    chomp ($input);
    if ( $input =~ /quit|exit|^q$|^e$/i )
      {
        exit;
      }
    elsif ($input eq 'help')
      {
        print_help();
      }
    else  # evaluate expression
      {
        if ( $input =~ /commas\s*$|c\s*$/i )  # turn
show_commas on
          {
            $show_commas = 'yes';
            $input =~ s/\s*commas\s*$|\s*c\s*$//i;  #
erase "c" or "commas" and
                                                    #
surrounding whitespace
          }

        if ($input eq '')  # do next iteration if the
rest of $input is empty
          {
            next;
          }

        my $result = evaluate_perl($input);
        print $input, " = ", $result, "\n";
      }
  }



sub evaluate_perl
{
   $_ = shift;  # $_ is a string, or arithmetic
expression

#print "----- $_ -----\n";

   # note: n! is not understood by Perl. We use
regular expressions to
   #       translate many cases of exp! to
factorial(exp).
   #       We can thus handle n!, (n-r)! and  ( (n+r)!
+ m! ), but not ((n)!)!
   #       or ((n+m)!)!

   # replace n! with factorial(n);
   s/
      (\d+)!            # match and capture integer!
   /factorial($1)/gx;

   # replace (exp)! with factorial(exp),  where exp is
like "num1 arith_op num2"
   s/
      (\([\d\+\-\*\/\s]+\))!   # match and capture
(integer arith_operator integer)!
   /factorial($1)/x;           # replace $1 with
factorial($1)

   s/ln/log/g;      # allow the use ln for the natural
log

# questions: why does eval return NULL on "(2 *
(3+2))!" instead of returning
#            itself as a string?

#print "----- $_ -----\n";

   my $result = eval;

   if ($show_commas and length($result) > 4)   # put
commas in $result
     {
        $result = commify($result);
     }

   $show_commas = '';   # turn off commas
```

```
   return $result;  # note: $result is local to this
sunroutine
}


sub factorial
{
#print "in factorial\n";
   my $n = shift;
   if ($n <= 0)
     {
        return 1;
     }
   my $ans = 1;
   for (my $i = 1; $i <= $n; $i++)
     {
       $ans = $ans*$i;
     }
   return $ans;
}


#binomial coefficient   c(n,r)
sub c
{
#print "in c\n";
   my $n = shift;
   my $r = shift;
   if ($r > $n or $r < 0)
     {
       return undefined;
     }
   else
     {
       return factorial($n) / ( factorial($r) *
factorial($n-$r) );
     }
}


# base 2 log
sub lg
{
my $n = shift;
return log($n)/log(2);   #  log = base e log
}

# base 10 log
sub Log
{
my $n = shift;
return log($n)/log(10);   #  log = base e log
}

sub commify
{
my $input = shift;

$input = reverse $input;
$input =~ s/(\d\d\d)(?=\d)(?!\d*\.)/$1,/g;
$input = reverse $input;
return $input;
}


sub print_help
{
print <<DONE

------------------------------------------------------
----------------------------
interactively evaluate perl expressions

   quit, exit, q or e will all exit

arithemetic expressions can contain
   perl's usual arithmetic operators:   +  -  *  /  %
and  **

   perl's builtin arithmetic functions:  exp()  sin()
 cos()  log()  abs()  and  sqrt()
   as well as the following 9 functions:
      the factorial functions:  !, factorial
      the log functions:  lg (base 2), Log (base 10)
and ln (base e -- synonym for log)
      the binomial cooefficient function:  c(n,i)

notes:
  1. We don't completely parse expressions, so that
some expressions cannot be
     corectly evaluated: e.g.

         (exp)!, when exp is an expression that
contains parenthesis,
         however factorial(exp) works for all
arithmetic expressions.

  2. Both arithmetic and numeric string results are
displayed with separating commas,
     when either "c" or "commas" is appended to any
input expression (with or without
     separating whitespace).
     "c" or "commas" should not be appended to string
expressions as meaningless output
     may be produced.

  3. Given any arithmetic or string function f(), this
script will evaluate expressions
     containing f whenever a subroutine, returning the
value of f() is included in the
     script. This is the way factorial(n) and c(n,m)
are implemented.
------------------------------------------------------
----------------------------

DONE
}



==========
my_transform.pl
==========
#!/usr/bin/perl -w

# displays the result of applying the substitution
operation of the 2nd
# argument on the 1st argument


our $ARGC = @ARGV ;

if ( $ARGC < 2)
  {
     print "\n",'usage:  $0  "string"
"substitution_exp"',"\n\n";
     print "              where substitution_exp is
of the form          \n";
     print "
s/regex1/regex2/modifier                  \n";
     print "               place quotes around the two
arguments          \n";
     print "                 - double quotes always
work for the 2nd arg  \n";
     print "                 - double quotes usually
work for the 1st arg \n";
     print "                   (unless it contains a \
$ sign -- in which  \n";
     print "                   case single quotes must
be used and the    \n";
     print "                   \$ sign must be
escaped                 \n\n";
     exit;
  }

$_ = $ARGV[0];          # string
```

```
$sub = $ARGV[1];        # substition expression

# construct the following:
$_=string;substition_expression;$_
# and then pass it to eval

$exp='$_="'.$ARGV[0].'"'.";$sub;".'$_';  #
#print "$exp    [debug]\n";

$n=eval $exp;

print "$n\n";
```

```
===================================
refresh1.sql
===================================
select 'Stream 0 Query RF1 begin - ',
LOCALTIMESTAMP(3);
select 'Stream 0 Query RF1 START - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimestamp;

begin;
copy orders from 'local' with parallel subdir
'orders_u1' delimiter '|';
copy lineitem from 'local' with parallel subdir
'lineitem_u1' delimiter '|';
commit;



select 'Stream 0 Query RF1 STOP - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimestamp;


===================================
refresh2.sql
===================================
select 'Stream 0 Query RF2 START - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimestamp;

begin;
copy delete_ids from 'local' with parallel subdir
'delete1' delimiter '|';
delete from lineitem where l_orderkey in (select
d_orderkey from delete_ids);
delete from orders where o_orderkey in (select
d_orderkey from delete_ids);
truncate table delete_ids;
commit;
select 'Stream 0 Query RF2 STOP - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimestamp;
select 'Stream 0 Query RF2 complete - ',
LOCALTIMESTAMP(3);


===================================
round_to_tenths.pl
===================================
#!/usr/bin/perl -w

# used for tpch; spec requires most numbers to be
rounded to the nearest tenth

our $ARGC = @ARGV ;

if ( $ARGC < 1)
  {
     print "\nusage:  $0  number  \n\n";
     print "               displays number rounded to
the nearest tenth \n\n";
     exit
  }

$_=$ARGV[0];
$_ =~ s/,//g;
printf "%.1f\n", $_;
```

```
===================================
run_2_refresh1.sql
===================================
select 'Stream 0 Query RF1 begin - ',
LOCALTIMESTAMP(3);
select 'Stream 0 Query RF1 START - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimestamp;

begin;
copy orders from 'local' with parallel subdir
'orders_u12' delimiter '|';
copy lineitem from 'local' with parallel subdir
'lineitem_u12' delimiter '|';
commit;

select 'Stream 0 Query RF1 STOP - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimestamp;


===================================
run_2_refresh2.sql
===================================
select 'Stream 0 Query RF2 START - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimestamp;

begin;
copy delete_ids from 'local' with parallel subdir
'delete12' delimiter '|';
delete from lineitem where l_orderkey in (select
d_orderkey from delete_ids);
delete from orders where o_orderkey in (select
d_orderkey from delete_ids);
truncate table delete_ids;
commit;

select 'Stream 0 Query RF2 STOP - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimestamp;
select 'Stream 0 Query RF2 complete - ',
LOCALTIMESTAMP(3);


===================================
s2dhms.bash
===================================
#!/bin/bash


if (( $# < 1 ))
then
  echo ""
  echo "usage:  $0  time_in_secs"
  echo ""
  echo '           - converts "secs" to
"days:hrs:min:secs'
  echo " "
  exit
fi


ts=$1  # time in secs

td=`my_calc.pl "int ($ts/86400)"`
remsecs=`my_calc.pl $ts%86400`

th=`my_calc.pl "int $remsecs/3600"`
remsecs=`my_calc.pl $remsecs%3600`

tm=`my_calc.pl "int ($remsecs/60)"`
remsecs=`my_calc.pl "$remsecs%60"`

ts=$remsecs


if [[ $td > 1 ]]
then
  echo -e "${td}days\c"
```

```
  colon=:
elif [[ $td = 1 ]]
then
  echo -e "${td}day\c"
  colon=:
else
  colon=
fi

if [[ $th > 1 ]]
then
  echo -e "$colon${th}hrs\c"
  colon=:
elif [[ $th = 1 ]]
then
  echo -e "$colon${th}hr\c"
  colon=:
else
  colon=
fi

if [[ $tm > 1 ]]
then
  echo -e "$colon${tm}mins\c"
  colon=:
elif [[ $tm = 1 ]]
then
  echo -e "$colon${tm}min\c"
  colon=:
else
  colon=
fi

if [[ $ts > 1 ]]
then
  echo -e "$colon${ts}secs"
elif [[ $lts = 1 ]]
then
  echo -e "$colon${ts}sec"
else
  echo ""    # writes out newline
fi
```

```
====================================
secs_from_epoch.pl
====================================
#!/usr/bin/perl -w

# computes the number of seconds from epoch for a
given timestamp in the form
#    yyyy-mm-dd hh:mm:ss
#
# the SybaseIQ TPC-H kit uses this timestamp format

use Time::Local;

our $ARGC = @ARGV ;

if ( $ARGC < 1 )
  {
    print "\nusage:  $0  time_stamp \n";
    print " \n";
    print "            time_stamp is in the form:
yyyy-mm-dd hh:mm:ss  \n";
    print "                 put time_stamp in single or
double quotes  \n\n";
    exit;
  }

# $ARGV[0] is the first actual argument, not the
script name as in C or the shell
$_=$ARGV[0];

my $arg_length = length($_);
if ($arg_length == 0)
{
   print "Input invalid\n";
   exit;
}

# print "$_     [debug1  secs_from_epoch.pl]\n";
s/-/:/g;
s/ /:/g;
# print "$_     [debug2  secs_from_epoch.pl]\n";

@fields=split(/:/,$_);   # separates $_ using : as
delimiter

# print "0 = $fields[0]   [debug   secs_from_epoch.pl\
n";
# print "1 = $fields[1]   [debug   secs_from_epoch.pl\
n";
# print "2 = $fields[2]   [debug   secs_from_epoch.pl\
n";
# print "3 = $fields[3]   [debug   secs_from_epoch.pl\
n";
# print "4 = $fields[4]   [debug   secs_from_epoch.pl\
n";
# print "5 = $fields[5]   [debug   secs_from_epoch.pl\
n";

my $yr   = $fields[0]-1900;    # epoch begins at 1900
my $mon  = $fields[1]-1;       # months go from 0-11
my $day  = $fields[2];         # days go from 1-31
my $hr   = $fields[3];         # hours go from 0-23
my $min  = $fields[4];         # mins  go from 0-59
my $sec  = $fields[5];         # mins  go from 0-59

# print "$yr,$mon,$day,$hr,$min,$sec   [debug3
secs_from_epoch.pl\n";
$time = timelocal($sec,$min,$hr,$day,$mon,$yr);
print "$time\n";
```

```
====================================
show_machine
====================================
#!/bin/bash

# March 2006: used psrinfo and prtconf, when prtdiag
is not properly implemented
# April 2006: added online versus total processors

# shows hostname, ipaddress, num processors, processor
type and memory size
# for Solaris, Linux and CYGWIN systems

# won't run on Solaris unless picld is running


os=$(uname)
host=$(hostname)

if [[ $os = SunOS ]]
then

   hipa=$(arp `uname -n` | cut -d ' ' -f1,2)    #
hostname (ip_address)

   op=`psrinfo | grep on-line | wc -l`     # processor
name begins with US
   tp=`psrinfo | wc -l`

   ((online_processors=op))
   ((total_processors=tp))

   speed=$(psrinfo -v | sort | uniq | grep MHz | tr -s
' ' | cut -d ' ' -f 7,8 | tr -d ',')

   Ultra=no
   prtdiag > /dev/null 2>&1
   rc=$?
```

```
   if (( rc==0 ))    # prtdiag displays processor type

   then
     nup=`prtdiag | grep Ultra | wc -l`    # processor
name begins with Ultra
     if ((nup>0))
     then
        Ultra=yes
     fi

      if [[ $Ultra = no ]]   # look for US
      then
          for i in $(prtdiag | grep US | head -1)
          do
              if [[ $i = US* ]]
              then
                 ptype=$i
              fi
          done
      else
          for i in $(prtdiag | grep Ultra | head -1 |
tr ','   ' ')
          do
              if [[ $i = Ultra* ]]
              then
                 ptype=$i
              fi
          done
      fi
   else           # get generic processor type from
psrinfo
      ptype=$(psrinfo -v | egrep "sparc|386" | head -1
| awk '{ print $2 }')
   fi


   speed_ptype="$speed $ptype"

   mem=$(prtconf | grep 'Mem' | cut -d' ' -f 3,4)


elif [[ $os = Linux ]]
then

   hipa=$(arp `uname -n` | cut -d ' ' -f1,2)

   online_processors=$(cat /proc/cpuinfo | grep 'model
name' | wc -l)
   total_processors=$online_processors

   st=$(cat /proc/cpuinfo | grep 'model name' | uniq |
cut -d ':' -f 2)
   st2=${st//(R)/}      # remove all occurances of (R)
in $st
   st3=${st2//(TM)/}    # remove all occurances of
(TM) in $st2
   st4=${st3# Intel }   # remove " Intel " from
beginning of $st3
   speed_ptype=${st4/ CPU /, }    # change " CPU " in
$st4 to ", "


   #if [[ $speed_ptype = *Xeon* || $speed_ptype =
XEON* ]] # if Xeon or XEON divide by 2
   #then
                            # since all Xeons are
hyperthreaded
   #
((online_processors=online_processors/2))
   # and hyperthreaded processors are

#fi
 # regarded as 2 processors

   m1=$(cat /proc/meminfo | grep MemTotal|tr -s ' ' |
cut -d' ' -f 2)
   ((m2=m1/1024))
   mem="$m2 Megabytes"

elif [[ $os = CYGWIN* ]]
then

   ipa=$(ipconfig | grep 'IP Address' | grep -v Auto |
\
       my_transform.pl "232T . . . . . :
123.34.56.43" "s/\. //g" | cut -d ' ' -f3)

   hipa="$host ($ipa)"

   online_processors=$(cat /proc/cpuinfo | grep 'cpu
count' | wc -l)    # change linux to this
   total_processors=$online_processors

   st=$(cat /proc/cpuinfo | grep 'model name' | uniq |
tr -s ' ' | cut -d ' ' -f 4,5,6,7,8,9)
   st2=${st//(R)/}      # remove all occurances of (R)
in $st
   st3=${st2#Intel }    # remove "Intel " from
beginning of $st
   speed_ptype=${st3/ CPU /,   }    # change " CPU   "
in $st3 to ", "

   m1=$(cat /proc/meminfo | grep MemTotal|tr -s ' ' |
cut -d' ' -f 2)
   ((m2=m1/1024))
   mem="$m2 Megabytes"

else
   echo $host is a $os system, no further information
at this time
   exit
fi

result="$hipa:   $online_processors X $speed_ptype,
$mem"
((res_len=${#result}))

echo " "
((i=0))    # print a line consisting of res_len dashes
echo "  "
while ((i<res_len + 15))
do
   printf "%s" "-"
   ((i=i+1))
done
# now print a newline
echo "  "

#printf "%s" 1 $result
echo "   $hipa:   $online_processors(of
$total_processors) X $speed_ptype,   $mem"

((i=0))    # print a line consisting of res_len dashes
while ((i<res_len + 15))
do
   printf "%s" "-"
   ((i=i+1))
done
# now print a newline
echo " "
#end with a blank line
echo " "
```

```
==================================
show_refresh_start_stop_times
==================================
#!/bin/bash

# displays list of refresh start and stop times from
timestamps in update_throughput.out
# this works as long as no refresh takes more than 24
hours
```

```
if (( $# < 1 ))
then
     echo
     echo  "usage:  num_streams"
     exit
     echo
     exit
fi

ns=$1   # num streams

start_date=$(egrep "updates b" update_throughput.out
|
             cut -d '|' -f2 | sed "s/\
(..\:..\:..\....\)//")

end_date=$(egrep "updates c" update_throughput.out |
             cut -d '|' -f2 | sed "s/\
(..\:..\:..\....\)//")

echo "Refresh Streams - Start and End Times"
echo "-------------------------------------"
echo

echo "Stream ID   Refresh ID  Duration"
echo "-----------------------------------------------"
((i=1))
while ((i<=ns))
do
   for j in 1 2
   do
     start_time=$(egrep "START" update_throughput.out
|
     grep RF$j |
     grep "Stream $i" |
     cut -d '|' -f2 |
     sed "s/ //g")
     end_time=$(egrep "STOP" update_throughput.out |
     grep RF$j |
     grep "Stream $i" |
     cut -d '|' -f2 |
     sed "s/ //g")
     calc_args="$end_time-$start_time"
     elapsed_time=$(my_calc.pl $calc_args)
     elapsed_time=$(round_to_tenths.pl $elapsed_time)
     echo "stream0$i   RF$j         $elapsed_time
secs"
   done
((i=i+1))
done
echo "-----------------------------------------------"
last_date=$start_date

echo
```

```
==================================
start_throughput_interval.bash
==================================
#!/bin/bash

if (( $# < 1 ))
then
  echo
  echo "usage:  $0  num_streams  "
  echo
  echo "           - returns starting time of the
first started throughput stream"
  echo
  exit
fi

ns=$1

for i in stream[1-9]*.out;   # gets every streamN.out
except stream0.out
do
    num_plus_out=${i#stream}
    num=${num_plus_out%.out}
    #echo $num
    if ((num<=ns))
    then
         string=$string" "$i
    fi
done
# $string is "stream1.out stream2.out ...
stream$ns.out"


grep 'Stream begin' $string          | cut -d'|' -f 2
| sed "s/^ //" | \

                                                sort > /
tmp/ascnd_times

pre_max=`head -1 /tmp/ascnd_times`     # get the
highest time

pre_max=${pre_max%%.*}   # remove the trailing .ddd
from $pre_max
max=${pre_max#\'}     # remove the leading ' from
$pre_max

echo $max
```

```
==================================
storage_ratio_new.bash
==================================
#!/bin/bash

if (( $# < 2 ))
then
  echo "usage:  $0   scale_factor  num_disks
disk_size(GB)   "
  echo " "
  echo "       - assumes all disks are the same size"
  echo " "
  exit
fi

sf=$1
ds=$2
nd=$3

((ts=nd*ds))

ratio=`my_calc.pl "($ts*1000000000)/(${sf}*1024**3)"`

echo $ratio
```

```
==================================
time_wait.sql
==================================
create or replace function sleep (integer) returns
time as '
declare
  seconds alias for $1;
  later time;
  thetime time;
begin
  thetime := timeofday()::timestamp;
  later := thetime + (seconds::text || ''
seconds'')::interval;
  loop
  if thetime >= later then
    exit;
  else
    thetime := timeofday()::timestamp;
  end if;
end loop;
return later;
end;
```

```
' language plpgsql;


====================================
timing_intervals.bash
====================================
#!/bin/bash

#   to do:  RF1 and RF2

if (( $# < 1 ))
then
  echo "usage:  $0  num_streams  "
  echo ""
  echo "           - shows timing intervals as
required by the FDR   "
  exit
fi

ns=$1

for qnum in 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
18 19 20 21 22
do
   echo " "
   echo "Query $qnum Response Times"
   echo "-----------------------"

   grep LENGTH stream[0-5].out|grep "Query $qnum "|
grep -v cast|cut -d' ' -f1,2,3,4,7,8| \
                                                cut -d' '
-f5 | tr  "'" " "|tr , " " >/tmp/full

   echo  "stream 0  `head -1 /tmp/full`"
   echo "-------------------------"

   # we will now discard the first (stream0) row of
/tmp/full
   # and print the remaining rows together with the
min, avg and max
   tail -$ns /tmp/full  >/tmp/qrt

   sum=
   ((k=1))
   for j in `cat /tmp/qrt`
   do
      echo  "stream $k   $j"
      sum="$sum+$j"
      ((k=k+1))
   done

   sort -n /tmp/qrt >  /tmp/sorted
   echo "-----------------------"
   echo "minimum  `head -1   /tmp/sorted`"
   echo "-----------------------"

   s="($sum)/$ns"
   #echo s=$s   [trace throughput_response_times.bash]
   avg=`my_calc.pl "$s"`
   echo "average    $avg"
   echo "-----------------------"

   echo "maximum  `tail -1   /tmp/sorted`"
   echo "-----------------------"
   echo " "
   echo " "

done



====================================
tpch.sh
====================================
#$SHELL

source $HOME/.bashrc

# Set env var's needed for ntest
export xenv="xen-TPCH9-1"
nq=22

cur_dir=$(pwd)
if [[ $cur_dir != $HOME/run/scripts ]]
then
      echo
      echo ERROR: do_test must be run from
$HOME/run/scripts
      echo
      echo "       the current dir is $cur_dir "
      echo
      exit
fi

audit_file_dir=`cat /tmp/this_tpch`/

if (( $# < 1 ))
then
   echo
   echo "usage: $0    scope   (plus additional args
depending upon scope)                                 "
   echo
   echo "   scope
values:
                      "
   echo
   echo "   - load            create and load a tpch
database from scratch                  "
   echo "   - qi        do query i
(i=1,2,...,22)
 "
   echo "   - stream0   do all 22 queries without any
refreshes                        "
   echo "   - refresh1  do refresh
1                                    "
   echo "   - refresh2  do refresh 2"
   echo "   - thrustreams     do a single thrurefresh
run without any refreshes              "
   echo "   - thrurefresh     do all the throughput
refreshes without query streams"
   echo "   - all       do full audit run: 2 power-
thrurefresh cycles:"
   echo "   run1 <onerefresh 1 + stream0 + onerefresh
2 + thrustreans + thrurefresh> + run2<>"
   echo
   exit
fi

scope=$1

case $scope in
          load)  if (( $# < 3  || $# > 3 ))
                 then
                       echo
                       echo "usage: $0    load
scale_factor    num_query_streams"
                       echo
                       exit
                 fi
             seed=1
             sf=$2
             nqs=$3;;


            q*)  if (( $# < 2 || $# > 2 ))
                 then
                       echo
                       echo "usage: $0    qi
scale_factor                                "
                       echo
                       echo "     note: database must
be running; if not restart it first"
                       echo
                       exit
                 fi
```

```
                query_num=${scope:1}
                case $query_num in
                   [1-9])   ;;  # q1  to q9
                  1[0-9])   ;;  # q10 to q19
                  2[0-2])   ;;  # q20 to q22
                       *)  echo
                           echo "ERROR: query number
($query_num) must be between 1 and 22"
                           echo
                           exit
                esac

                sf=$2  ;;

       stream0)  if (( $# < 3 || $# > 3 ))
                 then
                       echo
                       echo "usage: $0    stream0
scale_factor  input_seed"
                       echo
                       exit
                 fi

                 sf=$2
                 let input_seed=$3   ;;

       refresh1)  if (( $# < 2 || $# > 2 ))
                 then
                       echo
                       echo "usage: $0    refresh1
scale_factor  "
                       echo
                       exit
                 fi

                 sf=$2 ;;

       refresh2)  if (( $# < 2 || $# > 2 ))
                 then
                       echo
                       echo "usage: $0    refresh2
scale_factor  "
                       echo
                       exit
                 fi

                 sf=$2 ;;


    thrustreams)  if (( $# < 4 || $# > 4 ))
                 then
                       echo
                       echo "usage: $0
thrustreams    scale_factor  "
                       echo
"                              num_query_streams
input_seed "
                       echo
                       exit
                 fi

                 sf=$2
                 nqs=$3
                 let input_seed=$4   ;;

    thrurefresh)  if (( $# < 4 || $# > 4 ))
                 then
                       echo
                       echo "usage: $0
thrurefresh    scale_factor  "
                       echo
"                                    num_query_streams
input_seed "
                       echo
                       exit
                 fi

                 sf=$2
                 nqs=$3
                 let input_seed=$4   ;;



           all)  if (( $# < 6 || $# > 6 ))
                 then
                       echo
                       echo "usage: $0    all
scale_factor  "
                       echo "
num_query_streams                          "
                       echo "
num_disks  disk_size(in GB)  system_cost    "
                       echo
                       exit
                       echo
                 fi

                 sf=$2
                 nqs=$3
                 let nd=$4
                 let ds=$5
                 let sc=$6
                 input_seed=$(grep "as a seed"
stream0.sql | cut -d 'g' -f2 | cut -d 'a' -f1) ;;
             *)  echo
                 echo "ERROR: scope (=$scope) must be
one of:                    "
                 echo
                 echo "
load,q1,q2,...,q22,stream0,refresh1,refresh2,"
                 echo "
thrustreams,thrurefresh,lthrurefresh,all     "
                 echo
                 echo
                 exit
esac

# we impose the minimum stream requirement only in the
"all" case
# for all other scopes the min number of query streams
is 1
# aminqs is thus the "allowed minimum" for the current
run

aminqs=1
#
# For testing purposes, we aren't enforcing this,
though.
#
# if [[ ($scope = all ) && -z $real_scope && -z
$real_scope2 ]]
# then
#    case $sf in
#          .1)  aminqs=1
#               ;;
#           1)  aminqs=2
#               ;;
#          10)  aminqs=3
#               ;;
#          30)  aminqs=4
#               ;;
#         100)  aminqs=5
#               ;;
#         300)  aminqs=6
#               ;;
#        1000)  aminqs=7
#               ;;
#        3000)  aminqs=8
#               ;;
#       10000)  aminqs=9
#               ;;
#       30000)  aminqs=10
#               ;;
#          * )
#               echo
#               echo "ERROR:  scale factor (=$sf) must
```

```
be one of (.1,1,10,30,100,300,1000,3000,10000,30000)"
#               echo
#               exit
#               echo;;
#    esac
# fi

# These are the minimum number of streams for a
compliant run

case $sf in
       1)  cmin=2
           ;;
      10)  cmin=3
           ;;
      30)  cmin=4
           ;;
     100)  cmin=5
           ;;
     300)  cmin=6
           ;;
    1000)  cmin=7
           ;;
    3000)  cmin=8
           ;;
   10000)  cmin=9
           ;;
   30000)  cmin=10
           ;;
       *)  cmin=1
           ;;
esac


PLATFORM=`uname -m`

# make sure input_seed value is legal (i.e. >= 0 ) for
the scopes where it is been
# supplied as a commandline arg ("lrun1" and
"lthrurefresh" are covered by "all"

if [[ $scope = stream0    || $scope = power || $scope
= thrustreams ||  \
      $scope = thrurefresh || $scope = run1  || $scope
= all           ]]
then
      if [[ $input_seed -lt 0 ]]
      then
           echo
           echo "ERROR: input_seed($input_seed) must
be >= 0"
           echo
           exit
      fi
fi


# check that the scale factor is legal where it has
been supplied as
# a command line arg

      if [[ $sf != .1 && $sf != 1 && $sf != 10 &&
$sf != 30 && \
            $sf != 100 && $sf != 300 && $sf != 1000 &&
$sf != 3000 && $sf != 10000 && $sf != 30000 ]]
      then
           echo
           echo "ERROR: scale factor (=$sf) must be
one of [.1,1,10,30,100,300,1000,3000,10000,30000]"
           echo
           exit
      fi


if [[ $scope = thrurefresh || $scope = all  ]]
then
   # generate update_throughputX.sql
   i=2
   run=1
   q="'"
   ((max_streams=nqs + 1))
   while((run<=2))
   do
      upd_tput_fname="update_throughput_run_${run}_$
{nqs}.sqlx"
      cat update_throughput_header > $upd_tput_fname
      stream_number=1
      while ((i<=max_streams))
      do
         echo "select 'Stream ${stream_number} RF1
START - ', date_part('epoch',LOCALTIMESTAMP(3)),
localtimestamp;" >> $upd_tput_fname
       echo "begin;" >> $upd_tput_fname
       echo "copy orders from 'local' with parallel
subdir ${q}orders_u${i}${q} delimiter '|';" >>
$upd_tput_fname
       echo "copy lineitem from 'local' with parallel
subdir ${q}lineitem_u${i}${q} delimiter '|';" >>
$upd_tput_fname
       echo "commit;" >> $upd_tput_fname
         echo "select 'Stream ${stream_number} RF1
STOP - ', date_part('epoch',LOCALTIMESTAMP(3)),
localtimestamp;" >> $upd_tput_fname

         echo "select 'Stream ${stream_number} RF2
START - ', date_part('epoch',LOCALTIMESTAMP(3)),
localtimestamp;" >> $upd_tput_fname
       echo "begin;" >> $upd_tput_fname
       echo "copy delete_ids from 'local' with
parallel subdir ${q}delete${i}${q} delimiter '|';" >>
$upd_tput_fname
       echo "delete from lineitem where l_orderkey in
(select d_orderkey from delete_ids);" >>
$upd_tput_fname
       echo "delete from orders where o_orderkey in
(select d_orderkey from delete_ids);" >>
$upd_tput_fname
       echo "truncate table delete_ids;" >>
$upd_tput_fname
       echo "commit;" >> $upd_tput_fname
         echo "select 'Stream ${stream_number} RF2
STOP - ', date_part('epoch',LOCALTIMESTAMP(3)),
localtimestamp;" >> $upd_tput_fname
        ((i=i+1))
         ((stream_number=stream_number+1))
      done
      cat update_throughput_footer >> $upd_tput_fname
      ((i=nqs + 2))
      ((max_streams=max_streams + nqs))
      ((run=run + 1))
   done
fi

if [[ $scope = thrurefresh || $scope = run1 || $scope
= all || $scope = mall ]]
then
      run=1
      while((run<=2))
      do
         if [[ -f update_throughput_run_${run}_$
{nqs}.sqlx ]]
         then
              echo
         else
              echo
              echo "ERROR: update_throughput_run_$
{run}_${nqs}.sqlx must exist for a run of $nqs
streams"
              echo
              exit
      fi
      ((run=run + 1))
      done
fi

# make sure that the number of query streams >=
```

```
minimum allowed num streams
# aminqs has been previously set to 1 except when
$scope is "all" in which
# case aminqs is the complient minimun

if [[ $scope = all ]]
then
      if ((nqs < aminqs))    # nqs: requested number
of query streams
      then
            echo
            echo "ERROR: requested query streams
(=$nqs) must be >= $aminqs for sf=$sf & scope=$scope"
            echo
            exit
      fi
fi

# we have checked the commandline args for
reasonableness, now
# describe the scope of the run and its configuration

echo " "
case $scope in
           q* )
                 echo "DOING QUERY $query_num  with: "
;;
          all )
                 if [[ $real_scope = lrun1 ]]
                 then
                       echo "DOING LOAD, FOLLOWED BY
POWER, FOLLOWED BY THROUGHPUT with:"
                 else
                       echo "DOING FULL AUDIT TEST
(load, plus 2 full runs) with: "
                 fi ;;
         load )
                 echo "ONLY DOING LOAD (and create
database) with: " ;;
      refresh1 )
                 echo "ONLY DOING ONE REFRESH 1 with:
" ;;
      refresh2 )
                 echo "ONLY DOING ONE REFRESH 2 with:
" ;;
   thrustreams )
                 echo "ONLY DOING ONE THROUGHPUT TEST
WITHOUT REFRESHES with:" ;;
   thrurefresh )
                 echo "ONLY DOING ONE THROUGHPUT TEST
with:" ;;
      stream0 )
                 echo "ONLY DOING 22 SINGLE-USER
QUERIES (NO REFRESHES):" ;;
            * )
                 echo "ERROR:  scope (=$scope) must be
one of"
                 echo "
(all,load,power,refresh,stream0,thrurefresh,run1)"
                 echo
                 exit ;;
esac

echo " "



      echo "      scale factor = $sf "

if [[ $scope = all || $scope = thrurefresh || $scope =
thrustreams ]]
then
   echo "      num thrurefresh streams = $nqs
(compliant minimum for this scale factor: $cmin)"
fi


if [[ $scope != load && $scope != refresh1 && $scope !
= refresh2 ]]
then
   if [[ $input_seed == 1 ]]
   then
         seed=$input_seed
         echo "      using newly gererated seed"
   elif  [[ $input_seed == 0 ]]
   then
         if [[ -f stream0.sql ]]
         then
               existing_seed=$(grep "as a seed"
stream0.sql | cut -d 'g' -f2 | cut -d 'a' -f1)
               seed=$existing_seed
               echo  "      using existing seed =
$seed"
         else
               echo "ERROR: Cannot use existing seed
when stream0.sql does not exist"
               exit
         fi
   else
      echo "      using $input_seed as seed"
      seed=$input_seed
   fi
fi


echo "      Using the following nodes"
echo "      --------------------------"
cqi show | tee node_config.out

# echo -e "Are these OK? (type y or n) \c"
# read ans
#
#
# if [[ $ans = n || $ans = no ]]
# then
#      echo
#      echo change one or more of
#      echo "  options.sql, tpch.cfg,
create_database.sql, create_tables or /etc/system"
#      echo and try again
#      echo
#      exit
# fi

# -------------  start real execution
-----------------------------------

power=0           # used in run1 to determine if
Composite should be calculated
throughput=0      # this might not be necessary

if [[ $scope = q* ]]
then
     # assumes database is running
     one_query $query_num $sf
     echo
     exit
fi

echo
echo

if [[ $scope = load  ]]
then

     echo " "
     echo " "
     echo " "

     echo "Creating tpch database: `date`..."
#    psql dev -c "drop database tpch"
     $HOME/xen/rel/bin/createdb tpch
     echo "  "

     echo "Installing sleep stored procedure"
     psql -d tpch -f time_wait.sql
```

```
     echo "  "

     echo "Performing Pre-Test clean-up."
     sh clean.sh
     echo "  "

     echo "Creating TPCH schema"
     psql -d tpch -f create_tables.sql
     echo "  "

     #
     # Load the database
     #

     starttime=`date "+%Y-%m-%d %H:%M:%S"`
     echo "    Load started  $starttime " >
start_load.out
     echo "Load region `date`"
     psql tpch -c "copy region from 'copied' with
parallel delimiter '|'"
     echo "Load nation `date`"
     psql tpch -c "copy nation from 'copied' with
parallel delimiter '|'"
     echo "Load customer `date`"
     psql tpch -c "copy customer from 'copied' with
parallel delimiter '|'"
     echo "Load lineitem `date`"
     psql tpch -c "copy lineitem from 'copied' with
parallel delimiter '|'"
     echo "Load orders `date`"
     psql tpch -c "copy orders from 'copied' with
parallel delimiter '|'"
     echo "Load partsupp `date`"
     psql tpch -c "copy partsupp from 'copied' with
parallel delimiter '|'"
     echo "Load part `date`"
     psql tpch -c "copy part from 'copied' with
parallel delimiter '|'"
     echo "Load supplier `date`"
     psql tpch -c "copy supplier from 'copied' with
parallel delimiter '|'"
     echo "Defrag storage `date`"
     psql tpch -c "xpx 'defrag'"
     echo "Build statistics `date`"
     psql -d tpch -a -c 'analyze' >> analyze.out
     psql -d tpch -a -c 'set random_stat_sample to on;
analyze partsupp(ps_partkey,ps_suppkey); set
random_stat_sample to off;'
     echo "Load completed `date`"
     endyear=`date '+%Y'`
     enddate=`date '+%m%d'`
     endtime=`date '+%H%M%S'`
     endstamp=`date "+%Y-%m-%d %H:%M:%S"`
     echo "    Load Finished  $endstamp" >
end_load.out
     echo "    Seed time is:
$endyeaar$enddate$endtime"
     echo
     slt=`tr -s ' ' <start_load.out | cut -d' ' -f 4,5
`
     elt=`tr -s ' ' <end_load.out | cut -d' ' -f 4,5 `
     lt=`calculate_load_time.bash "$elt" "$slt" `  #
calls secs_from_epoch.pl
     echo "Database Load Time:  $lt"
     echo " "
     if ((seed==1))
     then
           echo "Generating $((nqs+1)) Query Streams "
           echo "Using the appropriate timestamp seed
= $enddate$endtime"
           gen_streams.sh $enddate $endtime $sf $nqs
     else
        if [[ $input_seed == 0 ]]
        then
           echo  "Using existing seed = $seed
####################"
        else
           echo  "Using provided seed = $seed
####################"
           rm -f stream*.sql
           gen_streams.sh 0 $seed $sf $nqs
        fi
     fi
     echo
     if [[ $scope = load ]]
     then
          echo exiting $0
          echo
          exit
     fi
fi

#Load completed

if [[ $scope = all ]]
then
     echo "Starting Audit Verification Scripts
`date` "
     echo

     # After the load Completes run the Audit SQL

     psql -d tpch -f dbtables-xen.sql >
rdbtablest_start.out

     echo
     echo "Finished Audit Start Verification Scripts
`date` "
     echo
fi


if [[ ( $scope = refresh1 || $scope = refresh2 ||
$scope = stream0 || $scope = all ) ]]
then
     if [[  $scope = all ]]
     then
#          psql tpch -c "xpx 'optimize_reads y'"
>/dev/null
          echo
          echo "Start Run 1 Power Test   `date` "
          echo "----------------------          "
     elif [[ $scope = stream0 ]]
     then
          echo
          echo "Start Stream0   `date` "
          echo "-------------           "
     else
          echo
          echo "Start Refresh   `date` "
          echo "-------------           "
     fi


     if [[ $scope = refresh1 || $scope = all ]]
     then
          echo
          echo "     Start refresh 1   `date`  "
          psql -d tpch -f refresh1.sql > rf1.out
          echo
          echo "     End refresh 1   `date`  "
        #create the refresh file
        cat rf1.out  > update_power.out
     fi


     if [[ $scope = stream0  || $scope = all ]]
     then
          echo
          echo "     Start Query Stream 0   `date` "
          psql -d tpch -f stream0.sql > stream0.out
          cat s0q01-1.out s0q01-4.out s0q02-1.out
s0q02-4.out s0q03-1.out s0q03-4.out s0q04-1.out s0q04-
4.out s0q05-1.out s0q05-4.out s0q06-1.out s0q06-4.out
s0q07-1.out s0q07-4.out s0q08-1.out s0q08-4.out s0q09-
1.out s0q09-4.out s0q10-1.out s0q10-4.out s0q11-1.out
```

```
s0q11-4.out s0q12-1.out s0q12-4.out s0q13-1.out s0q13-
4.out s0q14-1.out s0q14-4.out s0q15-1.out s0q15-4.out
s0q16-1.out s0q16-4.out s0q17-1.out s0q17-4.out s0q18-
1.out s0q18-4.out s0q19-1.out s0q19-4.out s0q20-1.out
s0q20-4.out s0q21-1.out s0q21-4.out s0q22-1.out s0q22-
4.out >> stream0.out
            echo
            echo "     Finish Query Stream 0  `date` "
      fi


      if [[ $scope = refresh2 || $scope = all ]]  # do
refresh2
      then
            echo
            echo "     Start refresh 2   `date` "
          psql -d tpch -f refresh2.sql > rf2.out
            echo
            echo "     End refresh 2    `date`  "
            # append the refresh file
            cat rf2.out >> update_power.out
      else
            echo
            echo "End Stream0   `date`"
            echo "-----------         "
            echo will now make report
            echo
      fi


      if [[ $scope = refresh2 ]]
      then
            echo
            echo "     End Refresh 2 `date`     "
            echo "--------------------------"
            echo
            tpch_stats.pl power $sf $nqs none
            echo
            if [[ $scope = refresh2 ]]
            then
            exit   # on refresh
            fi
      else
            echo
            echo "End Run 1 Power Test     `date` "
            echo "---------------------          "
            echo "   Computing response times.."
            echo " "
            echo " "
            tpch_stats.pl power $sf "1" "none"
            power=$(tpch_stats.pl stream0 $sf 1 power)
            echo " "
            echo " "
            echo "   Done computing response times"
            echo " "
            echo "       Power = $power"
            echo
            echo
            ps -eaf | grep asiq| grep -v grep | tr -s '
' | cut -d ' ' -f8,9
            echo
      fi
fi

if [[ ($scope = thrurefresh || $scope = all || $scope
= thrustreams) ]]
then
      echo
      echo "Start Run 1 Throughput `date`  "
      echo "-----------------------         "
      echo "  "
      echo "     Start Query Streams  `date` "
      echo "  "

      ((i=1))
      while ((i<=nqs))  # run all query streams
concurrently
      do
            echo "        Starting stream ${i}"
            psql -d tpch -f stream${i}.sql > stream$
{i}.out &
       echo "        Stream ${i} started   `date`"
            ((i=i+1))
      done

      wait   # for everything

      ((i=1))
      while ((i<=nqs))
      do
       cat s${i}q01-1.out s${i}q01-4.out s${i}q02-1.out
s${i}q02-4.out s${i}q03-1.out s${i}q03-4.out s${i}q04-
1.out s${i}q04-4.out s${i}q05-1.out s${i}q05-4.out s$
{i}q06-1.out s${i}q06-4.out s${i}q07-1.out s${i}q07-
4.out s${i}q08-1.out s${i}q08-4.out s${i}q09-1.out s$
{i}q09-4.out s${i}q10-1.out s${i}q10-4.out s${i}q11-
1.out s${i}q11-4.out s${i}q12-1.out s${i}q12-4.out s$
{i}q13-1.out s${i}q13-4.out s${i}q14-1.out s${i}q14-
4.out s${i}q15-1.out s${i}q15-4.out s${i}q16-1.out s$
{i}q16-4.out s${i}q17-1.out s${i}q17-4.out s${i}q18-
1.out s${i}q18-4.out s${i}q19-1.out s${i}q19-4.out s$
{i}q20-1.out s${i}q20-4.out s${i}q21-1.out s${i}q21-
4.out s${i}q22-1.out s${i}q22-4.out >> stream${i}.out
            ((i=i+1))
      done

      if [[ $scope != thrustreams ]]  # thrustreams
does a thrurefresh without refreshes
      then
            echo "     Start Refresh Stream for set
$nqs  `date` "
            psql -d tpch -f update_throughput_run_1_$
{nqs}.sqlx >> update_throughput.out
      fi

      echo " "
      echo "     All Query and Refresh Streams have
completed `date` "
      echo " "


      throughput_interval=$
(tpch_throughput_interval.bash  $nqs)
      tpch_throughput_interval.bash $nqs >
throughput_interval.out
      throughput=$(tpch_stats.pl "thrurefresh"  $sf
$nqs "throughput")
      echo
      echo "     Throughput Interval =
$throughput_interval secs "
      echo
      echo "     Throughput = $throughput"
      echo
      if [[ $power != 0 ]]
      then
            composite=$(my_calc.pl
"sqrt($power*$throughput)")
            echo "     Composite = $composite"
            echo
      fi
      echo
      #ps -eaf | grep asiq | grep -v grep | tr -s ' ' |
cut -d ' ' -f8,9
      echo
fi

if [[ $scope = stream0 || $scope = power || $scope =
thrustreams ||
       $scope = thrurefresh || $scope = all  ]]
then
      dayHr=`date '+%m%d%H'`
      echo
      echo
      if [[ -z $real_scope2 ]]     # non_null means
lthrurefresh
      then
```

```
        if [[ -z $real_scope ]]   # non_null means
lrun1
        then
           if [[ $scope = stream0 ]]
           then
              composite=0
           fi
           if [[ $scope = power ]]
           then
              composite=$power
           fi
           if [[ $scope = thrurefresh || $scope =
all ]]
           then
              composite=$throughput
           fi
           rpt_file_name="run1.log"
         #rpt_file_name="mrun1_${scope}_${sf}g_$
{nqs}s_${dayHr}_${xenv}_1n_$(round_to_tenths.pl $
{composite}).r"
           echo "Producing ${rpt_file_name}"
           echo "Running tpch_report
##########################################"
           tpch_report.new $scope
$sf                                \
                           run1_${scope}_${sf}g_$
{nqs}s_${dayHr}_${xenv}_1n.r \
                           $seed $nqs $nd $ds
$sc                                 \
                        > ${rpt_file_name}
           mv ${rpt_file_name} $audit_file_dir
        else
        echo" "
        fi
     else
        # lthrurefresh
        composite=$throughput
        rpt_file_name="run1.log"
        echo "Producing ${rpt_file_name}"
        tpch_report.new $scope
$sf                                    \
                         run1_${scope}_${sf}g_$
{nqs}s_${dayHr}_${xenv}_1n.r \
                         $seed $nqs $nd $ds
$sc                                        \
                        > ${rpt_file_name}
           mv ${rpt_file_name} $audit_file_dir
     fi
     echo
fi


if [[ $scope = all ]]
then
     echo
     ((i=0))
     while ((i<=nqs))  # move the streamX.out files to
audit naming standard
     do
        ((q=1))
        while ((q<=22))
        do
           if [ $q -lt 10 ]
           then
              qn="0$q"
           else
              qn="$q"
           fi
           cat s${i}q${qn}-?.out > $
{audit_file_dir}m1s${i}q${qn}.out
           lc=`wc -l < s${i}q${qn}-3.out`
           if [ $lc -gt 200 ]
           then
              head -102 s${i}q${qn}-3.out >  s${i}q$
{qn}-3.trunc
              echo
"********************************************" >>
s${i}q${qn}-3.trunc
              tail -102 s${i}q${qn}-3.out >> s${i}q$
{qn}-3.trunc
              cat s${i}q${qn}-1.out  s${i}q${qn}-2.out
s${i}q${qn}-3.trunc s${i}q${qn}-4.out > $
{audit_file_dir}m1s${i}q${qn}.trunc.out
           fi
        ((q=q+1))
        done
        ((i=i+1))
     done

     mv update_power.out $audit_file_dir"m1s00rf.out"
#change to move
     mv update_throughput.out
$audit_file_dir"m1s01rf.out" #change to move

     echo "Moved  *.out files to " $audit_file_dir
     echo
     echo
     echo "FINISHED Run1      `date`"
     echo
fi
####################
# Run 2
####################
if [[ ( $scope = refresh1 || $scope = refresh2 ||
$scope = stream0 || $scope = all ) ]]
then
     if [[  $scope = all ]]
     then
#       echo
#       echo "Restart Database   `date` "
#       cqi xstop
#       sleep 20
#       cqi xstart
          echo
          echo "Start Run 2 Power Test   `date` "
          echo "----------------------          "
     elif [[ $scope = stream0 ]]
     then
          echo
          echo "Start Stream0   `date` "
          echo "-------------           "
     else
          echo
          echo "Start Refresh   `date` "
          echo "-------------           "
     fi


     if [[ $scope = refresh1 || $scope = all ]]
     then
          echo
          echo "     Start refresh 1   `date`  "
          psql -d tpch -f run_2_refresh1.sql > rf1.out
          echo
          echo "     End refresh 1   `date`  "
        #create the refresh file
        cat rf1.out  > update_power.out
     fi


     if [[ $scope = stream0  || $scope = all ]]
     then
           echo
           echo "     Start Query Stream 0   `date` "
           psql -d tpch -f stream0.sql > stream0.out
           cat s0q01-1.out s0q01-4.out s0q02-1.out
s0q02-4.out s0q03-1.out s0q03-4.out s0q04-1.out s0q04-
4.out s0q05-1.out s0q05-4.out s0q06-1.out s0q06-4.out
s0q07-1.out s0q07-4.out s0q08-1.out s0q08-4.out s0q09-
1.out s0q09-4.out s0q10-1.out s0q10-4.out s0q11-1.out
s0q11-4.out s0q12-1.out s0q12-4.out s0q13-1.out s0q13-
4.out s0q14-1.out s0q14-4.out s0q15-1.out s0q15-4.out
s0q16-1.out s0q16-4.out s0q17-1.out s0q17-4.out s0q18-
1.out s0q18-4.out s0q19-1.out s0q19-4.out s0q20-1.out
s0q20-4.out s0q21-1.out s0q21-4.out s0q22-1.out s0q22-
4.out >> stream0.out
```

```
          echo
          echo "     Finish Query Stream 0  `date` "
     fi


     if [[ $scope = refresh2 || $scope = all ]]  # do
refresh2
     then
          echo
          echo "     Start refresh 2   `date` "
         psql -d tpch -f run_2_refresh2.sql > rf2.out
          echo
          echo "     End refresh 2    `date`  "
          # append the refresh file
          cat rf2.out >> update_power.out
     else
          echo
          echo "End Stream0   `date`"
          echo "-----------         "
          echo will now make report
          echo
     fi


     if [[ $scope = refresh2 ]]
     then
          echo
          echo "     End Refresh 2 `date`    "
          echo "-------------------------"
          echo
          tpch_stats.pl power $sf $nqs none
          echo
          if [[ $scope = refresh2 ]]
          then
          exit   # on refresh
          fi
     else
          echo
          echo "End Run 2 Power Test     `date` "
          echo "---------------------          "
          echo "   Computing response times.."
          echo " "
          echo " "
          tpch_stats.pl power $sf "1" "none"
          power=$(tpch_stats.pl stream0 $sf 1 power)
          echo " "
          echo " "
          echo "   Done computing response times"
          echo " "
          echo "       Power = $power"
          echo
          echo
          ps -eaf | grep asiq| grep -v grep | tr -s '
' | cut -d ' ' -f8,9
          echo
     fi
fi

if [[ ($scope = thrurefresh || $scope = all || $scope
= thrustreams) ]]
then
     echo
     echo "Start Run 2 Throughput `date`  "
     echo "-----------------------         "
     echo "  "
     echo "     Start Query Streams  `date` "
     echo "  "

     ((i=1))
     while ((i<=nqs))  # run all query streams
concurrently
     do
        echo "        Starting stream ${i}"
        psql -d tpch -f stream${i}.sql > stream$
{i}.out &
        echo "        Stream ${i} started `date`"
        ((i=i+1))
     done

     wait   # for everything

     ((i=1))
     while ((i<=nqs))  # get timings for each stream
     do
      cat s${i}q01-1.out s${i}q01-4.out s${i}q02-1.out
s${i}q02-4.out s${i}q03-1.out s${i}q03-4.out s${i}q04-
1.out s${i}q04-4.out s${i}q05-1.out s${i}q05-4.out s$
{i}q06-1.out s${i}q06-4.out s${i}q07-1.out s${i}q07-
4.out s${i}q08-1.out s${i}q08-4.out s${i}q09-1.out s$
{i}q09-4.out s${i}q10-1.out s${i}q10-4.out s${i}q11-
1.out s${i}q11-4.out s${i}q12-1.out s${i}q12-4.out s$
{i}q13-1.out s${i}q13-4.out s${i}q14-1.out s${i}q14-
4.out s${i}q15-1.out s${i}q15-4.out s${i}q16-1.out s$
{i}q16-4.out s${i}q17-1.out s${i}q17-4.out s${i}q18-
1.out s${i}q18-4.out s${i}q19-1.out s${i}q19-4.out s$
{i}q20-1.out s${i}q20-4.out s${i}q21-1.out s${i}q21-
4.out s${i}q22-1.out s${i}q22-4.out >> stream${i}.out
        ((i=i+1))
     done

     if [[ $scope != thrustreams ]]  # thrustreams
does a thrurefresh without refreshes
     then
          echo "     Start Refresh Stream for set
$nqs  `date` "
          psql -d tpch -f update_throughput_run_2_$
{nqs}.sqlx >> update_throughput.out
     fi

     echo " "
     echo "     All Query and Refresh Streams have
completed `date` "
     echo " "

     throughput_interval=$
(tpch_throughput_interval.bash  $nqs)
     tpch_throughput_interval.bash $nqs >
throughput_interval.out
     throughput=$(tpch_stats.pl "thrurefresh"  $sf
$nqs "throughput")
     echo
     echo "     Throughput Interval =
$throughput_interval secs "
     echo
     echo "     Throughput = $throughput"
     echo
     if [[ $power
```

==========

ACID Test Execution Code

```
====================================
tpch_acid_wrapper.sh
====================================
#$SHELL
sh clean.sh
source $HOME/.bashrc
echo "-----------------------------------------"
echo " Shutting down the database. "
echo "-----------------------------------------"
#cqi xstop
#sleep 5
echo "-----------------------------------------"
echo " Initializing the database. "
echo "-----------------------------------------"
#cqi initdb -f
#sleep 5
echo "-----------------------------------------"
echo " Starting the database. "
echo "-----------------------------------------"
#cqi xstart
#sleep 20
```

```
echo "----------------------------------------"
echo " Loading the 1GB TPC-H data set. "
echo "----------------------------------------"
cd tpch1g/
./load_tpch.sh
echo "----------------------------------------"
echo " Starting the acid tests. "
echo "----------------------------------------"
# go back to the home directory for the acid tests
cd ..
echo "----------------------------------------"
echo " Creating the history table. "
echo "----------------------------------------"
psql tpch -f create_history.ddl >> create_history.out
echo "----------------------------------------"
echo " Running the verification script. "
echo "----------------------------------------"
psql tpch -f dbtables.sql >> dbtables.out
echo "Qualificaion complete"
echo
cd tpch1g/
run_tpch.sh
cd ..
echo "----------------------------------------"
echo " Creating the seed file with three records "
echo "----------------------------------------"
acid_setup.sh 3 >> acid_setup.out
echo "----------------------------------------"
echo " Running the atomicity test "
echo "----------------------------------------"
acid_atomicity.sh >> acid_atomicity.out
echo "----------------------------------------"
echo " Running the consistency test "
echo "----------------------------------------"
acid_consistency_main_v2.sh >>
acid_consistency_main.out
echo "----------------------------------------"
echo " Running the isolation group "
echo "----------------------------------------"
for arg in 1 2 3 4 5 6; do acid_isolation_main$arg.sh
>> acid_isolation_main$arg.out ;done
echo "----------------------------------------"
echo " Running durability "
echo "----------------------------------------"
acid_durability_main.sh > acid_durability_main.out &
check_progress.sh
```

```
==================================
acid_atomicity.sh
==================================
#$SHELL
#
# TPC-H ACID Atomicity Test
#
# Copyright 2007 ParAccel, Inc
#

echo Atomicity Test -- Started at: `date`
echo

#
######################################################
####
# Run ROLLBACK case

# Clean history table
echo Atomicity Test -- Clearing History Table
psql -q -t tpch <<EOF11
     delete from HISTORY;
     \q
EOF11
echo

# Testing ROLLBACK case
echo Atomicity Test -- Starting ROLLBACK loop at:
`date`
echo
i="0"
while read O_KEY L_KEY DELTA
do
if [ ${L_KEY}x != "x" ]; then
  echo
-------------------------------------------------
-------
  echo Atom - Rollback -- For O_KEY = $O_KEY L_KEY =
$L_KEY DELTA = $DELTA
  echo Atom - Rollback -- Before Values
  i=$[$i+1]
  psql -q -t tpch <<EOF12
     select
            L_EXTENDEDPRICE as L_EXTENDED,
            L_QUANTITY as L_QUANTITY,
            L_DISCOUNT as L_DISCOUNT,
            L_TAX as L_TAX,
            O_TOTALPRICE as O_TOTAL
       from ORDERS,
            LINEITEM
      where O_ORDERKEY = $O_KEY
        and L_ORDERKEY = O_ORDERKEY
        and L_LINENUMBER = $L_KEY;
     \q
EOF12

./acid_trans.sh $O_KEY $L_KEY $DELTA ROLLBACK

  echo
  echo Atom - Rollback -- After Values
  psql -q -t tpch <<EOF13
     select
            L_EXTENDEDPRICE as L_EXTENDED,
            L_QUANTITY as L_QUANTITY,
            L_DISCOUNT as L_DISCOUNT,
            L_TAX as L_TAX,
            O_TOTALPRICE as O_TOTAL
       from ORDERS,
            LINEITEM
      where O_ORDERKEY = $O_KEY
        and L_ORDERKEY = O_ORDERKEY
        and L_LINENUMBER = $L_KEY;
     \q
EOF13

fi
done < orderkeys.txt

echo
echo Atom - Rollback -- Records in History Table
psql -q -t tpch <<EOF14
     select *
       from HISTORY;
     \q
EOF14

#
######################################################
####
# Run COMMIT case

echo
echo Atomicity Test -- Starting COMMIT loop at: `date`

echo
i="0"
while read O_KEY L_KEY DELTA
do
if [ ${L_KEY}x != "x" ]; then
  echo
-------------------------------------------------
-------
  echo Atom - Commit -- For O_KEY = $O_KEY L_KEY =
$L_KEY DELTA = $DELTA
  echo Atom - Commit -- Before Values
```

```
  i=$[$i+1]
  psql -q -t tpch <<EOF22
     select
            L_EXTENDEDPRICE as L_EXTENDED,
            L_QUANTITY as L_QUANTITY,
            L_DISCOUNT as L_DISCOUNT,
            L_TAX as L_TAX,
            O_TOTALPRICE as O_TOTAL
       from ORDERS,
            LINEITEM
      where O_ORDERKEY = $O_KEY
        and L_ORDERKEY = O_ORDERKEY
        and L_LINENUMBER = $L_KEY;
     \q
EOF22

./acid_trans.sh $O_KEY $L_KEY $DELTA COMMIT

  echo
  echo Atom - Commit -- After Values
  psql -q -t tpch <<EOF23
     select
            L_EXTENDEDPRICE as L_EXTENDED,
            L_QUANTITY as L_QUANTITY,
            L_DISCOUNT as L_DISCOUNT,
            L_TAX as L_TAX,
            O_TOTALPRICE as O_TOTAL
       from ORDERS,
            LINEITEM
      where O_ORDERKEY = $O_KEY
        and L_ORDERKEY = O_ORDERKEY
        and L_LINENUMBER = $L_KEY;
     \q
EOF23

fi
done < orderkeys.txt

echo
echo Atom - Commit -- Records in History Table
psql -q -t tpch <<EOF24
     select *
       from HISTORY;
     \q
EOF24

echo
echo Atomicity Test -- Completed at: `date`
exit
```

```
====================================
acid_consistency_loop.sh
====================================
#$SHELL
#
# TPC-H ACID Consistency Transaction Loop
#
# Copyright 2007 ParAccel, Inc
#

echo Transaction Loop -- Starts `date`
echo
KEYFILE=$1
echo Transaction Loop -- Reading from $KEYFILE
# Loop through the orderkey and linenumber in
consistency_values.txt

while read O_KEY L_KEY DELTA
do
   ./acid_trans.sh $O_KEY $L_KEY $DELTA COMMIT
   sleep 1
done < $KEYFILE

echo Transaction Loop -- Completed `date`
```

```
====================================
acid_consistency_main.sh
====================================
#$SHELL
#
# TPC-H ACID Consistency Main
#
# Copyright 2007 ParAccel, Inc
#
NUMOFKEYS=100
echo Consistency Test -- Starts `date`
echo
./acid_setup.sh $NUMOFKEYS con_orderkeys1.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys2.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys3.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys4.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys5.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys6.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys7.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys8.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys9.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys10.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys11.txt
# Get sample number of keys for verification
tail -3 con_orderkeys1.txt > con_samplekeys.txt
tail -3 con_orderkeys2.txt >> con_samplekeys.txt
tail -3 con_orderkeys3.txt >> con_samplekeys.txt
tail -3 con_orderkeys4.txt >> con_samplekeys.txt
tail -3 con_orderkeys5.txt >> con_samplekeys.txt
tail -3 con_orderkeys6.txt >> con_samplekeys.txt
tail -3 con_orderkeys7.txt >> con_samplekeys.txt
tail -3 con_orderkeys8.txt >> con_samplekeys.txt
tail -3 con_orderkeys9.txt >> con_samplekeys.txt
tail -3 con_orderkeys10.txt >> con_samplekeys.txt
tail -3 con_orderkeys11.txt >> con_samplekeys.txt
echo
echo Consistency Test -- Pre Verify Start `date`
echo
# loop through sample keys for verification
while read O_KEY LKEY DELTA
do
    psql -q -t tpch <<EOF10
    select O_ORDERKEY,
           O_TOTALPRICE,
           sum(trunc(trunc(L_EXTENDEDPRICE*
           (1-L_DISCOUNT),2)*(1+L_TAX),2)),
           trunc(O_TOTALPRICE -
sum(trunc(trunc(L_EXTENDEDPRICE*
           (1-L_DISCOUNT),2)*(1+L_TAX),2)),1)
     from ORDERS join LINEITEM
       on O_ORDERKEY = L_ORDERKEY
      and O_ORDERKEY = $O_KEY
   group by 1, 2;
\q
EOF10
done < con_samplekeys.txt
echo
echo Consistency Test -- Pre Verify Complete `date`
echo
echo Consistency Test -- Start Transactions
./acid_consistency_loop.sh con_orderkeys1.txt >
con_log1 &
sleep 3
./acid_consistency_loop.sh con_orderkeys2.txt >
con_log2 &
sleep 3
./acid_consistency_loop.sh con_orderkeys3.txt >
con_log3 &
sleep 3
./acid_consistency_loop.sh con_orderkeys4.txt >
con_log4 &
sleep 3
./acid_consistency_loop.sh con_orderkeys5.txt >
con_log5 &
sleep 3
./acid_consistency_loop.sh con_orderkeys6.txt >
con_log6 &
```

```
sleep 3
./acid_consistency_loop.sh con_orderkeys7.txt >
con_log7 &
sleep 3
./acid_consistency_loop.sh con_orderkeys8.txt >
con_log8 &
sleep 3
./acid_consistency_loop.sh con_orderkeys9.txt >
con_log9 &
sleep 3
./acid_consistency_loop.sh con_orderkeys10.txt >
con_log10 &
sleep 3
./acid_consistency_loop.sh con_orderkeys11.txt >
con_log11 &
wait
echo
echo Consistency Test -- Transactions Completed
echo
echo Consistency Test -- Post Verify Starts `date`
echo

# loop through sample keys for verification
while read O_KEY LKEY DELTA
do
    psql -q -t tpch <<EOF10
    select O_ORDERKEY,
           O_TOTALPRICE,
           sum(trunc(trunc(L_EXTENDEDPRICE*
           (1-L_DISCOUNT),2)*(1+L_TAX),2)),
           trunc(O_TOTALPRICE -
sum(trunc(trunc(L_EXTENDEDPRICE*
           (1-L_DISCOUNT),2)*(1+L_TAX),2)),1)
    from ORDERS join LINEITEM
      on O_ORDERKEY = L_ORDERKEY
     and O_ORDERKEY = $O_KEY
   group by 1, 2;
\q
EOF10
done < con_samplekeys.txt

echo Consistency Test -- Post Verify Complete `date`
echo
echo Consistency Test -- Complete `date`
```

```
===================================
acid_consistency_main_v2.sh
===================================
#$SHELL
#
# TPC-H ACID Consistency Main
#
# Copyright 2007 ParAccel, Inc
#
NUMOFKEYS=10
echo Consistency Test -- Starts `date`
echo
./acid_setup.sh $NUMOFKEYS con_orderkeys1.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys2.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys3.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys4.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys5.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys6.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys7.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys8.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys9.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys10.txt
./acid_setup.sh $NUMOFKEYS con_orderkeys11.txt
./acid_consistency_loop.sh con_orderkeys1.txt >
con_log1 &
sleep 3
./acid_consistency_loop.sh con_orderkeys2.txt >
con_log2 &
sleep 3
./acid_consistency_loop.sh con_orderkeys3.txt >
con_log3 &
sleep 3
./acid_consistency_loop.sh con_orderkeys4.txt >
con_log4 &
sleep 3
./acid_consistency_loop.sh con_orderkeys5.txt >
con_log5 &
sleep 3
./acid_consistency_loop.sh con_orderkeys6.txt >
con_log6 &
sleep 3
./acid_consistency_loop.sh con_orderkeys7.txt >
con_log7 &
sleep 3
./acid_consistency_loop.sh con_orderkeys8.txt >
con_log8 &
sleep 3
./acid_consistency_loop.sh con_orderkeys9.txt >
con_log9 &
sleep 3
./acid_consistency_loop.sh con_orderkeys10.txt >
con_log10 &
sleep 3
./acid_consistency_loop.sh con_orderkeys11.txt >
con_log11 &
wait
echo Consistency Test -- Transactions Completed
echo
echo Consistency Test -- Verify Starts `date`
echo

# Get sample number of keys to verify
tail -3 con_orderkeys1.txt > con_samplekeys.txt
tail -3 con_orderkeys2.txt >> con_samplekeys.txt
tail -3 con_orderkeys3.txt >> con_samplekeys.txt
tail -3 con_orderkeys4.txt >> con_samplekeys.txt
tail -3 con_orderkeys5.txt >> con_samplekeys.txt
tail -3 con_orderkeys6.txt >> con_samplekeys.txt
tail -3 con_orderkeys7.txt >> con_samplekeys.txt
tail -3 con_orderkeys8.txt >> con_samplekeys.txt
tail -3 con_orderkeys9.txt >> con_samplekeys.txt
tail -3 con_orderkeys10.txt >> con_samplekeys.txt
tail -3 con_orderkeys11.txt >> con_samplekeys.txt

# loop through smaple keys for verification
while read O_KEY LKEY DELTA
do
    psql -q -t tpch <<EOF10
    select O_ORDERKEY,
           O_TOTALPRICE,
           sum(trunc(trunc(L_EXTENDEDPRICE*
           (1-L_DISCOUNT),2)*(1+L_TAX),2)),
           trunc(O_TOTALPRICE -
sum(trunc(trunc(L_EXTENDEDPRICE*
           (1-L_DISCOUNT),2)*(1+L_TAX),2)),2)
    from ORDERS join LINEITEM
      on O_ORDERKEY = L_ORDERKEY
     and O_ORDERKEY = $O_KEY
   group by 1, 2;
\q
EOF10

done < con_samplekeys.txt

echo Consistency Test -- Verify Complete `date`
echo
echo Consistency Test -- Complete `date`
```

```
===================================
acid_durability_loop.sh
===================================
#$SHELL
#
# TPC-H ACID Durability Transaction Loop
#
# Copyright 2007 ParAccel, Inc
#
```

```
echo Transaction Loop -- Starts `date`
echo
KEYFILE=$1
echo Transaction Loop -- Reading from $KEYFILE
# Loop through the orderkey and linenumber in
consistency_values.txt

while read O_KEY L_KEY DELTA
do
   ./acid_trans.sh $O_KEY $L_KEY $DELTA COMMIT
   if [ $? -eq 0 ]
      then
      echo $O_KEY "|" $L_KEY >> dur_keys.tbl
      sudo sync;sudo sync;sudo sync
   else
      exit $2
   fi
   sleep 1
done < $KEYFILE

echo
echo Durability Test -- First loop completed for `wc -
l $KEYFILE` transactions
echo
echo Durability Test --
-----------------------------------------------------
echo

while read O_KEY L_KEY DELTA
do
   ./acid_trans.sh $O_KEY $L_KEY $DELTA COMMIT
   if [ $? -eq 0 ]
      then
      echo $O_KEY "|" $L_KEY >> dur_keys.tbl
      sudo sync;sudo sync;sudo sync
   else
      exit $2
   fi
   sleep 1
done < $KEYFILE

echo
echo Transaction Loop -- Complete `date`
```

```
==================================
acid_durability_main.sh
==================================
#$SHELL
#
# TPC-H ACID Durability Main
#
# Copyright 2007 ParAccel, Inc
#
NUMOFKEYS=100
echo Durability Test -- Starts `date`
echo
echo XXXXXXXX ./acid_setup.sh $NUMOFKEYS
dur_orderkeys1.txt
psql tpch -At -c "select max(query) from stl_query"
./acid_setup.sh $NUMOFKEYS dur_orderkeys1.txt
echo XXXXXXXX ./acid_setup.sh $NUMOFKEYS
dur_orderkeys2.txt
psql tpch -At -c "select max(query) from stl_query"
./acid_setup.sh $NUMOFKEYS dur_orderkeys2.txt
echo XXXXXXXX ./acid_setup.sh $NUMOFKEYS
dur_orderkeys3.txt
psql tpch -At -c "select max(query) from stl_query"
./acid_setup.sh $NUMOFKEYS dur_orderkeys3.txt
echo XXXXXXXX ./acid_setup.sh $NUMOFKEYS
dur_orderkeys4.txt
psql tpch -At -c "select max(query) from stl_query"
./acid_setup.sh $NUMOFKEYS dur_orderkeys4.txt
echo XXXXXXXX ./acid_setup.sh $NUMOFKEYS
dur_orderkeys5.txt
psql tpch -At -c "select max(query) from stl_query"
./acid_setup.sh $NUMOFKEYS dur_orderkeys5.txt
echo XXXXXXXX ./acid_setup.sh $NUMOFKEYS
dur_orderkeys6.txt
psql tpch -At -c "select max(query) from stl_query"
./acid_setup.sh $NUMOFKEYS dur_orderkeys6.txt
echo XXXXXXXX ./acid_setup.sh $NUMOFKEYS
dur_orderkeys7.txt
psql tpch -At -c "select max(query) from stl_query"
./acid_setup.sh $NUMOFKEYS dur_orderkeys7.txt
echo XXXXXXXX ./acid_setup.sh $NUMOFKEYS
dur_orderkeys8.txt
psql tpch -At -c "select max(query) from stl_query"
./acid_setup.sh $NUMOFKEYS dur_orderkeys8.txt
echo XXXXXXXX ./acid_setup.sh $NUMOFKEYS
dur_orderkeys9.txt
psql tpch -At -c "select max(query) from stl_query"
./acid_setup.sh $NUMOFKEYS dur_orderkeys9.txt
echo XXXXXXXX ./acid_setup.sh $NUMOFKEYS
dur_orderkeys10.txt
psql tpch -At -c "select max(query) from stl_query"
./acid_setup.sh $NUMOFKEYS dur_orderkeys10.txt
echo XXXXXXXX ./acid_setup.sh $NUMOFKEYS
dur_orderkeys11.txt
psql tpch -At -c "select max(query) from stl_query"
./acid_setup.sh $NUMOFKEYS dur_orderkeys11.txt

if [ -f dur_keys.tbl ]
then
  rm dur_keys.tbl
fi
# Pre Verify Step
./acid_durability_verify.sh
# Start test
echo Durability Test - Start Transaction Loop
./acid_durability_loop.sh dur_orderkeys1.txt >
dur_log1 &
sleep 3
./acid_durability_loop.sh dur_orderkeys2.txt >
dur_log2 &
sleep 3
./acid_durability_loop.sh dur_orderkeys3.txt >
dur_log3 &
sleep 3
./acid_durability_loop.sh dur_orderkeys4.txt >
dur_log4 &
sleep 3
./acid_durability_loop.sh dur_orderkeys5.txt >
dur_log5 &
sleep 3
./acid_durability_loop.sh dur_orderkeys6.txt >
dur_log6 &
sleep 3
./acid_durability_loop.sh dur_orderkeys7.txt >
dur_log7 &
sleep 3
./acid_durability_loop.sh dur_orderkeys8.txt >
dur_log8 &
sleep 3
./acid_durability_loop.sh dur_orderkeys9.txt >
dur_log9 &
sleep 3
./acid_durability_loop.sh dur_orderkeys10.txt >
dur_log10 &
sleep 3
./acid_durability_loop.sh dur_orderkeys11.txt >
dur_log11 &
wait
echo Durability Test -- Transactions Completed
echo
# Get count from the history table
HIST_COUNT=`psql -q -t tpch <<EOF10
    select count(*)
      from HISTORY;
\q
EOF10`

echo
echo Durability Test -- Complete `date`
```

```
====================================
acid_durability_verify.sh
====================================
#$SHELL
#
# TPC-H ACID Durability Main
#
# Copyright 2007 ParAccel, Inc
#
# Get count from the history table
HIST_COUNT=`psql -q -t tpch <<EOF10
    select count(*)
      from HISTORY;
\q
EOF10`
if [ $? -ne 0 ]
   then
      echo
      echo Durability Test -- Database is DOWN
      echo
      exit $2
   fi

echo Durability Test -- History count $HIST_COUNT
echo
echo Durability Test -- Verify Starts `date`
echo

# Get sample number of keys to verify
tail -3 dur_orderkeys1.txt > dur_samplekeys.txt
tail -3 dur_orderkeys2.txt >> dur_samplekeys.txt
tail -3 dur_orderkeys3.txt >> dur_samplekeys.txt
tail -3 dur_orderkeys4.txt >> dur_samplekeys.txt
tail -3 dur_orderkeys5.txt >> dur_samplekeys.txt
tail -3 dur_orderkeys6.txt >> dur_samplekeys.txt
tail -3 dur_orderkeys7.txt >> dur_samplekeys.txt
tail -3 dur_orderkeys8.txt >> dur_samplekeys.txt
tail -3 dur_orderkeys9.txt >> dur_samplekeys.txt
tail -3 dur_orderkeys10.txt >> dur_samplekeys.txt
tail -3 dur_orderkeys11.txt >> dur_samplekeys.txt

# loop through smaple keys for verification
while read O_KEY LKEY DELTA
do
    psql -q -t tpch <<EOF10
    select O_ORDERKEY,
           O_TOTALPRICE,
           sum(trunc(trunc(L_EXTENDEDPRICE*
           (1-L_DISCOUNT),2)*(1+L_TAX),2)),
           trunc(O_TOTALPRICE -
sum(trunc(trunc(L_EXTENDEDPRICE*
           (1-L_DISCOUNT),2)*(1+L_TAX),2)),1)
    from ORDERS join LINEITEM
      on O_ORDERKEY = L_ORDERKEY
     and O_ORDERKEY = $O_KEY
   group by 1, 2;
\q
EOF10

done < dur_samplekeys.txt

echo Durability Test -- Verify Complete `date`
echo



====================================
acid_isolation_main1.sh
====================================
#$SHELL
#
# TPC-H ACID Isolation Main 1
#
# Copyright 2007 ParAccel, Inc
#
echo ISO Test 1 -- Test Starting at `date`
read O_KEY L_KEY DELTA < orderkeys.txt
echo ISO Test 1 -- Before Query
./acid_query.sh $O_KEY
echo ISO Test 1 -- Starting Transaction
# Run Transaction
./acid_trans.sh $O_KEY $L_KEY $DELTA COMMIT 15 &
sleep 5
echo ISO Test 1 -- Starting Query
./acid_query.sh $O_KEY
echo ISO Test 1 -- Query Completed
wait
echo ISO Test 1 -- All Completed
./acid_query.sh $O_KEY
echo ISO Test 1 -- End of Test at `date`
exit 0



====================================
acid_isolation_main2.sh
====================================
#$SHELL
#
# TPC-H ACID Isolation Main 2
#
# Copyright 2007 ParAccel, Inc
#

echo ISO Test 2 -- Test Starting at `date`
read O_KEY L_KEY DELTA < orderkeys.txt
echo ISO Test 2 -- Before Query
./acid_query.sh $O_KEY
echo ISO Test 2 -- Starting Transaction
# Run Transaction
./acid_trans.sh $O_KEY $L_KEY $DELTA ROLLBACK 15 &
sleep 5
echo ISO Test 2 -- Starting Query
./acid_query.sh $O_KEY
echo ISO Test 2 -- Query Completed
wait
echo ISO Test 2 -- All Completed
./acid_query.sh $O_KEY
echo ISO Test 2 -- End of Test at `date`
exit 0



====================================
acid_isolation_main3.sh
====================================
#$SHELL
#
# TPC-H ACID Isolation Main 1
#
# Copyright 2007 ParAccel, Inc
#

echo ISO Test 3 -- Test Starting at `date`
read O_KEY L_KEY DELTA < orderkeys.txt
echo ISO Test 3 -- Before Query
./acid_query.sh $O_KEY
echo ISO Test 3 -- Starting Transaction 1
# Run Transaction 1
./acid_trans.sh $O_KEY $L_KEY $DELTA COMMIT 15 &
sleep 5
echo ISO Test 3 -- Starting Transaction 2
# Run Transaction 2
./acid_trans.sh $O_KEY $L_KEY $DELTA COMMIT
echo ISO Test 3 -- Transaction 2 Completed
wait
echo ISO Test 3 -- All Completed
./acid_query.sh $O_KEY
echo ISO Test 3 -- Test Completed at `date`
exit 0
```

```
====================================
acid_isolation_main4.sh
====================================
#$SHELL
#
# TPC-H ACID Isolation Main 1
#
# Copyright 2007 ParAccel, Inc
#

echo ISO Test 4 -- Test Starting at `date`
read O_KEY L_KEY DELTA < orderkeys.txt
echo ISO Test 4 -- Before Query
./acid_query.sh $O_KEY
echo ISO Test 4 -- Starting Transaction 1
# Run Transaction 1
./acid_trans.sh $O_KEY $L_KEY $DELTA ROLLBACK 15 &
sleep 5
echo ISO Test 4 -- Starting Transaction 2
# Run Transaction 2
./acid_trans.sh $O_KEY $L_KEY $DELTA COMMIT
echo ISO Test 4 -- Transaction 2 Completed
wait
echo ISO Test 4 -- All Completed
./acid_query.sh $O_KEY
echo ISO Test 4 -- Test Completed at `date`
exit 0
```

```
====================================
acid_isolation_main5.sh
====================================
#$SHELL
#
# TPC-H ACID Isolation Main 5
#
# Copyright 2007 ParAccel, Inc
#

echo ISO Test 5 -- Test Starting at `date`

read O_KEY L_KEY DELTA < orderkeys.txt

echo ISO Test 5 -- Fetching PartKey and SupKey
P_KEY=`psql -q -t tpch <<EOF1
     select L_PARTKEY
       from LINEITEM
      where L_ORDERKEY = $O_KEY
        and L_LINENUMBER  = $L_KEY
     \q
EOF1`

S_KEY=`psql -q -t tpch <<EOF2
     select L_SUPPKEY
       from LINEITEM
      where L_ORDERKEY = $O_KEY
        and L_LINENUMBER  = $L_KEY
     \q
EOF2`
echo ISO Test 5 -- PartKey = $P_KEY, SuppKey = $S_KEY

echo ISO Test 5 -- Before Query
./acid_query.sh $O_KEY
echo ISO Test 5 -- Starting Transaction 1
# Run Transaction 1
./acid_trans.sh $O_KEY $L_KEY $DELTA COMMIT 15 &
sleep 5
echo ISO Test 5 -- Starting Partsup Query
# Run Transaction 2
./iso5_query.sh $P_KEY $S_KEY
echo ISO Test 5 -- Partsup Query Completed
wait
echo ISO Test 5 -- All Completed
./acid_query.sh $O_KEY
echo ISO Test 5 -- Test Completed at `date`
exit 0
```

```
====================================
acid_isolation_main6.sh
====================================
#$SHELL
#
# TPC-H ACID Isolation Main 6
#
# Copyright 2007 ParAccel, Inc
#

echo ISO Test 6 -- Test Starting at `date`
read O_KEY1 L_KEY1 DELTA1 < orderkeys.txt
tail -1 orderkeys.txt > /tmp/orderkeys.tail
read O_KEY2 L_KEY2 DELTA2 < /tmp/orderkeys.tail

echo ISO Test 6 -- Before Query
./acid_query.sh $O_KEY1
echo ISO Test 6 -- Starting ISO 6 Query
./iso6_query.sh $O_KEY2 &
sleep 2
echo ISO Test 6 -- Starting Transaction
./acid_trans.sh $O_KEY1 $L_KEY1 $DELTA1 COMMIT
echo ISO Test 6 -- Transaction Completed
wait
echo ISO Test 6 -- All Completed
./acid_query.sh $O_KEY1
echo ISO Test 6 -- Test Completed at `date`
exit 0
```

```
====================================
acid_query.sh
====================================
#$SHELL
#
# TPC-H ACID Query
#
# Copyright 2007 ParAccel, Inc
#
O_KEY=$1

# Execute read-only query for isolation tests

echo "ACID Query -- Started at `date`"
O_TOTAL=`psql -q -t tpch <<EOF13
     select round(o_totalprice,2)
     from ORDERS
      where O_ORDERKEY = $O_KEY;
     \q
EOF13`
echo "ACID Query -- o_orderkey = $O_KEY"
echo "ACID Query -- o_total = $O_TOTAL"
echo "ACID Query -- Completed at `date`"
exit 0
```

```
====================================
acid_setup.sh
====================================
#$SHELL
#
# TPC-H ACID Setup
#
# Copyright 2007 ParAccel, Inc
#

echo ParAccel ACID Setup Starts `date`
echo


if [ -z "$2" ]
then
  OUTFILE="orderkeys.txt"
```

```
else
  OUTFILE="$2"
fi

# Clean history table
psql -q -t tpch <<EOF10
     delete from HISTORY;
     \q
EOF10

# Create file with order keys to be used for
consistency testing

if [ -f $OUTFILE ]
then
  rm $OUTFILE
fi

# Get Max Order Key
MAX_O_KEY=`psql -q -t tpch <<EOF10
     select  max(O_ORDERKEY)
        from ORDERS;
     \q
EOF10`

# echo Maximum Order Key $MAX_O_KEY

i="0"
while [ $i -lt $1 ]
do
# Get Random Orderkey and maximum lineitem
  DELTA=$RANDOM
  let "DELTA %= 101"
  if [ $DELTA -eq 0 ]; then
     DELTA=1
  fi
  O_KEY=$RANDOM
  let "O_KEY %= ${MAX_O_KEY}"
  L_KEY=`psql -q -t tpch <<EOF11
     select  max(L_LINENUMBER)
        from LINEITEM
       where L_ORDERKEY = $O_KEY;
  \q
EOF11`
if [ ${L_KEY}x != "x" ]; then
  echo $O_KEY $L_KEY $DELTA
  echo $O_KEY $L_KEY $DELTA >> $OUTFILE
  i=$[$i+1]

fi
done

echo
echo ParAccel ACID  Setup Complete `date`

exit



====================================
acid_trans.sh
====================================
#$SHELL
#
# TPC-H ACID Transaction
#
# Copyright 2007 ParAccel, Inc
#
O_KEY=$1
L_KEY=$2
DELTA=$3
COMMIT=$4
SLEEP=$5

echo ACID Trans -- Started at: `date`

if [ -z $SLEEP ]
   then
     let "SLEEP = 0"
     echo ACID Trans -- No Sleep before ${COMMIT}
   else
     let "SLEEP = $SLEEP"
     echo ACID Trans -- With Sleep = $SLEEP before $
{COMMIT}
fi

# Perform updates and insert for the ACID test

echo ACID Trans -- o_orderkey = ${O_KEY}, line_number
= ${L_KEY}, delta = $DELTA

psql -q -t tpch <<EOFb >> /dev/null
BEGIN TRANSACTION;

-- 1. Update Orders a first time to remove old
lineitem from total price

update ORDERS
   set O_TOTALPRICE = O_TOTALPRICE - (
     select TRUNC( TRUNC(L_EXTENDEDPRICE * (1.0 -
L_DISCOUNT), 2) * (1.0 + L_TAX), 2 )
       from LINEITEM
      where L_ORDERKEY = $O_KEY
        and L_LINENUMBER = $L_KEY )
 where O_ORDERKEY = $O_KEY ;

-- 2. Update Linetime to increase quantity

update LINEITEM
   set L_EXTENDEDPRICE = L_EXTENDEDPRICE +
TRUNC( TRUNC(L_EXTENDEDPRICE/L_QUANTITY, 2) * $DELTA,
2 ),
       L_QUANTITY = L_QUANTITY +  $DELTA
 where L_ORDERKEY = $O_KEY
   and L_LINENUMBER = $L_KEY ;

-- 3. Update Orders a second time to add updated
lineitem to total price

update ORDERS
   set O_TOTALPRICE = O_TOTALPRICE + (
     select TRUNC( TRUNC(L_EXTENDEDPRICE * (1.0 -
L_DISCOUNT), 2)* (1 + L_TAX), 2 )
       from LINEITEM
      where L_ORDERKEY = $O_KEY
        and L_LINENUMBER = $L_KEY )
 where O_ORDERKEY = $O_KEY ;

-- 4. Insert History records of transaction

insert into HISTORY
   (select L_PARTKEY, L_SUPPKEY, L_ORDERKEY,
L_LINENUMBER, $DELTA, LOCALTIMESTAMP
      from LINEITEM
     where L_ORDERKEY = $O_KEY
       and L_LINENUMBER = $L_KEY );

select sleep($SLEEP);

$COMMIT TRANSACTION;
EOFb
RC=$?
echo ACID Trans -- Completed at: `date`
exit $RC



====================================
check_counts.sh
====================================
#$SHELL
while(true)
do
num=`(cat dur_keys.tbl | wc -l)`
if (( $num < 10000 ))
then
```

```
   echo "The test has not yet reached the mid-point"
   echo
"------------------------------------------------------"
   date
   for arg in 1 2 3 4 5 6 7 8 9 ; do cat dur_log$arg |
grep -i "first loop"; done
   echo "Transaction count: $num"
   echo
"------------------------------------------------------"
fi
sleep $1
done



====================================
check_progress.sh
====================================
#$SHELL
source $HOME/.bashrc
while true; do
for arg in 1 2 3 4 5 6 7 8 9 10 11; do
if [ -f dur_log$arg ]
then
count=`(cat dur_log$arg | grep -i Completed | wc -l)`
echo "Current count for dur_log$arg = $count"
else
echo "dur$arg does not exist yet. It is probably too
early in the test sequence."
fi;
done
echo
"------------------------------------------------------
-----------------------"
echo "To check progress, run this script. Also tail -f
acid_durability_main.out. "
echo "Once all streams have reached 100 transactions:"

echo "     1) Pull a drive on any node"
echo "     2) Wait 60 seconds"
echo "     3) Crash the entire cluster (26 nodes)"
echo "     4) After a couple of minutes, power the
server back on and restart xen"
echo "     5) Verify that xen is running (if in doubt:
psql tpch and wait)  run acid_durability_verify.sh"
echo "     6) As a sanity check, compare a wc -l
dur_keys against psql tpch -c select count(*) from
history"
sleep 10;
clear;
done



====================================
check_transactions.sh
====================================
#$SHELL
count1=`(cat $HOME/run/tpch_acid/dur_keys.tbl | wc
-l)`
count2=0
while true
  do
  sleep 10
  date
  count1=`(cat $HOME/run/tpch_acid/dur_keys.tbl | wc -
l)`
  if (( $count2 >= $count1 ))
    then
    echo "Error: The transactions have stopped!"
    exit
  fi
  ((count2=count1))
  echo
"------------------------------------------------------
-"
  echo "Transactions are still running. Current count:
$count1"
  echo
"------------------------------------------------------
-"
done



====================================
clean.sh
====================================
:
rm -f *.txt
rm -f *.out
rm -f *_log*
rm -f *.tbl


====================================
create_history.ddl
====================================
create table HISTORY
(
H_P_KEY int,
H_S_KEY int,
H_O_KEY int,
H_L_KEY int,
H_DELTA int,
H_DATE_T datetime
);



====================================
iso5_query.sh
====================================
#$SHELL
#
# TPC-H Isolation Query for ISO 5
#
# Copyright 2007 ParAccel, Inc
#
P_KEY=$1
S_KEY=$2

echo ISO Query -- Starting at `date`

PS_DATA=`psql -q -t tpch <<EOF1
     select PS_AVAILQTY
       from PARTSUPP
      where PS_PARTKEY = $P_KEY
        and PS_SUPPKEY  = $S_KEY
     \q
EOF1`
echo ISO Query -- PS_AVAILQTY = $PS_DATA

PS_DATA=`psql -q -t tpch <<EOF2
     select PS_SUPPLYCOST
       from PARTSUPP
      where PS_PARTKEY = $P_KEY
        and PS_SUPPKEY  = $S_KEY
     \q
EOF2`
echo ISO Query -- PS_SUPPLYCOST = $PS_DATA

PS_DATA=`psql -q -t tpch <<EOF3
     select PS_COMMENT
       from PARTSUPP
      where PS_PARTKEY = $P_KEY
        and PS_SUPPKEY  = $S_KEY
     \q
EOF3`
echo ISO Query -- PS_COMMENT = $PS_DATA

echo ISO Query -- Completed at `date`
```

```
====================================
iso6_query.sh
====================================
#$SHELL
#
# TPC-H Isolation Query for ISO 6
#
# Copyright 2007 ParAccel, Inc
#
O_KEY=$1

echo ISO6 Query -- Starting at `date`

PS_DATA=`psql -q -t tpch <<EOF1
select L_EXTENDEDPRICE, SLEEP(L_LINENUMBER*4)
  from LINEITEM
 where L_ORDERKEY = $O_KEY
   and L_LINENUMBER < 4
     \q
EOF1`
echo ISO6 Query -- L_EXTENDEDPRICE = $PS_DATA

echo ISO6 Query -- Completed at `date`
```

```
====================================================
```

Disk Configuration Details

```
====================================================
```

Same as described for the SUT.

Appendix C. Query Text and Query Output

```
echo ============================
echo Query 1
echo ============================

select
      l_returnflag,
      l_linestatus,
      sum(l_quantity) as sum_qty,
      sum(l_extendedprice) as sum_base_price,
      sum(l_extendedprice * (1 - l_discount)) as
sum_disc_price,
      sum(l_extendedprice * (1 - l_discount) * (1 +
l_tax)) as sum_charge,
      avg(l_quantity) as avg_qty,
      avg(l_extendedprice) as avg_price,
      avg(l_discount) as avg_disc,
      count(*) as count_order
from
      lineitem
where
      l_shipdate <= cast(date '1998-12-01' - interval
'98 days' as date)
group by
      l_returnflag,
      l_linestatus
order by
      l_returnflag,
      l_linestatus;
 l_returnflag | l_linestatus |     sum_qty      |
sum_base_price    |   sum_disc_price    |
sum_charge      | avg_qty | avg_price | avg_disc |
count_order
--------------+--------------+------------------
+--------------------+--------------------
+--------------------+---------+-----------
+----------+-------------
 A            | F            | 1133846474781.00 |
1700201241530186.09 | 1615188885805628.31 |
1679795726223766.50 |   25.49 |  38236.97 |     0.05 |
44464843008
 N            | F            |   29599197360.00 |
44384596502697.04 |   42165304218999.33 |
43852111415466.34 |   25.50 |  38238.24 |     0.05 |
1160738327
 N            | O            | 2220738791701.00 |
3329997398065660.71 | 3163497867007125.27 |
3290038953060585.33 |   25.50 |  38237.32 |     0.04 |
87087607384
 R            | F            | 1133835832568.00 |
1700182891927325.96 | 1615173638069019.89 |
1679780176305915.52 |   25.49 |  38236.95 |     0.04 |
44464392940
(4 rows)

echo ============================
echo Query 2
echo ============================

select
      s_acctbal,
      s_name,
      n_name,
      p_partkey,
      p_mfgr,
      s_address,
      s_phone,
      s_comment
from
      part,
      supplier,
      partsupp,
      nation,
      region
where
      p_partkey = ps_partkey
      and s_suppkey = ps_suppkey
      and p_size = 30
      and p_type like '%BRASS'
      and s_nationkey = n_nationkey
      and n_regionkey = r_regionkey
      and r_name = 'AFRICA'
      and ps_supplycost = (
            select
                  min(ps_supplycost)
            from
                  partsupp,
                  supplier,
                  nation,
                  region
            where
                  p_partkey = ps_partkey
                  and s_suppkey = ps_suppkey
                  and s_nationkey = n_nationkey
                  and n_regionkey = r_regionkey
                  and r_name = 'AFRICA'
      )
order by
      s_acctbal desc,
      n_name,
      s_name,
      p_partkey
limit 100;
 s_acctbal |          s_name           |
n_name           | p_partkey  |
p_mfgr           |
s_address                 |     s_phone
|
s_comment

-----------+---------------------------
+---------------------------+------------
+---------------------------
+-----------------------------------------
+-----------------
+----------------------------------------------------
```

```
-------------------------------------------------
   9999.99 | Supplier#036934181        |
ALGERIA                   | 1911934162 |
Manufacturer#1            |
bIuDqKjAnXdk                              | 10-724-408-
2906 | uests. ironic instructions after the furiously
pendi
   9999.99 | Supplier#085919020        |
ALGERIA                   | 2485919019 |
Manufacturer#3            |
0udYKNIZrOIbDNTl4nmS4lM3                  | 10-932-484-
5529 | s. slyly daring deposits wake blithely ironic
excuses. quickly thin deposits are bli
   9999.99 | Supplier#167705450        |
ALGERIA                   | 1967705449 |
Manufacturer#3            | 4f
yllcJpDpJBMVepuLTopJY1nJ              | 10-785-172-
4238 | ld ideas cajole. foxes according to the
regular, express ideas
   9999.99 | Supplier#233460252        |
ALGERIA                   | 4283460223 |
Manufacturer#4            |
04WjOGZkWPcPj                             | 10-825-202-
5085 |  the unusual, special theodolites. quickly
regular foxes would believe blithely quickly regul
   9999.99 | Supplier#289029030        |
ALGERIA                   | 4714029014 |
Manufacturer#2            |
izoa3afnQrV0cEqqKZhRCYg                   | 10-226-704-
7003 | ithely express foxes x-ray. blithely even
instructions haggle blithely
   9999.99 | Supplier#013423473        |
ETHIOPIA                  | 4213423472 |
Manufacturer#5            |
wBmOeATmPfpmwAZmXniGC7AdBStL              | 15-277-860-
2930 | nal theodolites. blithely final ideas haggle
furiously ironic instructions. slyly even sentiments
   9999.99 | Supplier#013423473        |
ETHIOPIA                  | 5563423436 |
Manufacturer#5            |
wBmOeATmPfpmwAZmXniGC7AdBStL              | 15-277-860-
2930 | nal theodolites. blithely final ideas haggle
furiously ironic instructions. slyly even sentiments
   9999.99 | Supplier#016617001        |
ETHIOPIA                  |  166617000 |
Manufacturer#2            |
Lr6yxLrw72jrlm13qMkg1D                    | 15-906-109-
2822 | arefully regular foxes use carefully. final
packages wake carefully carefully bol
   9999.99 | Supplier#026080009        |
ETHIOPIA                  | 1526080008 |
Manufacturer#3            | bSlmo4l87j3JfsQTX2
c                     | 15-124-630-9564 | ely silent
requests. close requests nag furiously after the slyly
ironic r
   9999.99 | Supplier#122201595        |
ETHIOPIA                  | 1472201586 |
Manufacturer#2            |
wv4HwhcGflFHwn77iGbKu6ujz7NM5eTAskf       | 15-420-109-
1347 | e carefully special accounts! slyly silent
requests use furiously fluffi
   9999.99 | Supplier#205401343        |
ETHIOPIA                  |  205401342 |
Manufacturer#2            | k0HHLO 5oC
q1wBDKAjzxRXsIMFdLdu,llL9,    | 15-188-217-8358 | ng
courts. blithely even accounts
   9999.99 | Supplier#217312458        |
ETHIOPIA                  | 2167312443 |
Manufacturer#3            | q
uV07zBMXFXd26G2N                          | 15-644-770-
3500 | egrate. final packages haggle along the
   9999.99 | Supplier#219745743        |
ETHIOPIA                  | 1569745732 |
Manufacturer#2            | 1Ou l
PAjvTmdMh,EJFh2Yf2HhKzG2            | 15-146-523-8488
|  are slyly. even, ironic deposits haggle along the
blithe
   9999.99 | Supplier#263467795        |
ETHIOPIA                  |  113467794 |
Manufacturer#1            |
o7GuUYurcBXHq5aFebCkirNQW98odFv3          | 15-824-798-
5887 | s cajole. deposits nag blithel
   9999.99 | Supplier#029307963        |
KENYA                     | 5354307945 |
Manufacturer#4            |
DLvla,Bm7j0bk2Lg                          | 24-817-498-
2180 | sly. carefully final escapades detect at the
requests.
   9999.99 | Supplier#044630974        |
KENYA                     | 2294630959 |
Manufacturer#2            |
Dy,Ov3MqOeBy2VwoE9MEJHdsFwPnhnY           | 24-336-338-
6405 | requests nag quickly unusual pinto beans--
   9999.99 | Supplier#057603184        |
KENYA                     | 2082603177 |
Manufacturer#2            | t2iSiMIZxOi5
swfUA7I                     | 24-929-244-2741 | g to
the sometimes stealthy packages. braids b
   9999.99 | Supplier#117031184        |
KENYA                     | 4917031183 |
Manufacturer#1            | HZ2v4qno7QxmFfAeYYVOMen 2y
ygr3e BkwpxM  | 24-157-817-3700 | deposits. carefully
ironic packages until the furiously unusual theodol
   9999.99 | Supplier#297767722        |
KENYA                     | 5922767702 |
Manufacturer#1            |
2Cg1hRtTLxAeyXG7Fv0BErDUb71MDbyBm         | 24-763-157-
8594 |  dependencies are furiously
   9999.99 | Supplier#074747874        |
MOROCCO                   |  899747871 |
Manufacturer#2            | ZAD0pvE7
QvRukwV5QMy6                      | 25-987-255-3236 |
unusual requests. closely silent requests among
   9999.99 | Supplier#140347525        |
MOROCCO                   | 5615347470 |
Manufacturer#2            | K42v8ttApd ,lXUeS
OlGztlrHy,4x5eGaBLaW   | 25-427-673-7964 | s blithely
about the carefully unusual requests. pending in
   9999.99 | Supplier#204210637        |
MOROCCO                   | 3054210616 |
Manufacturer#2            | SUHKAF
dUh1Ojr7g                           | 25-252-167-3112 |
carefully regular, regular ideas. slyly regular
requests accordi
   9999.99 | Supplier#153860187        |
MOZAMBIQUE                | 4653860186 |
Manufacturer#1            | Hbc5
fje2fHzrUgcnSOpGZBf3geg5TxWdCG,E    | 26-996-935-3940
| e ironic, express foxes. bl
   9999.99 | Supplier#272361147        |
MOZAMBIQUE                | 5522361110 |
Manufacturer#4            | 8,
edqyUiKASU3c0GLhl3ND4GV                | 26-771-583-
9878 |  slyly even asymptotes. quickly even ideas
thrash against the fu
   9999.98 | Supplier#018473729        |
ALGERIA                   | 3468473706 |
Manufacturer#1            |
l1PutHC5K62rQHSUlb1v5Dw9J8hVm3ufAvQrRZ   | 10-972-321-
7112 | s. slyly regular ideas use blithely across the
slyly
   9999.98 | Supplier#086352502        |
ALGERIA                   |  761352495 |
Manufacturer#5            |  38j37zrDjX
tquF                        | 10-276-773-1988 |
ithes. silent, even ideas lose; slyly ironic
   9999.98 | Supplier#005826756        |
ETHIOPIA                  | 3680826719 |
Manufacturer#4            |
E05QYyaGRYlyq1xXQEvVikjAeiqKOAfStCRUL    | 15-753-827-
4027 | n accounts use blithely. express instructions
among the
   9999.98 | Supplier#078901520        |
ETHIOPIA                  | 4578901519 |
Manufacturer#3            |
0wkvyaws0fAYZXYnWsVpA3A25BL               | 15-677-306-
```

6937 | ter the fluffily idle ideas wake blithely unusual
   9999.98 | Supplier#006235057 | KENYA | 1881235038 | Manufacturer#4 | fiw,mI9hSZ66bUz0oHgW,q | 24-173-868-7564 | arefully blithely ironic
   9999.98 | Supplier#095346512 | KENYA | 1295346511 | Manufacturer#5 | 6 GzS2fh5W9c2Mk | 24-931-591-1511 | sleep closely regular packages. furiously special packages accordin
   9999.98 | Supplier#095346512 | KENYA | 4295346511 | Manufacturer#2 | 6 GzS2fh5W9c2Mk | 24-931-591-1511 | sleep closely regular packages. furiously special packages accordin
   9999.98 | Supplier#047783176 | MOROCCO | 3122783145 | Manufacturer#3 | 4MmxAnPMzmToKK4TOJ5BC3U73 | 25-956-569-6623 | luffily regular accounts nag quick
   9999.98 | Supplier#050091374 | MOROCCO | 3725091337 | Manufacturer#1 | cC4h5x4bi6I,cpeFsGzXf | 25-131-599-7821 | kly. carefully pending requests solve furiously. furiously regular accounts wake carefully fluffily
   9999.98 | Supplier#163297616 | MOROCCO | 463297615 | Manufacturer#5 | RICSLmjVuojVLwg0OucLX2GlSu27XRhSsvZL | 25-129-377-3590 | e slyly special attainments must use slyly qui
   9999.98 | Supplier#052506531 | MOZAMBIQUE | 3352506530 | Manufacturer#3 | jIVY0hMYwln | 26-267-967-7827 | uickly regular instructions cajole instructions. quickly regular accounts alo
   9999.98 | Supplier#101844131 | MOZAMBIQUE | 5726844111 | Manufacturer#4 | T0WFYnIyKiTi | 26-946-672-7424 | ng packages: slyly pending pinto beans after the th
   9999.98 | Supplier#208564090 | MOZAMBIQUE | 5008564089 | Manufacturer#3 | B8ozr5AQmCwhrguZxdFG4Si7 | 26-684-241-2760 | ts wake quickly against the furiously special r
   9999.98 | Supplier#230825120 | MOZAMBIQUE | 2255825112 | Manufacturer#3 | yFlol,GgxF2ZbA41 | 26-950-949-5143 | , even packages! quickly bold request
   9999.98 | Supplier#261744560 | MOZAMBIQUE | 2436744535 | Manufacturer#2 | cao38yypYJMGVYYp9UYe | 26-593-595-9827 | excuses nod slyly slyly express pinto beans. slyly bold requests haggle along the final dependencie
   9999.97 | Supplier#011109570 | ALGERIA | 1886109551 | Manufacturer#1 | 5uC3iOkaNOJ0pQU5w | 10-103-842-1027 | he final requests. carefully silent requests cajole along the fluffily enticing ins
   9999.97 | Supplier#001843313 | ETHIOPIA | 3526843301 | Manufacturer#3 | 8LTDEIjzfLOF8142Xsb6jSks | 15-625-702-3908 | about the fluffily even decoys cajole fluffily even,
   9999.97 | Supplier#070342352 | ETHIOPIA | 5395342334 | Manufacturer#2 | Pfj0DaHqZBM21jhLJURoEGwNUgf ,AJuDrED | 15-582-743-9076 | ounts haggle accounts. ironically regular requests cajole slyly. accounts sleep. slyly special
   9999.97 | Supplier#128237315 | ETHIOPIA | 503237311 | Manufacturer#2 | gzeyL2varHMqouh7npu10K1mlceqxbF52IN | 15-376-764-8329 | nts boost about the blithely ironic packages. slyly p
   9999.97 | Supplier#155996328 | ETHIOPIA | 3230996297 | Manufacturer#1 | aqE0rVDgV6ey CIQlGNGWSJi2Ujyj0x lnA | 15-332-441-3174 | s above the regular deposits boost according to the furiously final accounts. fluf
   9999.97 | Supplier#164687603 | ETHIOPIA | 1664687602 | Manufacturer#5 | IWHLHFWKCEpxos5OLGwQw93j5yaS | 15-258-925-6626 | hily slyly unusual dependencies. busily unusual deposits wake furiously against
   9999.97 | Supplier#228820775 | ETHIOPIA | 978820768 | Manufacturer#5 | lHF9YYTuFQZ5ZlCdFgAVxITUe6dQr37zYL1 | 15-968-365-7848 | resias cajole enticingly slyly ironic a
   9999.97 | Supplier#258682617 | ETHIOPIA | 4833682568 | Manufacturer#5 | Br5Kmw3gWB4l6QhJJ3hf4ewF8C79Oh,O | 15-462-307-8101 | blithely ironic dependencies. carefully pending packages nag f
   9999.97 | Supplier#010859835 | KENYA | 2860859816 | Manufacturer#2 | g0xY,J5Bm9aNGH | 24-734-655-3078 | ously ironic packages. slyly final deposits hang. blithely ironic pinto beans along the ironic inst
   9999.97 | Supplier#043666462 | KENYA | 4468666447 | Manufacturer#2 | dvN9uc6tEsvI0oaTUByCYJCmMQOosGQQfSGH4Z | 24-975-326-5528 | fluffily quickly final platelets. carefu
   9999.97 | Supplier#078091750 | KENYA | 2778091749 | Manufacturer#2 | IVGpdDn3skIHITiT0QDG35RUJMOBDoOPwCfQp7 | 24-765-982-4203 | l theodolites use special foxes. deposits
   9999.97 | Supplier#158077373 | KENYA | 5858077372 | Manufacturer#2 | OS869s1O3G7QYbNDx3qMmNjR,9 | 24-746-678-2321 | even dependencies. furious
   9999.97 | Supplier#163719913 | KENYA | 4288719898 | Manufacturer#1 | i,Kil3QfTJI3O4Q18hXZ9UdQ46MmjOC4R,v7sO | 24-129-620-3773 | are slyly even packages. fin
   9999.97 | Supplier#152654035 | MOROCCO | 3827653998 | Manufacturer#2 | MXOyJL0oTSjEFEQ30ZgmK | 25-354-841-1909 | ecial, final packages; furiously final requests cajole after the bold accounts. final in
   9999.97 | Supplier#186835724 | MOROCCO | 5136835689 | Manufacturer#1 | o5sBR3b5pD6dltif5wGkZ9WQG8cP1uc3 | 25-383-449-5203 | s hang quickly doggedly regular ideas. regular deposits sublat
   9999.97 | Supplier#187065420 | MOROCCO | 1687065419 | Manufacturer#2 | gBapBe,1Jq3E,bJjJOQfup,xW | 25-956-244-1561 | pinto beans. final sheaves cajole across the accounts. slowly f
   9999.97 | Supplier#221914566 |

MOROCCO | 3371914543 | Manufacturer#1 | nSPP0qSOBLhIqZvG9 n0nQzmHq,esoHOG | 25-809-886-2947 | packages. enticingly final pinto beans haggle perman
9999.97 | Supplier#025195142 | MOZAMBIQUE | 4375195113 | Manufacturer#4 | 25K8RHTf3LO6I0Aq0h5say0QFG49v | 26-453-801-5505 | thin pinto beans. even accounts cajole. final, regular deposits against the ideas wake against
9999.97 | Supplier#091893689 | MOZAMBIQUE | 2866893661 | Manufacturer#2 | cBp4op7VtG7cF | 26-552-773-4161 | regular tithes. express, final ideas haggle-- ironic, bold excuses boost about the pendin
9999.97 | Supplier#149775470 | MOZAMBIQUE | 4574775454 | Manufacturer#5 | Cwy5aaOvEPPKN | 26-737-763-4861 | bold dependencies. ideas are slyly carefully regula
9999.97 | Supplier#173558971 | MOZAMBIQUE | 473558970 | Manufacturer#5 | GAL50NTLqD6yescUGpH8Zghd47fYLTs | 26-564-531-2335 | accounts. busily bold theodolites wake. furiously final accounts hinder furi
9999.97 | Supplier#186662524 | MOZAMBIQUE | 3786662523 | Manufacturer#1 | FxUggGVON5vSL | 26-381-160-8000 | s use furiously regular requests. sp
9999.97 | Supplier#244073145 | MOZAMBIQUE | 394073142 | Manufacturer#2 | 5EicfWWvO S7J7gjqgxrsABsVodKoI9LulWDZ | 26-949-855-6647 | r, unusual ideas. furiously unusual accounts cajole fluffily e
9999.96 | Supplier#050258566 | ALGERIA | 2975258556 | Manufacturer#2 | uz9i3rrYajcLD9zQw3I | 10-841-590-5446 | lithely even ideas haggle across the ev
9999.96 | Supplier#028019445 | ETHIOPIA | 3778019420 | Manufacturer#5 | TMDMnCm0cHQKJL4uae4p7UrzzCvKMJ9 | 15-625-182-9493 | as. quickly regular foxes run packages. blithe
9999.96 | Supplier#028019445 | ETHIOPIA | 4453019430 | Manufacturer#2 | TMDMnCm0cHQKJL4uae4p7UrzzCvKMJ9 | 15-625-182-9493 | as. quickly regular foxes run packages. blithe
9999.96 | Supplier#048066700 | ETHIOPIA | 1173066696 | Manufacturer#1 | bX0VKHbUcHdCGXI | 15-948-221-6534 | yly silent packages sleep along the slyly regu
9999.96 | Supplier#244986396 | KENYA | 3619986359 | Manufacturer#4 | qxgNCb1Pg,UFtco5dE | 24-279-494-6311 | ggle accounts? fluffily pending packages after the blithely final deposits impress fur
9999.96 | Supplier#291081514 | KENYA | 2991081513 | Manufacturer#1 | YWb8WT2GRym4,x33 | 24-156-593-1934 | lyly pending instructions cajole b
9999.96 | Supplier#255169488 | MOROCCO | 2955169487 | Manufacturer#4 | QU2UhIJuErRwU4FXn | 25-416-609-9147 | nto beans. finally even Tires
9999.96 | Supplier#050100425 | MOZAMBIQUE | 2675100416 | Manufacturer#4 | lfPQ1ZgxErd,IuPjnLZJn | 26-684-908-2312 | s the theodolites. final requests after the accounts haggle furiously careful
9999.95 | Supplier#091868095 | ALGERIA | 391868094 | Manufacturer#3 | ZXBNgXsjjqW,gp3Dy0,vpM1CrA | 10-401-691-1111 | regular packages. blithely close dolphins wake fu
9999.95 | Supplier#114884700 | ALGERIA | 1089884690 | Manufacturer#2 | XTY8s0bNPRTm zv 9NwQrSH | 10-293-606-9777 | ts are regularly according to
9999.95 | Supplier#197612994 | ALGERIA | 1772612978 | Manufacturer#2 | hS5zNuNMotZpU | 10-646-350-7070 | se final packages. bold, special deposits impress b
9999.95 | Supplier#209905625 | ALGERIA | 5834905605 | Manufacturer#3 | B60pjE92Faz0Lq LJuQHEk4pg1R4Z | 10-855-582-7575 | ructions? blithely regular accounts according to the unusual f
9999.95 | Supplier#126828365 | ETHIOPIA | 1101828355 | Manufacturer#2 | 5xhvoHj2MXTYYkM2v3tUvQ7ZXHpJWtDDom | 15-646-320-5851 | gside of the final, regular pinto beans. carefully ironic ideas cajole carefully
9999.95 | Supplier#168465782 | ETHIOPIA | 4518465751 | Manufacturer#2 | Cz71qz5lCroswiQFcW5fh5SXsaD4ANhWayAMOe | 15-565-525-9861 | hely silent asymptotes boost carefully regular, final foxes.
9999.95 | Supplier#215569450 | ETHIOPIA | 1040569446 | Manufacturer#3 | 91lkRDeurk,ghvV653KEHUGh2TfRvNsv2v9W | 15-621-571-6447 | ual accounts after the furiously final reque
9999.95 | Supplier#215569450 | ETHIOPIA | 3440569438 | Manufacturer#2 | 91lkRDeurk,ghvV653KEHUGh2TfRvNsv2v9W | 15-621-571-6447 | ual accounts after the furiously final reque
9999.95 | Supplier#299758042 | ETHIOPIA | 224758041 | Manufacturer#5 | AiKgwsMRNbZ4KdoSwZ5,n6AzVzOBzlglEzUyf | 15-314-475-1069 | ven pinto beans haggle furiously around the sil
9999.95 | Supplier#299758042 | ETHIOPIA | 1424758037 | Manufacturer#4 | AiKgwsMRNbZ4KdoSwZ5,n6AzVzOBzlglEzUyf | 15-314-475-1069 | ven pinto beans haggle furiously around the sil
9999.95 | Supplier#015393170 | KENYA | 3465393147 | Manufacturer#5 | gEum7yPw9wob2ayqeT1U | 24-512-545-2076 | wake slyly fluffily bold deposits. bold instructions cajole quickly along the fo
9999.95 | Supplier#286535311 | MOROCCO | 2011535304 | Manufacturer#4 | liHOPRM0ONxYuhNY8HpdozxmlQod5eUvMZU | 25-613-944-4293 | the carefully pending theodolites; quickly final requests nag slyly furiously unusual ac
9999.95 | Supplier#286535311 | MOROCCO | 5011535294 | Manufacturer#3 | liHOPRM0ONxYuhNY8HpdozxmlQod5eUvMZU | 25-613-944-4293 | the carefully pending theodolites; quickly final requests nag slyly furiously unusual ac
9999.95 | Supplier#264495285 | MOZAMBIQUE | 5889495265 |

```
Manufacturer#2            |
2eZqrTwj98zvs                          | 26-724-771-
5327 | s. fluffily final deposits promise slyly bold
platelets. blithely even ideas are blithely.
   9999.94 | Supplier#018147712        |
ETHIOPIA                  | 4968147679 |
Manufacturer#1            |
DxcAe0FgX9OO1VKMM6MKhmP                | 15-554-811-
4684 | iously special instructions
   9999.94 | Supplier#097335751        |
ETHIOPIA                  | 5872335693 |
Manufacturer#2            |
DUTK0CQy7wnjGdtFQCBNP9UCXuHYhg5lAwtj     | 15-693-277-
6124 | cross the quickly bold ideas. pending, even
theodolites affix. care
   9999.94 | Supplier#245652524        |
ETHIOPIA                  | 1070652520 |
Manufacturer#2            |
N5cVDOq,X8amk                          | 15-443-599-
4524 | special instructions cajole fluffily brave
accounts. deposits sleep.
   9999.94 | Supplier#245652524        |
ETHIOPIA                  | 5645652523 |
Manufacturer#2            |
N5cVDOq,X8amk                          | 15-443-599-
4524 | special instructions cajole fluffily brave
accounts. deposits sleep.
   9999.94 | Supplier#264167361        |
ETHIOPIA                  |  714167356 |
Manufacturer#1            |
zHZnzCwzhJ7V54406rGeP43oIvd f6qhfJ       | 15-297-896-
9474 | es above the accounts haggle carefully even
theodoli
   9999.94 | Supplier#254007586        |
KENYA                     | 3104007565 |
Manufacturer#1            | jBUMxA9ScSw1PBiotPCel
c1WiUlbSAgOy3eA2   | 24-682-340-6526 | unts are
carefully alongside of the sometimes unusual pinto
bean
   9999.94 | Supplier#155385702        |
MOZAMBIQUE                |  305385699 |
Manufacturer#3            | nYYczE
Uyy                            | 26-228-391-7680 |
eep quickly at the requests. final packages are.
ironic packages snooze. reg
   9999.94 | Supplier#158917143        |
MOZAMBIQUE                | 4133917103 |
Manufacturer#2            | RssA8WOmixE8q rgXhbRjRykc
dvbCJNvMUY     | 26-244-461-9689 | hely special ideas
wake blithely above the furiously
   9999.94 | Supplier#252388972        |
MOZAMBIQUE                | 1527388956 |
Manufacturer#3            |
LTLwE8n66rSBvLulMkX                    | 26-358-881-
8892 | aggle. quickly regular packages above the
dependencies sleep carefully regular, final
theodolites. s
   9999.94 | Supplier#252388972        |
MOZAMBIQUE                | 3852388971 |
Manufacturer#5            |
LTLwE8n66rSBvLulMkX                    | 26-358-881-
8892 | aggle. quickly regular packages above the
dependencies sleep carefully regular, final
theodolites. s
   9999.94 | Supplier#283765549        |
MOZAMBIQUE                | 1408765544 |
Manufacturer#4            |
UpHeSLQ2KOOOJi7suw28honuagCxdHSJ7RjDfvDE | 26-868-107-
7796 | ng accounts boost slyly above the furiously
special warthogs. slyly regular accounts boost care
   9999.93 | Supplier#124574429        |
ALGERIA                   | 3574574406 |
Manufacturer#5            |
hL1BzN2Cmn55aCNB8W7                    | 10-994-787-
8847 | old, ironic ideas above the unusual, ironic
platelets use
   9999.93 | Supplier#124574429        |
ALGERIA                   | 5224574428 |
Manufacturer#3            |
hL1BzN2Cmn55aCNB8W7                    | 10-994-787-
8847 | old, ironic ideas above the unusual, ironic
platelets use
   9999.93 | Supplier#210663278        |
ALGERIA                   | 4710663277 |
Manufacturer#1            |
f3uPyisy,0IUJIIWNNS4mCmXnfLPmFs          | 10-827-977-
4996 | according to the carefully ironic p
   9999.93 | Supplier#137292852        |
ETHIOPIA                  | 5012292803 |
Manufacturer#5            |
vkmFjXYrbmuZytlJt,b2Woh7r,u Ci,          | 15-991-925-
6451 | ckly! regular requests dazzle above the
   9999.93 | Supplier#063443304        |
KENYA                     | 4863443303 |
Manufacturer#3            |
TT36suLbCW,9z0K                        | 24-969-480-
5231 | atelets. deposits wake. slyly even pearls
haggle slyly. quickly fi
(100 rows)

echo ============================
echo Query 3
echo ============================

select
      l_orderkey,
      sum(l_extendedprice * (1 - l_discount)) as
revenue,
      o_orderdate,
      o_shippriority
from
      customer,
      orders,
      lineitem
where
      c_mktsegment = 'BUILDING'
      and c_custkey = o_custkey
      and l_orderkey = o_orderkey
      and o_orderdate < date '1995-03-04'
      and l_shipdate > date '1995-03-04'
group by
      l_orderkey,
      o_orderdate,
      o_shippriority
order by
      revenue desc,
      o_orderdate
limit 10;
 l_orderkey |  revenue   | o_orderdate |
o_shippriority
------------+------------+-------------
+----------------
 1341849551 | 2021558.88 | 1995-02-24  |
0
 1319591884 | 1928608.92 | 1995-02-19  |
0
 1399675852 | 1894536.40 | 1995-02-28  |
0
 1287657800 | 1849887.84 | 1995-01-31  |
0
 1274502125 | 1847279.16 | 1995-02-28  |
0
 1396952682 | 1807874.40 | 1995-02-24  |
0
 1375322536 | 1786128.48 | 1995-02-10  |
0
 1308451533 | 1780338.52 | 1995-02-28  |
0
 1419464559 | 1778878.48 | 1995-02-08  |
0
 1290973133 | 1776426.00 | 1995-03-03  |
0
(10 rows)

echo ============================
echo Query 4
```

```
echo ============================

select
      o_orderpriority,
      count(*) as order_count
from
      orders
where
      o_orderdate >= date '1993-08-01'
      and o_orderdate < cast(date '1993-08-01' +
interval '3 months' as date)
      and exists (
            select
                  *
            from
                  lineitem
            where
                  l_orderkey = o_orderkey
                  and l_commitdate < l_receiptdate
      )
group by
      o_orderpriority
order by
      o_orderpriority;
 o_orderpriority | order_count
-----------------+-------------
 1-URGENT        |   315888815
 2-HIGH          |   315889044
 3-MEDIUM        |   315902917
 4-NOT SPECIFIED |   315916095
 5-LOW           |   315860493
(5 rows)

echo ============================
echo Query 5
echo ============================

select
      n_name,
      sum(l_extendedprice * (1 - l_discount)) as
revenue
from
      customer,
      orders,
      lineitem,
      supplier,
      nation,
      region
where
      c_custkey = o_custkey
      and l_orderkey = o_orderkey
      and l_suppkey = s_suppkey
      and c_nationkey = s_nationkey
      and s_nationkey = n_nationkey
      and n_regionkey = r_regionkey
      and r_name = 'AFRICA'
      and o_orderdate >= date '1993-01-01'
      and o_orderdate < cast(date '1993-01-01' +
interval '1 year' as date)
group by
      n_name
order by
      revenue desc;
          n_name           |     revenue
---------------------------+------------------
 MOROCCO                   | 1593099863972.89
 ETHIOPIA                  | 1591991165118.10
 MOZAMBIQUE                | 1591951399693.31
 KENYA                     | 1590997857798.10
 ALGERIA                   | 1590681940614.23
(5 rows)

echo ============================
echo Query 6
echo ============================

select
      sum(l_extendedprice * l_discount) as revenue
from
      lineitem
where
      l_shipdate >= date '1993-01-01'
      and l_shipdate < cast(date '1993-01-01' +
interval '1 year' as date)
      and l_discount between 0.06 - 0.01 and 0.06 +
0.01
      and l_quantity < 24;
     revenue
------------------
 3701836122560.55
(1 row)

echo ============================
echo Query 7
echo ============================

select
      supp_nation,
      cust_nation,
      l_year,
      sum(volume) as revenue
from
      (
            select
                  n1.n_name as supp_nation,
                  n2.n_name as cust_nation,
                  datepart(year, l_shipdate) as
l_year,
                  l_extendedprice * (1 - l_discount)
as volume
            from
                  supplier,
                  lineitem,
                  orders,
                  customer,
                  nation n1,
                  nation n2
            where
                  s_suppkey = l_suppkey
                  and o_orderkey = l_orderkey
                  and c_custkey = o_custkey
                  and s_nationkey = n1.n_nationkey
                  and c_nationkey = n2.n_nationkey
                  and (
                        (n1.n_name = 'KENYA' and
n2.n_name = 'EGYPT')
                        or (n1.n_name = 'EGYPT' and
n2.n_name = 'KENYA')
                  )
                  and l_shipdate between date '1995-
01-01' and date '1996-12-31'
      ) as shipping
group by
      supp_nation,
      cust_nation,
      l_year
order by
      supp_nation,
      cust_nation,
      l_year;
        supp_nation        |        cust_nation
| l_year |     revenue
---------------------------
+---------------------------+--------
+------------------
 EGYPT                     | KENYA
|   1995 | 1591763080055.98
 EGYPT                     | KENYA
|   1996 | 1596389306558.06
 KENYA                     | EGYPT
|   1995 | 1591404537759.62
 KENYA                     | EGYPT
|   1996 | 1595904762064.92
(4 rows)

echo ============================
```

```
echo Query 8
echo ============================

select
      o_year,
      sum(case
            when nation = 'EGYPT' then volume
            else 0
      end) / sum(volume) as mkt_share
from
      (
            select
                  datepart(year,o_orderdate) as
o_year,
                  l_extendedprice * (1 - l_discount)
as volume,
                  n2.n_name as nation
            from
                  part,
                  supplier,
                  lineitem,
                  orders,
                  customer,
                  nation n1,
                  nation n2,
                  region
            where
                  p_partkey = l_partkey
                  and s_suppkey = l_suppkey
                  and l_orderkey = o_orderkey
                  and o_custkey = c_custkey
                  and c_nationkey = n1.n_nationkey
                  and n1.n_regionkey = r_regionkey
                  and r_name = 'MIDDLE EAST'
                  and s_nationkey = n2.n_nationkey
                  and o_orderdate between date '1995-
01-01' and date '1996-12-31'
                  and p_type = 'LARGE BURNISHED TIN'
      ) as all_nations
group by
      o_year
order by
      o_year;
 o_year | mkt_share
--------+-----------
   1995 |      0.04
   1996 |      0.03
(2 rows)

echo ============================
echo Query 9
echo ============================

select
      nation,
      o_year,
      sum(amount) as sum_profit
from
      (
            select
                  n_name as nation,
                  datepart(year,o_orderdate) as
o_year,
                  l_extendedprice * (1 - l_discount) -
ps_supplycost * l_quantity as amount
            from
                  part,
                  supplier,
                  lineitem,
                  partsupp,
                  orders,
                  nation
            where
                  s_suppkey = l_suppkey
                  and ps_suppkey = l_suppkey
                  and ps_partkey = l_partkey
                  and p_partkey = l_partkey
                  and o_orderkey = l_orderkey
                  and s_nationkey = n_nationkey
                  and p_name like '%medium%'
      ) as profit
group by
      nation,
      o_year
order by
      nation,
      o_year desc;
      nation           | o_year |    sum_profit

---------------------------+--------
+------------------
 ALGERIA                  |   1998 |  792240617484.39
 ALGERIA                  |   1997 | 1351890892862.41
 ALGERIA                  |   1996 | 1355598563838.71
 ALGERIA                  |   1995 | 1352288323169.96
 ALGERIA                  |   1994 | 1351810429315.91
 ALGERIA                  |   1993 | 1351888630174.54
 ALGERIA                  |   1992 | 1355676756624.25
 ARGENTINA                |   1998 |  791991441182.39
 ARGENTINA                |   1997 | 1351879721133.51
 ARGENTINA                |   1996 | 1355061295474.92
 ARGENTINA                |   1995 | 1351121034827.54
 ARGENTINA                |   1994 | 1351216978449.56
 ARGENTINA                |   1993 | 1351382704111.74
 ARGENTINA                |   1992 | 1354601515781.89
 BRAZIL                   |   1998 |  792743766323.79
 BRAZIL                   |   1997 | 1351706558871.88
 BRAZIL                   |   1996 | 1355485536267.03
 BRAZIL                   |   1995 | 1352178269832.16
 BRAZIL                   |   1994 | 1352453481672.53
 BRAZIL                   |   1993 | 1352338569135.05
 BRAZIL                   |   1992 | 1355557212704.35
 CANADA                   |   1998 |  792770436738.66
 CANADA                   |   1997 | 1351983381122.56
 CANADA                   |   1996 | 1355838482333.46
 CANADA                   |   1995 | 1351991014921.54
 CANADA                   |   1994 | 1351988339672.62
 CANADA                   |   1993 | 1351824105739.56
 CANADA                   |   1992 | 1355694633257.69
 CHINA                    |   1998 |  792644689406.36
 CHINA                    |   1997 | 1351352703142.55
 CHINA                    |   1996 | 1355047535200.64
 CHINA                    |   1995 | 1351152346744.13
 CHINA                    |   1994 | 1351693582205.54
 CHINA                    |   1993 | 1351135065877.37
 CHINA                    |   1992 | 1354835769944.13
 EGYPT                    |   1998 |  792337081643.57
 EGYPT                    |   1997 | 1351842323222.51
 EGYPT                    |   1996 | 1355374553522.59
 EGYPT                    |   1995 | 1351656592140.42
 EGYPT                    |   1994 | 1351572941274.36
 EGYPT                    |   1993 | 1351483568680.28
 EGYPT                    |   1992 | 1355155798853.81
 ETHIOPIA                 |   1998 |  792261245334.99
 ETHIOPIA                 |   1997 | 1351614491583.46
 ETHIOPIA                 |   1996 | 1355169400400.77
 ETHIOPIA                 |   1995 | 1351131448599.77
 ETHIOPIA                 |   1994 | 1351269407432.38
 ETHIOPIA                 |   1993 | 1351657621907.52
 ETHIOPIA                 |   1992 | 1354781055740.41
 FRANCE                   |   1998 |  792157126363.15
 FRANCE                   |   1997 | 1351558646423.24
 FRANCE                   |   1996 | 1355393805815.12
 FRANCE                   |   1995 | 1351660928431.10
 FRANCE                   |   1994 | 1351257178338.27
 FRANCE                   |   1993 | 1351795648195.10
 FRANCE                   |   1992 | 1354927032046.84
 GERMANY                  |   1998 |  792434462467.50
 GERMANY                  |   1997 | 1351483926921.50
 GERMANY                  |   1996 | 1355553671837.71
 GERMANY                  |   1995 | 1351814536717.32
 GERMANY                  |   1994 | 1351096869105.32
 GERMANY                  |   1993 | 1351673536459.96
 GERMANY                  |   1992 | 1355163288979.37
 INDIA                    |   1998 |  792477718322.09
 INDIA                    |   1997 | 1352327428571.46
```

```
INDIA                     |   1996 | 1356051880969.37
INDIA                     |   1995 | 1352130934946.79
INDIA                     |   1994 | 1352307259217.32
INDIA                     |   1993 | 1352188103121.33
INDIA                     |   1992 | 1355796137997.48
INDONESIA                 |   1998 |  792240285929.82
INDONESIA                 |   1997 | 1351631413989.67
INDONESIA                 |   1996 | 1354905902066.33
INDONESIA                 |   1995 | 1351182882753.51
INDONESIA                 |   1994 | 1351615093216.44
INDONESIA                 |   1993 | 1351332364577.87
INDONESIA                 |   1992 | 1354810632708.11
IRAN                      |   1998 |  792268820093.02
IRAN                      |   1997 | 1351090208877.34
IRAN                      |   1996 | 1355245768922.46
IRAN                      |   1995 | 1351215257391.70
IRAN                      |   1994 | 1351438362942.45
IRAN                      |   1993 | 1351126872454.13
IRAN                      |   1992 | 1354817460477.31
IRAQ                      |   1998 |  792614954418.53
IRAQ                      |   1997 | 1351868999476.03
IRAQ                      |   1996 | 1355908755716.02
IRAQ                      |   1995 | 1352225342115.56
IRAQ                      |   1994 | 1351578113113.70
IRAQ                      |   1993 | 1351943041881.06
IRAQ                      |   1992 | 1355340930768.05
JAPAN                     |   1998 |  792346935092.90
JAPAN                     |   1997 | 1351788037704.71
JAPAN                     |   1996 | 1355516023892.01
JAPAN                     |   1995 | 1351896463984.31
JAPAN                     |   1994 | 1351828718359.55
JAPAN                     |   1993 | 1351861118121.98
JAPAN                     |   1992 | 1355259927374.59
JORDAN                    |   1998 |  793104468761.73
JORDAN                    |   1997 | 1352931045905.64
JORDAN                    |   1996 | 1356229625101.35
JORDAN                    |   1995 | 1352257794523.64
JORDAN                    |   1994 | 1352720101757.20
JORDAN                    |   1993 | 1353064206333.09
JORDAN                    |   1992 | 1356412997366.02
KENYA                     |   1998 |  792348233166.99
KENYA                     |   1997 | 1351727225871.80
KENYA                     |   1996 | 1355590243791.44
KENYA                     |   1995 | 1352119617868.20
KENYA                     |   1994 | 1351625478245.33
KENYA                     |   1993 | 1351469212904.48
KENYA                     |   1992 | 1355622861799.16
MOROCCO                   |   1998 |  793365211242.63
MOROCCO                   |   1997 | 1352912213599.54
MOROCCO                   |   1996 | 1356989768505.57
MOROCCO                   |   1995 | 1353300321385.54
MOROCCO                   |   1994 | 1353122781800.32
MOROCCO                   |   1993 | 1352830651897.98
MOROCCO                   |   1992 | 1356364171938.01
MOZAMBIQUE                |   1998 |  792525642332.72
MOZAMBIQUE                |   1997 | 1351677136816.50
MOZAMBIQUE                |   1996 | 1355786715936.95
MOZAMBIQUE                |   1995 | 1351945873710.02
MOZAMBIQUE                |   1994 | 1351223362444.02
MOZAMBIQUE                |   1993 | 1351547812962.75
MOZAMBIQUE                |   1992 | 1355417353440.85
PERU                      |   1998 |  792426558136.27
PERU                      |   1997 | 1351318288831.86
PERU                      |   1996 | 1355283156105.05
PERU                      |   1995 | 1350961390444.65
PERU                      |   1994 | 1351133796160.57
PERU                      |   1993 | 1351307059287.40
PERU                      |   1992 | 1354705336338.16
ROMANIA                   |   1998 |  792499710132.77
ROMANIA                   |   1997 | 1351919063649.53
ROMANIA                   |   1996 | 1355942306415.03
ROMANIA                   |   1995 | 1351962865763.85
ROMANIA                   |   1994 | 1352013487036.11
ROMANIA                   |   1993 | 1352623378611.17
ROMANIA                   |   1992 | 1355947484137.55
RUSSIA                    |   1998 |  792459318527.14
RUSSIA                    |   1997 | 1351252152355.77
RUSSIA                    |   1996 | 1355005726086.70
RUSSIA                    |   1995 | 1351836926340.93
RUSSIA                    |   1994 | 1351184302262.19
RUSSIA                    |   1993 | 1351225339437.39
RUSSIA                    |   1992 | 1354887329629.97
SAUDI ARABIA              |   1998 |  792780420339.81
SAUDI ARABIA              |   1997 | 1352622833723.35
SAUDI ARABIA              |   1996 | 1356526197014.13
SAUDI ARABIA              |   1995 | 1352579993173.39
SAUDI ARABIA              |   1994 | 1352430610217.92
SAUDI ARABIA              |   1993 | 1351794274116.25
SAUDI ARABIA              |   1992 | 1356189150242.92
UNITED KINGDOM            |   1998 |  792694625297.78
UNITED KINGDOM            |   1997 | 1352504778350.53
UNITED KINGDOM            |   1996 | 1356207633072.17
UNITED KINGDOM            |   1995 | 1351899156367.35
UNITED KINGDOM            |   1994 | 1352117397125.46
UNITED KINGDOM            |   1993 | 1351714738436.04
UNITED KINGDOM            |   1992 | 1355631151926.83
UNITED STATES             |   1998 |  792458430849.87
UNITED STATES             |   1997 | 1351267621324.95
UNITED STATES             |   1996 | 1354916487803.60
UNITED STATES             |   1995 | 1351609540948.75
UNITED STATES             |   1994 | 1351322364475.26
UNITED STATES             |   1993 | 1351901030234.96
UNITED STATES             |   1992 | 1355018261122.78
VIETNAM                   |   1998 |  792690823317.80
VIETNAM                   |   1997 | 1351899725106.07
VIETNAM                   |   1996 | 1355990608094.20
VIETNAM                   |   1995 | 1352130591900.71
VIETNAM                   |   1994 | 1351956993586.00
VIETNAM                   |   1993 | 1352162018185.64
VIETNAM                   |   1992 | 1355974501074.81
(175 rows)

echo ============================
echo Query 10
echo ============================

select
      c_custkey,
      c_name,
      sum(l_extendedprice * (1 - l_discount)) as
revenue,
      c_acctbal,
      n_name,
      c_address,
      c_phone,
      c_comment
from
      customer,
      orders,
      lineitem,
      nation
where
      c_custkey = o_custkey
      and l_orderkey = o_orderkey
      and o_orderdate >= date '1994-09-01'
      and o_orderdate < cast(date '1994-09-01' +
interval '3 months' as date)
      and l_returnflag = 'R'
      and c_nationkey = n_nationkey
group by
      c_custkey,
      c_name,
      c_acctbal,
      c_phone,
      n_name,
      c_address,
      c_comment
order by
      revenue desc
limit 20;
 c_custkey  |       c_name        |  revenue   |
c_acctbal |          n_name           |
c_address                 |     c_phone
|
 c_comment
```

```
------------+--------------------+------------
+-----------+----------------------------
+-----------------------------------------
+-----------------
+-----------------------------------------------------
-----------------------------------------------------
-----
 1259486104 | Customer#1259486104 | 1970668.02 |
1389.65 | MOROCCO                  |
,XHeN9D2,NOTVa9jAlJip0NA6tK           | 25-118-504-
2296 |  ironic, unusual accounts. blithely fina
 3274492162 | Customer#3274492162 | 1968542.34 |
6084.03 | MOZAMBIQUE               |
hBoZQwyBIhfvFzuynKFCPKbY8             | 26-394-953-
6793 | riously inside the furiously furious deposits.
slyly ironic packages sleep. reg
  581267440 | Customer#581267440  | 1967580.29 |
8403.05 | RUSSIA                   |
KVoL9ntBv2Q4BHTlGY27FvU               | 32-752-388-
3637 |  quickly regular gifts according to the quickly
regular dolp
 3398291194 | Customer#3398291194 | 1952364.89 |
1694.32 | CHINA                    | kweIOG75XEEXu
wxN5Wc1c               | 28-469-120-9985 | sly silent
deposits above the furiously even acco
 4091948086 | Customer#4091948086 | 1927635.26 |
813.37 | ETHIOPIA                 |
eddNKhR9FEyX80YhWdIKXb6               | 15-970-130-
5043 | cajole slyly. unusual, furious excuses use
furiously after the furiously unusual packages. fina
 3817356148 | Customer#3817356148 | 1904886.79 |
3166.11 | ALGERIA                  |
DeQmCxPgPsXY                          | 10-739-610-
3795 | . unusual braids haggle boldly regular, bold
pinto beans.
 3714341050 | Customer#3714341050 | 1884445.46 |
9149.15 | GERMANY                  |
EoOc3YR7an5RHlzgwHOnjS7H87I85WTWWL1tK | 17-997-634-
1048 | above the requests. permanently final ideas are
carefully. slyly bold ideas cajo
 4111155598 | Customer#4111155598 | 1866329.93 |
325.49 | JAPAN                    |
kDSgGmJT0ry7hL5fVev6R,q4PsAOdQE2      | 22-774-661-
6063 | lithely even requests cajole carefully blithely
final dolphins. slyly ironic accounts alo
 1352335189 | Customer#1352335189 | 1858501.01 |
268.89 | IRAN                     |
OGLpuRlV,ZSlZM1TusF                   | 20-650-433-
9718 | g accounts sleep carefully at the ironic
somas-- quietly regular requests haggle slyly slyly
ironic orb
  738425140 | Customer#738425140  | 1851675.83 |
5575.48 | PERU                     | w
8BgYFAkeYfKL0                         | 27-338-518-9622
| ideas use quickly. theodolites mold carefully
fluffily final packages. stealthily ironic foxes wake
blithely a
 2076814082 | Customer#2076814082 | 1822013.48 |
799.85 | INDONESIA                | 7RInShjq8QMWRNOx
6TL6S                | 19-319-494-2302 | yly between
the waters. slyly final ideas according to the
blithely ironic packa
 3249649679 | Customer#3249649679 | 1821893.12 |
8606.82 | INDONESIA                | duA
FFTrHI3TVeLcTh7uweSC9E2Yn3Cw      | 19-345-234-9073 |
the furiously pending accounts cajole carefully bold
 1166662445 | Customer#1166662445 | 1809946.16 |
8899.40 | KENYA                    | PEUb9fBhhksje
4QweD4w4dWvpFlAPPOJ     | 24-993-170-6592 |  wake
blithely about the blithely regular theodolites.
silent packages are
  826915426 | Customer#826915426  | 1807644.25 |
2409.75 | IRAQ                     |
GMetfkHij5M,LhdOPSpPjA9jRo9vUuJ       | 21-788-825-
9795 |  final deposits at the final deposits haggle s
 3238127386 | Customer#3238127386 | 1802310.44 |
375.60 | CANADA                   |
D1TDV5xLg4wIXPce2dO sKZz2Jwn          | 13-281-145-
6834 | about the carefully express packages. furiously
final instructions use quickly final packages.
packages ar
 2351860097 | Customer#2351860097 | 1790061.11 |
7472.90 | ROMANIA                  |
ZpYeZSxHvaL4JB9FE                     | 29-849-228-
4057 | accounts. stealthy, pending accounts haggle
across the daring theodolites. s
 1021954033 | Customer#1021954033 | 1784130.43 |
8187.03 | KENYA                    |
g2rO1aTE0Cyj                          | 24-200-103-
6964 | he regular deposits! fluffily special accou
 2774935453 | Customer#2774935453 | 1759196.97 |
6368.06 | IRAN                     |
TzbiaDlRpHTfBcTPCggPBL6               | 20-906-858-
3744 | the sentiments. regular packages shall have t
 2005854604 | Customer#2005854604 | 1758124.19 |
5364.71 | VIETNAM                  | SGBmOhr59Q6SVK
tdna hZaltXnyPyqeUJ    | 31-266-941-9365 | s. regular
requests boost across the carefully
 1162395220 | Customer#1162395220 | 1757901.20 |
9170.35 | KENYA                    |
WERUluoY,Js4Bg                    | 24-760-303-5677
| s. quickly express accounts boost through the even,
bold warhorses? final,
(20 rows)

echo ============================
echo Query 11
echo ============================

select
      ps_partkey,
      sum(ps_supplycost * ps_availqty) as value
from
      partsupp,
      supplier,
      nation
where
      ps_suppkey = s_suppkey
      and s_nationkey = n_nationkey
      and n_name = 'ETHIOPIA'
group by
      ps_partkey having
            sum(ps_supplycost * ps_availqty) > (
                  select
                        sum(ps_supplycost *
ps_availqty) * 0.0000000033
                  from
                        partsupp,
                        supplier,
                        nation
                  where
                        ps_suppkey = s_suppkey
                        and s_nationkey = n_nationkey
                        and n_name = 'ETHIOPIA'
            )
order by
      value desc;
 ps_partkey |    value
------------+-------------
 5294416577 | 27889262.56
 1152592518 | 27503090.54
 2570364790 | 27060605.29
 1352013439 | 26946180.66
 1321553167 | 26808760.40
 5232954542 | 26669770.13
 3118577918 | 26572317.30
 4218541120 | 26508932.93
  212864836 | 26451804.66
 3378015457 | 26281019.54
 3401822536 | 26271562.96
 5231143863 | 26177479.42
 5677498874 | 26099858.18
 1293456262 | 26070366.24
 3273794353 | 26065844.48
 3645806184 | 26035069.30
  920858453 | 26017871.91
```

```
1842973456 | 25951061.96
2630662589 | 25939317.12
 410290642 | 25933363.94
2174904701 | 25931435.62
4401475742 | 25850927.50
4151546786 | 25839791.76
2493483722 | 25828328.13
4400727449 | 25784068.42
2492998080 | 25770011.32
5848095815 | 25750013.51
2911906310 | 25741247.98
4453249033 | 25739702.45
4582760487 | 25722584.95
 210587263 | 25719136.52
 443449524 | 25708035.09
5818955758 | 25693488.76
1432628258 | 25679174.80
2618855249 | 25647956.01
1376488647 | 25519978.24
4044642765 | 25449324.84
1033660113 | 25447971.32
  51022804 | 25422224.08
2876215230 | 25396779.30
5523809735 | 25364282.16
5726880494 | 25341803.14
3373614233 | 25304710.08
 321074851 | 25301105.81
5743611838 | 25288674.11
 309986375 | 25279668.56
3402145138 | 25260541.30
4365678740 | 25227025.26
2617285609 | 25197316.45
 399856673 | 25164955.74
 542098378 | 25154835.35
4592619464 | 25118611.90
3558925120 | 25114086.14
3401595155 | 25095689.17
3495657003 | 25089010.70
3029650763 | 25069602.91
1299860736 | 25062309.07
 904624413 | 25061826.45
4582598239 | 25049437.04
5841178231 | 25013511.25
 686150589 | 25001311.63
 113503708 | 24972179.42
1593101560 | 24963171.71
2248728615 | 24949032.20
3256730735 | 24939587.40
 727205015 | 24933211.94
3637415332 | 24921031.54
2057587579 | 24906254.12
4296923661 | 24880202.60
5373008683 | 24866362.68
3368476888 | 24826006.32
3985758803 | 24799474.73
 837761706 | 24790470.16
1474467179 | 24779999.83
1524145325 | 24762189.62
1190786539 | 24760988.38
5539673425 | 24758395.00
2134866410 | 24719457.38
1721977452 | 24711668.39
3174541718 | 24711079.48
2942032317 | 24709000.17
5692797811 | 24697578.82
1713168352 | 24690413.35
5315312246 | 24675340.29
 552818250 | 24649076.83
4825973202 | 24646842.20
5859387675 | 24636771.60
4490296060 | 24582518.71
5637780992 | 24570034.29
 223291468 | 24563233.27
5987346944 | 24554386.83
3300058308 | 24549295.61
 653655206 | 24534597.76
 882696984 | 24533640.01
4822962957 | 24525580.22
3354582910 | 24523058.38
2141611295 | 24520882.52
5926932528 | 24520641.04
2795543925 | 24519728.85
 713295861 | 24515335.04
********** | **********
2885501916 | 7926553.40
4484749416 | 7926553.34
1502355827 | 7926553.25
1288619870 | 7926553.20
4625502717 | 7926553.09
5665158674 | 7926553.08
1941846258 | 7926553.06
3278799759 | 7926552.92
 742001937 | 7926552.92
 153801499 | 7926552.92
2838156057 | 7926552.92
2610598948 | 7926552.84
4070262456 | 7926552.74
  61839523 | 7926552.72
5903176118 | 7926552.67
5011248602 | 7926552.44
3311013588 | 7926552.44
5732885796 | 7926552.04
 529384751 | 7926552.00
5629573271 | 7926551.82
4987635953 | 7926551.82
1229539572 | 7926551.82
1070657323 | 7926551.82
3230938042 | 7926551.82
1899640997 | 7926551.75
1042976658 | 7926551.71
 618826490 | 7926551.64
2989184589 | 7926551.58
 360300962 | 7926551.46
 794937236 | 7926551.44
5041052990 | 7926551.44
 209214787 | 7926551.44
3967311167 | 7926551.44
5904791169 | 7926551.39
4288784017 | 7926551.38
1314638767 | 7926551.23
5460169923 | 7926551.22
1974219759 | 7926551.11
1074946736 | 7926551.10
 333303683 | 7926551.10
4861422445 | 7926551.10
3251935740 | 7926551.04
 992698128 | 7926551.00
1096450105 | 7926550.98
2770565569 | 7926550.94
1696823746 | 7926550.94
 494147533 | 7926550.94
3672168022 | 7926550.94
2801915374 | 7926550.74
3478765096 | 7926550.72
3437618364 | 7926550.72
 687444939 | 7926550.72
 447832693 | 7926550.66
4715352210 | 7926550.60
5811794585 | 7926550.44
1915913358 | 7926550.26
3722759410 | 7926550.24
4027233587 | 7926550.24
4796235757 | 7926550.20
5210263283 | 7926550.09
1493935905 | 7926550.09
2030806817 | 7926550.09
1605525382 | 7926550.05
  94706600 | 7926550.03
1063571728 | 7926550.00
 961300643 | 7926550.00
4586550214 | 7926549.96
 236732501 | 7926549.96
3457957970 | 7926549.96
4772429072 | 7926549.90
1069165654 | 7926549.90
4342258797 | 7926549.88
```

```
 4801494583 | 7926549.88
  174160114 | 7926549.84
 2231512119 | 7926549.84
 3842124854 | 7926549.84
 2192639855 | 7926549.83
 2850706776 | 7926549.81
 2601498621 | 7926549.70
 4283527463 | 7926549.70
  739186653 | 7926549.68
 1863588159 | 7926549.64
 5029231210 | 7926549.60
 2296492006 | 7926549.54
  344143653 | 7926549.48
 2061685470 | 7926549.48
 1942874407 | 7926549.25
 5244861877 | 7926549.12
 5660859623 | 7926549.00
  913916221 | 7926548.90
 3598270779 | 7926548.90
 2935269712 | 7926548.80
 2117461200 | 7926548.76
 3441495196 | 7926548.76
 1622962855 | 7926548.64
 1963536749 | 7926548.63
 5256377832 | 7926548.63
 2066279325 | 7926548.56
 3251304694 | 7926548.48
 2174578120 | 7926548.38
(29945823 rows)

echo ============================
echo Query 12
echo ============================

select
      l_shipmode,
      sum(case
            when o_orderpriority = '1-URGENT'
                  or o_orderpriority = '2-HIGH'
                  then 1
            else 0
      end) as high_line_count,
      sum(case
            when o_orderpriority <> '1-URGENT'
                  and o_orderpriority <> '2-HIGH'
                  then 1
            else 0
      end) as low_line_count
from
      orders,
      lineitem
where
      o_orderkey = l_orderkey
      and l_shipmode in ('SHIP', 'FOB')
      and l_commitdate < l_receiptdate
      and l_shipdate < l_commitdate
      and l_receiptdate >= date '1994-01-01'
      and l_receiptdate < cast(date '1994-01-01' +
interval '1 year' as date)
group by
      l_shipmode
order by
      l_shipmode;
 l_shipmode | high_line_count | low_line_count
------------+-----------------+----------------
 FOB        |       187549730 |      281280466
 SHIP       |       187550423 |      281286260
(2 rows)

echo ============================
echo Query 13
echo ============================

select
      c_count,
      count(*) as custdist
from
      (
            select
                  c_custkey,
                  count(o_orderkey)
            from
                  customer left outer join orders on
                        c_custkey = o_custkey
                        and o_comment not like
'%express%requests%'
            group by
                  c_custkey
      ) as c_orders (c_custkey, c_count)
group by
      c_count
order by
      custdist desc,
      c_count desc;
 c_count | custdist
---------+----------
(0 rows)

echo ============================
echo Query 14
echo ============================

select
      100.00 * sum(case
            when p_type like 'PROMO%'
                  then l_extendedprice * (1 -
l_discount)
            else 0
      end) / sum(l_extendedprice * (1 - l_discount))
as promo_revenue
from
      lineitem,
      part
where
      l_partkey = p_partkey
      and l_shipdate >= date '1994-10-01'
      and l_shipdate < cast(date '1994-10-01' +
interval '1 month' as date);
 promo_revenue
---------------
         -5.20
(1 row)

echo ============================
echo Query 15
echo ============================

create view revenue0 (supplier_no, total_revenue) as
      select
            l_suppkey,
            sum(l_extendedprice * (1 - l_discount))
      from
            lineitem
      where
            l_shipdate >= date '1996-07-01'
            and l_shipdate < cast(date '1996-07-01' +
interval '3 months' as date)
      group by
            l_suppkey;
select
      s_suppkey,
      s_name,
      s_address,
      s_phone,
      total_revenue
from
      supplier,
      revenue0
where
      s_suppkey = supplier_no
      and total_revenue = (
            select
                  max(total_revenue)
            from
                  revenue0
      )
```

```
order by
      s_suppkey;
drop view revenue0;

CREATE VIEW
 s_suppkey |          s_name           |
s_address          |     s_phone     | total_revenue
-----------+---------------------------
+-----------------------------+-----------------
+---------------
 231745920 | Supplier#231745920        |
ayC3cy5U2a2QptYInOEPajlMdLX | 32-719-221-3083 |
2906798.59
(1 row)

DROP VIEW
echo ============================
echo Query 16
echo ============================

select
      p_brand,
      p_type,
      p_size,
      count(distinct ps_suppkey) as supplier_cnt
from
      partsupp,
      part
where
      p_partkey = ps_partkey
      and p_brand <> 'Brand#44'
      and p_type not like 'ECONOMY PLATED%'
      and p_size in (5, 32, 16, 13, 36, 28, 40, 11)
      and ps_suppkey not in (
            select
                  s_suppkey
            from
                  supplier
            where
                  s_comment like '%Customer%Complaints
%'
      )
group by
      p_brand,
      p_type,
      p_size
order by
      supplier_cnt desc,
      p_brand,
      p_type,
      p_size;
  p_brand   |          p_type           | p_size |
supplier_cnt
------------+---------------------------+--------
+--------------
 Brand#51   | STANDARD BRUSHED NICKEL   |     13
|       128662
 Brand#51   | PROMO ANODIZED COPPER     |     40
|       128656
 Brand#55   | LARGE POLISHED STEEL      |      5
|       128638
 Brand#25   | LARGE BURNISHED COPPER    |     13
|       128613
 Brand#34   | LARGE BURNISHED STEEL     |     11
|       128611
 Brand#34   | PROMO BURNISHED NICKEL    |     36
|       128593
 Brand#12   | SMALL BURNISHED NICKEL    |     11
|       128575
 Brand#13   | MEDIUM POLISHED TIN       |     16
|       128574
 Brand#21   | ECONOMY ANODIZED TIN      |      5
|       128573
 Brand#15   | SMALL BURNISHED STEEL     |     40
|       128562
 Brand#31   | ECONOMY BRUSHED COPPER    |     11
|       128561
 Brand#23   | SMALL ANODIZED TIN        |     28
|       128554
 Brand#13   | MEDIUM POLISHED BRASS     |      5
|       128549
 Brand#55   | LARGE POLISHED BRASS      |     13
|       128548
 Brand#32   | PROMO BRUSHED COPPER      |     36
|       128545
 Brand#12   | STANDARD POLISHED BRASS   |     40
|       128536
 Brand#41   | MEDIUM BURNISHED TIN      |      5
|       128536
 Brand#51   | MEDIUM BRUSHED TIN        |     40
|       128536
 Brand#22   | ECONOMY BRUSHED TIN       |     40
|       128532
 Brand#13   | PROMO POLISHED NICKEL     |     28
|       128530
 Brand#33   | MEDIUM BURNISHED NICKEL   |     28
|       128530
 Brand#21   | MEDIUM BRUSHED NICKEL     |     40
|       128523
 Brand#42   | ECONOMY ANODIZED NICKEL   |     32
|       128523
 Brand#31   | SMALL PLATED BRASS        |      5
|       128521
 Brand#33   | STANDARD BURNISHED COPPER |     16
|       128519
 Brand#24   | MEDIUM BRUSHED COPPER     |     13
|       128518
 Brand#53   | ECONOMY BURNISHED BRASS   |     13
|       128515
 Brand#13   | LARGE PLATED STEEL        |      5
|       128511
 Brand#21   | PROMO PLATED STEEL        |     40
|       128510
 Brand#24   | PROMO BURNISHED BRASS     |     16
|       128507
 Brand#41   | STANDARD ANODIZED STEEL   |     13
|       128505
 Brand#12   | SMALL PLATED STEEL        |      5
|       128504
 Brand#24   | PROMO PLATED NICKEL       |     28
|       128504
 Brand#32   | MEDIUM ANODIZED COPPER    |     28
|       128504
 Brand#14   | MEDIUM BURNISHED TIN      |      5
|       128503
 Brand#21   | PROMO POLISHED STEEL      |     11
|       128502
 Brand#11   | ECONOMY ANODIZED STEEL    |     28
|       128499
 Brand#31   | PROMO POLISHED TIN        |     16
|       128499
 Brand#33   | SMALL PLATED STEEL        |     13
|       128493
 Brand#11   | PROMO BURNISHED NICKEL    |     16
|       128492
 Brand#54   | PROMO ANODIZED COPPER     |      5
|       128487
 Brand#41   | STANDARD ANODIZED TIN     |     32
|       128485
 Brand#15   | LARGE ANODIZED NICKEL     |     16
|       128482
 Brand#21   | MEDIUM POLISHED STEEL     |      5
|       128482
 Brand#54   | MEDIUM ANODIZED BRASS     |     16
|       128480
 Brand#25   | STANDARD PLATED COPPER    |     32
|       128479
 Brand#21   | PROMO BRUSHED NICKEL      |      5
|       128478
 Brand#55   | SMALL BRUSHED STEEL       |     16
|       128478
 Brand#55   | LARGE BURNISHED STEEL     |     32
|       128476
 Brand#11   | PROMO BURNISHED TIN       |     40
|       128475
 Brand#45   | SMALL PLATED NICKEL       |     16
```

```
|       128475
 Brand#55   | LARGE BRUSHED BRASS       |     32
|       128469
 Brand#12   | SMALL BRUSHED BRASS       |     16
|       128468
 Brand#23   | ECONOMY POLISHED TIN      |     32
|       128466
 Brand#33   | SMALL POLISHED COPPER     |     16
|       128466
 Brand#54   | PROMO POLISHED NICKEL     |     28
|       128466
 Brand#33   | PROMO BRUSHED STEEL       |      5
|       128462
 Brand#54   | STANDARD BRUSHED BRASS    |     40
|       128462
 Brand#25   | MEDIUM ANODIZED NICKEL    |     40
|       128458
 Brand#54   | STANDARD BRUSHED TIN      |      5
|       128458
 Brand#25   | SMALL POLISHED COPPER     |     32
|       128457
 Brand#32   | ECONOMY ANODIZED NICKEL   |     11
|       128457
 Brand#52   | PROMO PLATED NICKEL       |     11
|       128457
 Brand#13   | PROMO BRUSHED STEEL       |     16
|       128455
 Brand#23   | ECONOMY BRUSHED STEEL     |     13
|       128453
 Brand#25   | SMALL PLATED TIN          |     13
|       128453
 Brand#41   | LARGE POLISHED TIN        |     16
|       128453
 Brand#11   | PROMO BRUSHED BRASS       |     40
|       128452
 Brand#23   | PROMO ANODIZED TIN        |     36
|       128452
 Brand#35   | MEDIUM BRUSHED TIN        |      5
|       128452
 Brand#54   | LARGE BRUSHED TIN         |     36
|       128452
 Brand#32   | LARGE ANODIZED TIN        |     16
|       128451
 Brand#15   | PROMO POLISHED STEEL      |     13
|       128448
 Brand#45   | ECONOMY ANODIZED BRASS    |     40
|       128448
 Brand#42   | MEDIUM POLISHED COPPER    |     32
|       128447
 Brand#51   | SMALL ANODIZED TIN        |     40
|       128447
 Brand#11   | SMALL BURNISHED STEEL     |     13
|       128445
 Brand#22   | MEDIUM PLATED NICKEL      |     13
|       128444
 Brand#34   | STANDARD BURNISHED COPPER |     28
|       128444
 Brand#25   | SMALL BURNISHED TIN       |     28
|       128443
 Brand#35   | LARGE PLATED NICKEL       |     28
|       128443
 Brand#35   | SMALL POLISHED BRASS      |     32
|       128442
 Brand#34   | SMALL PLATED COPPER       |     28
|       128440
 Brand#41   | ECONOMY POLISHED COPPER   |      5
|       128440
 Brand#14   | ECONOMY BRUSHED STEEL     |     16
|       128439
 Brand#41   | PROMO BRUSHED NICKEL      |     28
|       128438
 Brand#42   | ECONOMY ANODIZED TIN      |     11
|       128438
 Brand#45   | PROMO BRUSHED BRASS       |     40
|       128437
 Brand#32   | PROMO BRUSHED BRASS       |      5
|       128436
 Brand#11   | ECONOMY POLISHED TIN      |     11
|       128435
 Brand#43   | ECONOMY POLISHED TIN      |     36
|       128435
 Brand#32   | MEDIUM PLATED TIN         |     13
|       128434
 Brand#5
```

Appendix D. Seed and Query Substitution Parameters

This Appendix contains Seed values and substitution parameters for each stream

Seed Values

```
stream0  606202239
stream1  606202240
stream2  606202241
stream3  606202242
stream4  606202243
stream5  606202244
stream6  606202245
stream7  606202246
stream8  606202247
stream9  606202248
stream10 606202249
```

Query Parameters

```
========================
stream0 params
========================
14      1995-12-01
2       33      BRASS   EUROPE
9       cornflower
20      lace    1995-01-01      ETHIOPIA
6       1997-01-01      0.05    25
17      Brand#23        LG BOX
18      314
8       RUSSIA EUROPE SMALL ANODIZED STEEL
21      JORDAN
13      pending         accounts
3       AUTOMOBILE      1995-03-23
22      15      19      24      12      16      27      30
16      Brand#22        LARGE PLATED    13      1
3       25      18      17      28      16
4       1996-08-01
11      ROMANIA         0.0000000033
15      1996-07-01
1       106
10      1994-06-01
19      Brand#23        Brand#32        Brand#41
1       13      28
5       EUROPE 1997-01-01
7       JAPAN   RUSSIA
12      MAIL    RAIL    1995-01-01


========================
stream1 params
========================
21      JORDAN
3       AUTOMOBILE      1995-03-23
18      314
5       EUROPE 1997-01-01
```

```
11      ROMANIA       0.0000000033
7       JAPAN  RUSSIA
6       1997-01-01    0.05   25
20      lace   1995-01-01    ETHIOPIA
17      Brand#23      LG BOX
12      MAIL   RAIL   1995-01-01
16      Brand#22      LARGE PLATED  13      1
3       25      18      17      28      16
15      1996-07-01
13      pending       accounts
10      1994-06-01
2       33      BRASS  EUROPE
8       RUSSIA EUROPE SMALL ANODIZED STEEL
14      1995-12-01
19      Brand#23      Brand#32      Brand#41
1       13      28
9       cornflower
22      15      19      24      12      16      27      30
1       106
4       1996-08-01


========================
stream2 params
========================
6       1997-01-01    0.05   25
17      Brand#23      LG BOX
14      1995-12-01
16      Brand#22      LARGE PLATED  13      1
3       25      18      17      28      16
19      Brand#23      Brand#32      Brand#41
1       13      28
10      1994-06-01
9       cornflower
2       33      BRASS  EUROPE
15      1996-07-01
8       RUSSIA EUROPE SMALL ANODIZED STEEL
5       EUROPE 1997-01-01
22      15      19      24      12      16      27      30
12      MAIL   RAIL   1995-01-01
7       JAPAN  RUSSIA
13      pending       accounts
18      314
1       106
4       1996-08-01
20      lace   1995-01-01    ETHIOPIA
3       AUTOMOBILE    1995-03-23
11      ROMANIA       0.0000000033
21      JORDAN


========================
stream3 params
========================
8       RUSSIA EUROPE SMALL ANODIZED STEEL
5       EUROPE 1997-01-01
4       1996-08-01
6       1997-01-01    0.05   25
17      Brand#23      LG BOX
7       JAPAN  RUSSIA
1       106
18      314
22      15      19      24      12      16      27      30
14      1995-12-01
9       cornflower
10      1994-06-01
15      1996-07-01
11      ROMANIA       0.0000000033
20      lace   1995-01-01    ETHIOPIA
2       33      BRASS  EUROPE
21      JORDAN
19      Brand#23      Brand#32      Brand#41
1       13      28
13      pending       accounts
16      Brand#22      LARGE PLATED  13      1
3       25      18      17      28      16
12      MAIL   RAIL   1995-01-01
3       AUTOMOBILE    1995-03-23


========================
stream4 params
========================
5       EUROPE 1997-01-01
21      JORDAN
14      1995-12-01
19      Brand#23      Brand#32      Brand#41
1       13      28
15      1996-07-01
17      Brand#23      LG BOX
12      MAIL   RAIL   1995-01-01
6       1997-01-01    0.05   25
4       1996-08-01
9       cornflower
8       RUSSIA EUROPE SMALL ANODIZED STEEL
16      Brand#22      LARGE PLATED  13      1
3       25      18      17      28      16
11      ROMANIA       0.0000000033
2       33      BRASS  EUROPE
10      1994-06-01
18      314
1       106
13      pending       accounts
7       JAPAN  RUSSIA
22      15      19      24      12      16      27      30
3       AUTOMOBILE    1995-03-23
20      lace   1995-01-01    ETHIOPIA


========================
stream5 params
========================
21      JORDAN
15      1996-07-01
4       1996-08-01
6       1997-01-01    0.05   25
7       JAPAN  RUSSIA
16      Brand#22      LARGE PLATED  13      1
3       25      18      17      28      16
19      Brand#23      Brand#32      Brand#41
1       13      28
18      314
14      1995-12-01
22      15      19      24      12      16      27      30
11      ROMANIA       0.0000000033
13      pending       accounts
3       AUTOMOBILE    1995-03-23
1       106
2       33      BRASS  EUROPE
5       EUROPE 1997-01-01
8       RUSSIA EUROPE SMALL ANODIZED STEEL
20      lace   1995-01-01    ETHIOPIA
12      MAIL   RAIL   1995-01-01
17      Brand#23      LG BOX
10      1994-06-01
9       cornflower


========================
stream6 params
========================
10      1994-06-01
3       AUTOMOBILE    1995-03-23
15      1996-07-01
13      pending       accounts
6       1997-01-01    0.05   25
8       RUSSIA EUROPE SMALL ANODIZED STEEL
9       cornflower
7       JAPAN  RUSSIA
4       1996-08-01
11      ROMANIA       0.0000000033
22      15      19      24      12      16      27      30
18      314
12      MAIL   RAIL   1995-01-01
1       106
5       EUROPE 1997-01-01
```

```
16      Brand#22        LARGE PLATED  13      1
3       25      18      17      28      16
2       33      BRASS   EUROPE
14      1995-12-01
19      Brand#23        Brand#32        Brand#41
1       13      28
20      lace    1995-01-01      ETHIOPIA
17      Brand#23        LG BOX
21      JORDAN


========================
stream7 params
========================
18      314
8       RUSSIA EUROPE SMALL ANODIZED STEEL
20      lace    1995-01-01      ETHIOPIA
21      JORDAN
2       33      BRASS   EUROPE
4       1996-08-01
22      15      19      24      12      16      27      30
17      Brand#23        LG BOX
1       106
11      ROMANIA         0.0000000033
9       cornflower
19      Brand#23        Brand#32        Brand#41
1       13      28
3       AUTOMOBILE      1995-03-23
13      pending         accounts
5       EUROPE 1997-01-01
7       JAPAN   RUSSIA
10      1994-06-01
16      Brand#22        LARGE PLATED  13      1
3       25      18      17      28      16
6       1997-01-01      0.05    25
14      1995-12-01
15      1996-07-01
12      MAIL    RAIL    1995-01-01


========================
stream8 params
========================
19      Brand#23        Brand#32        Brand#41
1       13      28
1       106
15      1996-07-01
17      Brand#23        LG BOX
5       EUROPE 1997-01-01
8       RUSSIA EUROPE SMALL ANODIZED STEEL
9       cornflower
12      MAIL    RAIL    1995-01-01
14      1995-12-01
7       JAPAN   RUSSIA
4       1996-08-01
3       AUTOMOBILE      1995-03-23
20      lace    1995-01-01      ETHIOPIA
16      Brand#22        LARGE PLATED  13      1
3       25      18      17      28      16
6       1997-01-01      0.05    25
22      15      19      24      12      16      27      30
10      1994-06-01
13      pending         accounts
2       33      BRASS   EUROPE
21      JORDAN
18      314
11      ROMANIA         0.0000000033


========================
stream9 params
========================
8       RUSSIA EUROPE SMALL ANODIZED STEEL
13      pending         accounts
2       33      BRASS   EUROPE
20      lace    1995-01-01      ETHIOPIA
17      Brand#23        LG BOX
3       AUTOMOBILE      1995-03-23
6       1997-01-01      0.05    25
21      JORDAN
18      314
11      ROMANIA         0.0000000033
19      Brand#23        Brand#32        Brand#41
1       13      28
10      1994-06-01
15      1996-07-01
4       1996-08-01
22      15      19      24      12      16      27      30
1       106
7       JAPAN   RUSSIA
12      MAIL    RAIL    1995-01-01
9       cornflower
14      1995-12-01
5       EUROPE 1997-01-01
16      Brand#22        LARGE PLATED  13      1
3       25      18      17      28      16


========================
stream10 params
========================
6       1997-01-01      0.05    25
15      1996-07-01
18      314
17      Brand#23        LG BOX
12      MAIL    RAIL    1995-01-01
1       106
7       JAPAN   RUSSIA
2       33      BRASS   EUROPE
22      15      19      24      12      16      27      30
13      pending         accounts
21      JORDAN
10      1994-06-01
14      1995-12-01
9       cornflower
3       AUTOMOBILE      1995-03-23
16      Brand#22        LARGE PLATED  13      1
3       25      18      17      28      16
20      lace    1995-01-01      ETHIOPIA
19      Brand#23        Brand#32        Brand#41
1       13      28
11      ROMANIA         0.0000000033
4       1996-08-01
8       RUSSIA EUROPE SMALL ANODIZED STEEL
5       EUROPE 1997-01-01
```

Appendix E. Implementation-Specific Layer/Driver Code

```
================================================
```

do30tb.sh

```
================================================

#/usr/bin/bash

# Create results directory

timestamp=`date "+%m_%d_%H:%M"`
dir=$HOME/results/$timestamp
mkdir -p $dir
echo $dir > /tmp/this_tpch

# Load

cd $HOME/run/scripts >> $dir/tpch.log
sh clean.sh
tpch.sh load 30000 10 < go >> $dir/tpch.log
```

```
cqi xstop
sleep 15
cqi xstart
#
# # Run
#
tpch.sh all 30000 10 48 132 50000 < go >>
$dir/tpch.log
```

Appendix F. Misc database scripts

Auditor Scripts

==========

dbtables.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', LOCALTIMESTAMP;
--
================================================================
============
--           TABLE: LINEITEM
--
================================================================
============
SELECT COUNT(*) FROM LINEITEM;
SELECT * FROM LINEITEM
WHERE L_ORDERKEY IN
      ( 4, 26598,  148577,  387431, 56704, 517442,
600000)
      AND L_LINENUMBER = 1
ORDER BY L_ORDERKEY;
--
================================================================
============
--           TABLE: ORDERS
--
================================================================
============
-- GET TIMESTAMP
SELECT 'TIME', LOCALTIMESTAMP;
SELECT COUNT(*) FROM ORDERS;
SELECT * FROM ORDERS
WHERE O_ORDERKEY IN ( 7, 44065, 287590, 411111,
483876, 599942 )
ORDER BY O_ORDERKEY;
--
================================================================
============
--           TABLE: PART
--
================================================================
============
-- GET TIMESTAMP
SELECT 'TIME', LOCALTIMESTAMP;
SELECT COUNT(*) FROM PART;
SELECT * FROM PART
WHERE P_PARTKEY IN (1,984,8743,9028,13876,17899,20000)

ORDER BY P_PARTKEY;
--
================================================================
============
--           TABLE: PARTSUPP
--
================================================================
============
-- GET TIMESTAMP
SELECT 'TIME', LOCALTIMESTAMP;
SELECT COUNT(*) FROM PARTSUPP;
SELECT* FROM PARTSUPP
   WHERE PS_PARTKEY = 3398
   AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
      FROM PARTSUPP WHERE PS_PARTKEY = 3398);
SELECT* FROM PARTSUPP
   WHERE PS_PARTKEY =15873
   AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
      FROM PARTSUPP WHERE PS_PARTKEY = 15873);
SELECT* FROM PARTSUPP
   WHERE PS_PARTKEY = 11394
   AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
      FROM PARTSUPP WHERE PS_PARTKEY = 11394);
SELECT* FROM PARTSUPP
   WHERE PS_PARTKEY = 6743
   AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
      FROM PARTSUPP WHERE PS_PARTKEY = 6743);
SELECT* FROM PARTSUPP
   WHERE PS_PARTKEY = 19763
   AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
      FROM PARTSUPP WHERE PS_PARTKEY =19763);
--
================================================================
============
--           TABLE: SUPPLIER
--
================================================================
============
-- GET TIMESTAMP
SELECT 'TIME', LOCALTIMESTAMP;
SELECT COUNT(*) FROM SUPPLIER;
SELECT * FROM SUPPLIER
WHERE  S_SUPPKEY IN (83,265,492,784,901,1000)
ORDER BY S_SUPPKEY;
--
================================================================
============
--           TABLE: CUSTOMER
--
================================================================
============
-- GET TIMESTAMP
SELECT 'TIME', LOCALTIMESTAMP;
SELECT COUNT(*) FROM CUSTOMER;
SELECT * FROM CUSTOMER
WHERE  C_CUSTKEY IN (832,2653,4924,7845,92016,108070)
ORDER BY C_CUSTKEY;
--
================================================================
============
--           TABLE: NATION & REGION
--
================================================================
============
-- GET TIMESTAMP
SELECT 'TIME', LOCALTIMESTAMP;
SELECT * FROM REGION;
SELECT COUNT(*) FROM NATION;
SELECT * FROM NATION
WHERE N_NATIONKEY IN (3,10,14,20)
ORDER BY N_NATIONKEY;
--
================================================================
============
--           CHECK KEY VALUES
```

```
--
=====================================================
============
-- GET TIMESTAMP
SELECT 'TIME', LOCALTIMESTAMP;
  DROP table MINMAX;
CREATE TABLE MINMAX
(TNAME CHAR(15),
KEYMIN INTEGER,
KEYMAX INTEGER);
INSERT INTO MINMAX
SELECT 'LINEITEM_ORD',MIN(L_ORDERKEY),MAX(L_ORDERKEY)
FROM LINEITEM;
INSERT INTO MINMAX
SELECT
'LINEITEM_NBR',MIN(L_LINENUMBER),MAX(L_LINENUMBER)
FROM LINEITEM;
INSERT INTO MINMAX
SELECT 'ORDERS',MIN(O_ORDERKEY),MAX(O_ORDERKEY)
FROM ORDERS;
INSERT INTO MINMAX
SELECT 'CUSTOMER',MIN(C_CUSTKEY),MAX(C_CUSTKEY)
FROM CUSTOMER;
INSERT INTO MINMAX
SELECT 'PART',MIN(P_PARTKEY),MAX(P_PARTKEY)
FROM PART;
INSERT INTO MINMAX
SELECT 'SUPPLIER',MIN(S_SUPPKEY),MAX(S_SUPPKEY)
FROM SUPPLIER;
INSERT INTO MINMAX
SELECT 'PARTSUPP_PART',MIN(PS_PARTKEY),MAX(PS_PARTKEY)
FROM PARTSUPP;
INSERT INTO MINMAX
SELECT 'PARTSUPP_SUPP',MIN(PS_SUPPKEY),MAX(PS_SUPPKEY)
FROM PARTSUPP;
INSERT INTO MINMAX
SELECT 'NATION',MIN(N_NATIONKEY),MAX(N_NATIONKEY)
FROM NATION;
INSERT INTO MINMAX
SELECT 'REGION',MIN(R_REGIONKEY),MAX(R_REGIONKEY)
FROM REGION;
SELECT * FROM MINMAX;
  DROP table MINMAX;
SELECT 'END TIME', LOCALTIMESTAMP;
```

```
====================================
```

dbinsert_30TB.sql

```
====================================

--
=====================================================
============+
--  FILENAME
--       DBINSERT.SQL
--  DESCRIPTION
--       INSERTS DUPLICATE ROWS WITH NEW KEY VALUES
AND
--       INSERTS ROWS WITH VALUES BEYOND THE TPC-H
RANGES.
--
--
=====================================================
============
--
--
--
=====================================================
============
-- DUPLICATES
--
=====================================================
============
-- GET TIMESTAMP
SELECT LOCALTIMESTAMP;

-- DROP temporary tables. Ignore error messages
DROP table TEMP_PART;
DROP table TEMP_SUPPLIER;
DROP table TEMP_PARTSUPP;
DROP table TEMP_CUSTOMER;
DROP table TEMP_ORDERS;
DROP table TEMP_LINEITEM;
DROP table TEMP_NATION;
DROP table TEMP_REGION;

-- CREATE temporary tables
create table temp_part (
  p_partkey bigint not null encode
delta,
  p_name varchar(55) not null,
  p_mfgr char(25) not null encode
bytedict,
  p_brand char(10) not null encode
bytedict,
  p_type varchar(25) not null,
  p_size int4 not null encode bytedict,
  p_container char(10) not null encode
bytedict,
  p_retailprice numeric(12,2) not null encode
delta32k,
  p_comment varchar(23) not null encode text255
);

create table temp_region (
  r_regionkey int4 not null encode always8,
  r_name char(25) not null encode bytedict,
  r_comment varchar(152) not null
);

create table temp_nation (
  n_nationkey int4 not null encode always8,
  n_name char(25) not null encode bytedict,
  n_regionkey int4 not null encode always8,
  n_comment varchar(152) not
null
);

create table temp_supplier (
  s_suppkey int4 not
null,
  s_name char(25) not null,

  s_address varchar(40) not null,

  s_nationkey int4 not null encode always8,
  s_phone char(15) not null,

  s_acctbal numeric(12,2) not null,

  s_comment varchar(101) not null encode
text255
);

create table temp_partsupp (
  ps_partkey bigint not null encode
delta,
  ps_suppkey int4 not null encode
delta32k,
  ps_availqty int4 not null encode
delta32k,
  ps_supplycost numeric(12,2) not null encode
zeros,
  ps_comment varchar(199) not null encode text255
);

create table temp_customer (
  c_custkey bigint not null encode
delta,
  c_name varchar(25) not null,
  c_address varchar(40) not
```

```
null,
  c_nationkey int4 not null encode
bytedict,
  c_phone char(15) not
null,
  c_acctbal numeric(12,2) not null encode
zeros,
  c_mktsegment char(10) not null encode
bytedict,
  c_comment varchar(117) not null encode
text255
);

create table temp_orders (
  o_orderkey bigint not null encode
delta,
  o_custkey bigint not null,
  o_orderstatus char(1) not
null,
  o_totalprice numeric(12,2) not null encode always32,
  o_orderdate date not null encode
delta32k,
  o_orderpriority char(15) not null encode
bytedict,
  o_clerk char(15) not null,
  o_shippriority int4 not null encode
runlength,
  o_comment varchar(79) not null encode
text255
);

create table temp_lineitem (
  l_orderkey bigint not null encode
delta,
  l_partkey bigint not null,

  l_suppkey int4 not null,
  l_linenumber int4 not null encode
always8,
  l_quantity numeric(12,2) not null encode
bytedict,
  l_extendedprice numeric(12,2) not null encode
always32,
  l_discount numeric(12,2) not null encode
bytedict,
  l_tax numeric(12,2) not null encode
bytedict,
  l_returnflag char(1) not
null,
  l_linestatus char(1) not
null,
  l_shipdate date not null encode
delta,
  l_commitdate date not null encode
delta,
  l_receiptdate date not null encode
delta,
  l_shipinstruct char(25) not null encode
bytedict,
  l_shipmode char(10) not null encode
bytedict,
  l_comment varchar(44) not null encode text255
);


BEGIN TRANSACTION;
INSERT INTO TEMP_PART
SELECT * FROM PART
       WHERE P_PARTKEY = 1;

UPDATE TEMP_PART
       SET P_PARTKEY = 2147483647;

INSERT INTO PART
       SELECT * FROM TEMP_PART;

SELECT * FROM PART
       WHERE P_PARTKEY = 2147483647
       OR P_PARTKEY = 1;

DELETE FROM PART
       WHERE P_PARTKEY = 2147483647;

COMMIT;

BEGIN TRANSACTION;
INSERT INTO TEMP_SUPPLIER
SELECT * FROM SUPPLIER
       WHERE S_SUPPKEY = 1;

UPDATE TEMP_SUPPLIER
       SET S_SUPPKEY = 2147483647;

INSERT INTO SUPPLIER
       SELECT * FROM TEMP_SUPPLIER;

SELECT * FROM SUPPLIER
       WHERE S_SUPPKEY = 2147483647
       OR S_SUPPKEY = 1;

DELETE FROM SUPPLIER
       WHERE S_SUPPKEY = 2147483647;

COMMIT;

BEGIN TRANSACTION;
INSERT INTO TEMP_PARTSUPP
SELECT * FROM PARTSUPP
       WHERE PS_PARTKEY = 1
       AND PS_SUPPKEY = 2;

UPDATE TEMP_PARTSUPP
       SET PS_PARTKEY = 2147483647,
           PS_SUPPKEY = 2147483647;

INSERT INTO PARTSUPP
       SELECT * FROM TEMP_PARTSUPP;

SELECT * FROM PARTSUPP
       WHERE (PS_PARTKEY = 2147483647
       AND    PS_SUPPKEY = 2147483647)
       OR    (PS_PARTKEY = 1
       AND    PS_SUPPKEY = 2);

DELETE FROM PARTSUPP
       WHERE PS_PARTKEY = 2147483647
       AND   PS_SUPPKEY = 2147483647;

COMMIT;

BEGIN TRANSACTION;
INSERT INTO TEMP_CUSTOMER
SELECT * FROM CUSTOMER
       WHERE C_CUSTKEY = 1;

UPDATE TEMP_CUSTOMER
       SET C_CUSTKEY = 2147483647;

INSERT INTO CUSTOMER
       SELECT * FROM TEMP_CUSTOMER;

SELECT * FROM CUSTOMER
       WHERE C_CUSTKEY = 2147483647
       OR C_CUSTKEY = 1;

DELETE FROM CUSTOMER
       WHERE C_CUSTKEY = 2147483647;

COMMIT;

BEGIN TRANSACTION;
INSERT INTO TEMP_ORDERS
SELECT * FROM ORDERS
       WHERE O_ORDERKEY = (SELECT MIN(O_ORDERKEY) FROM
ORDERS);
```

```
UPDATE TEMP_ORDERS
       SET O_ORDERKEY = 2147483647;

INSERT INTO ORDERS
       SELECT * FROM TEMP_ORDERS;

SELECT * FROM ORDERS
       WHERE O_ORDERKEY = 2147483647
       OR O_ORDERKEY = (SELECT MIN(O_ORDERKEY) FROM
ORDERS);

DELETE FROM ORDERS
       WHERE O_ORDERKEY = 2147483647;

COMMIT;

BEGIN TRANSACTION;
INSERT INTO TEMP_LINEITEM
SELECT * FROM LINEITEM
       WHERE L_ORDERKEY = (SELECT MIN(O_ORDERKEY) FROM
ORDERS)
       AND   L_LINENUMBER = 1;

UPDATE TEMP_LINEITEM
       SET L_ORDERKEY = 2147483647,
           L_PARTKEY  = 2147483647,
           L_SUPPKEY  = 2147483647,
           L_LINENUMBER = 114;

INSERT INTO LINEITEM
       SELECT * FROM TEMP_LINEITEM;

SELECT * FROM LINEITEM
       WHERE (L_ORDERKEY = 2147483647
       AND    L_PARTKEY  = 2147483647
       AND    L_SUPPKEY  = 2147483647
       AND    L_LINENUMBER = 114)
       OR     (L_ORDERKEY = (SELECT MIN(O_ORDERKEY)
FROM ORDERS)
       AND    L_LINENUMBER = 1);

DELETE FROM LINEITEM
       WHERE L_ORDERKEY = 2147483647
       AND   L_PARTKEY  = 2147483647
       AND   L_SUPPKEY  = 2147483647
       AND   L_LINENUMBER = 114;

COMMIT;

BEGIN TRANSACTION;
INSERT INTO TEMP_NATION
SELECT * FROM NATION
       WHERE N_NATIONKEY = 1;

UPDATE TEMP_NATION
       SET N_NATIONKEY = 114;

INSERT INTO NATION
       SELECT * FROM TEMP_NATION;

SELECT * FROM NATION
       WHERE N_NATIONKEY = 114
       OR N_NATIONKEY = 1;

DELETE FROM NATION
       WHERE N_NATIONKEY = 114;

COMMIT;

BEGIN TRANSACTION;
INSERT INTO TEMP_REGION
SELECT * FROM REGION
       WHERE R_REGIONKEY = 1;

UPDATE TEMP_REGION
       SET R_REGIONKEY = 114;

INSERT INTO REGION
       SELECT * FROM TEMP_REGION;

SELECT * FROM REGION
       WHERE R_REGIONKEY = 114
       OR R_REGIONKEY = 1;

DELETE FROM REGION
       WHERE R_REGIONKEY = 114;

COMMIT;

--
============================================================
============
-- DUPLICATES FINISHED STARTING INSERTS FOR DOMAIN
RANGE
--
============================================================
============

-- GET TIMESTAMP
SELECT LOCALTIMESTAMP;

INSERT INTO SUPPLIER
       (S_SUPPKEY, S_NAME, S_ADDRESS, S_NATIONKEY,
S_PHONE,
       S_ACCTBAL, S_COMMENT)
       VALUES
       (2147483647, 'NAME TEXT ............25E',
       'ADDRESS VARCHAR .............30.......40E',
       2147483647,'THIS IS PHONE E', 1234567890.12,
       'SUPPLIER COMMENT FIELD IS 101 LONG NO E');

SELECT * FROM SUPPLIER
       WHERE S_SUPPKEY = 2147483647;

DELETE FROM SUPPLIER
       WHERE S_SUPPKEY = 2147483647;

--
============================================================
============
INSERT INTO PART
       (P_PARTKEY, P_NAME, P_MFGR, P_BRAND, P_TYPE,
       P_SIZE, P_CONTAINER, P_RETAILPRICE, P_COMMENT)
       VALUES
       (2147483647, 'PNAME
TEXT .......2.........3.........4....5E',
       'PMFGR TEXT........2....5E','PBRAND 10E',
       'PTYPE VARCHAR.....2....5E', -2147483646,
       'PCONTAINRE', 1234567890.12,
       'PART COMMENT FIELD  23E');

SELECT * FROM PART
       WHERE P_PARTKEY = 2147483647;

DELETE FROM PART
       WHERE P_PARTKEY = 2147483647;

--
============================================================
============
INSERT INTO PARTSUPP
       (PS_PARTKEY, PS_SUPPKEY, PS_AVAILQTY,
PS_SUPPLYCOST,
       PS_COMMENT)
       VALUES
       (2147483647, 2147483647, -2147483646,
1234567890.12,
       'PS COMMENT FIELD IS 199 LONG NO E');

SELECT * FROM PARTSUPP
       WHERE PS_PARTKEY = 2147483647
       AND   PS_SUPPKEY = 2147483647;

DELETE FROM PARTSUPP
       WHERE PS_PARTKEY = 2147483647
       AND   PS_SUPPKEY = 2147483647;
```

```
--
============================================================
============
INSERT INTO CUSTOMER
       (C_CUSTKEY, C_NAME, C_ADDRESS, C_NATIONKEY,
       C_PHONE, C_ACCTBAL, C_MKTSEGMENT, C_COMMENT)
       VALUES
       (2147483647, 'CUSTOMER NAME GOES TO 25E',
       'CUSTOMER ADDRESS GOES HERE..3.........4E',
       2147483647, 'THIS IS PHONE E', 1234567890.12,
       'MARKT SEGE', 'CUSTOMER COMMENTS FIELS IS 117
LONG NO E');

SELECT * FROM CUSTOMER
       WHERE C_CUSTKEY = 2147483647;

DELETE FROM CUSTOMER
       WHERE C_CUSTKEY = 2147483647;

--
============================================================
============
INSERT INTO ORDERS
       (O_ORDERKEY, O_CUSTKEY, O_ORDERSTATUS,
O_TOTALPRICE,
       O_ORDERDATE, O_ORDERPRIORITY, O_CLERK,
O_SHIPPRIORITY,
       O_COMMENT)
       VALUES
       (2147483647, 2147483647, 'X', 1234567890.12,
localtimestamp,
       'ORDER PRIORTY5E', 'FIXED TEXT  15E', 6,
       'ORDER COMMENTS FIELD IS 79 NO E');

SELECT * FROM ORDERS
       WHERE O_ORDERKEY = 2147483647
       AND  O_CUSTKEY = 2147483647;

DELETE FROM ORDERS
       WHERE O_ORDERKEY = 2147483647
       AND  O_CUSTKEY = 2147483647;

--
============================================================
============
INSERT INTO LINEITEM
       (L_ORDERKEY, L_PARTKEY, L_SUPPKEY,
L_LINENUMBER,
       L_QUANTITY, L_EXTENDEDPRICE, L_DISCOUNT, L_TAX,
       L_RETURNFLAG, L_LINESTATUS, L_SHIPDATE,
L_COMMITDATE,
       L_RECEIPTDATE, L_SHIPINSTRUCT, L_SHIPMODE,
L_COMMENT)
       VALUES
       (2147483647,
       2147483647,
       2147483647,
       114,
       -1234567890.12,
       -1234567890.12,
       -1234567890.12,
       -1234567890.12,
       'Q',
       'R',
       localtimestamp,
       localtimestamp,
       localtimestamp,
       'SHIP BY CAMEL .........5E',
       'SHIP ASAPE',
       'IS THIS REALLY WHAT YOU WANTED? 44
LONG....E');

SELECT * FROM LINEITEM
       WHERE L_ORDERKEY = 2147483647
       AND L_PARTKEY = 2147483647
       AND L_SUPPKEY = 2147483647
       AND L_LINENUMBER = 114;

DELETE FROM LINEITEM
       WHERE L_ORDERKEY = 2147483647
       AND L_PARTKEY = 2147483647
       AND L_SUPPKEY = 114
       AND L_LINENUMBER = -2147483646;

--
============================================================
============
INSERT INTO NATION
       (N_NATIONKEY, N_NAME, N_REGIONKEY, N_COMMENT)
       VALUES
       (114,
       'ZE REPUBLIC D MAKEBELIEVE',
       114,
       'A NATION COMMENT FOR FIELD SIZE 152 NO E');

SELECT * FROM NATION
       WHERE N_NATIONKEY = 114
       AND N_REGIONKEY = 114;

DELETE FROM NATION
       WHERE N_NATIONKEY = 114
       AND N_REGIONKEY = 114;

--
============================================================
============
INSERT INTO REGION
       (R_REGIONKEY, R_NAME, R_COMMENT)
       VALUES
       (2147483647,
       'ZE ENDS OF THE EARTH....E',
       'A REASONABLE COMMENT WOULD GO HERE');

SELECT * FROM REGION
       WHERE R_REGIONKEY = 114;

DELETE FROM REGION
       WHERE R_REGIONKEY = 114;

-- GET TIMESTAMP
SELECT LOCALTIMESTAMP;

--
============================================================
============
-- DROP TEMP TABLES
--
============================================================
============
DROP table TEMP_PART;
DROP table TEMP_SUPPLIER;
DROP table TEMP_PARTSUPP;
DROP table TEMP_CUSTOMER;
DROP table TEMP_ORDERS;
DROP table TEMP_LINEITEM;
DROP table TEMP_NATION;
DROP table TEMP_REGION;

-- GET TIMESTAMP
SELECT LOCALTIMESTAMP;
--
============================================================
============
-- DONE
--
============================================================
============
```

Appendix G. Pricing information

Price quotes from ParAccel Inc, CDW and Agilsys are included below.

Sales Quote: ParAccel Inc.

Company Sun Microsystems
Contact Guido Ficco
Phone 781-442-0069
Fax
Address 1 Network Drive
Burlington MA 01803

ParAccel Sales Rep: MikeAzevedo
Phone: 858-309-4733
Fax: 866-903-0335

	Part Number	Description	List Price	Quantity	Discount Discount	Extended Price	Extended Support Fees 3 YEARS
1	PAR-ADB	ParAccel Analytic Database Per TB	100,000	30	57.00%	$1,290,000	
2	PAR-SUPP-B	1 yr 24/7 ParAccel Support Per TB	1,000	90	57.00%		$38,700.00
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							

Quote Date: ***6/16/09***
Valid thru:

Total $1,328,700
License + 3 year support

Payment terms : Net 30 Days

CDW Quote

Account #: 10000017

Contact: MATT DIENER
Quote Number: PBV8656
Order Date: 6/9/2009
Expiration Date: 9/7/2009
PO Number: 090609 QUOTE

Ship To:
PARACCEL ATTN: ACCOUNTS PAYABLE9920 PACIFIC HEIGHTS BLVD STE 450 SAN DIEGO CA,92121-4396
Bill To:
PARACCEL ATTN: ACCOUNTS PAYABL ATTN: ACCOUNTS PAYABLE9920 PACIFIC HEIGHTS BLVD SAN DIEGO CA,92121-4396

Payment Type: Net Terms
Shipping Type: UPS Ground (1-2 day)

Item(s):

Quantity: 4.000
Product: **Cisco Catalyst 3750G-24T Switch**
CDW Part#: 497268
Price: $3,615.00
Ext. Price: $14,460.00

Quantity: 3.000
Product: **Cisco Catalyst 3750G-48TS SMI**
CDW Part#: 722414
Price: $8,450.00
Ext. Price: $25,350.00

Quantity: 172.000
Product: **CDW 7' CAT5e or CAT5 RJ45 Patch Cable Blue**

CDW Part#: 630135
Price: $1.25
Ext. Price: $215.00

Quantity: 12.000
Product: **Cisco SMARTnet Onsite Premium extended service agreement - 1 year - on-site**
CDW Part#: 963920
Price: $825.00
Ext. Price: $9,900.00

Quantity: 9.000
Product: **Cisco SMARTnet Onsite Premium extended service agreement - 1 year - on-site**
CDW Part#: 963921

Price: $1,750.00
Ext. Price: $15,750.00

Quantity: 43.000
Product: **Minuteman Pro 1500E**
CDW Part#: 682296
Price: $266.22
Ext. Price: $11,447.46

==========

Sub Total: $77,122.46

Freight: $1,714.10
*Sales Tax: $4,503.85
Total: $83,340.41

* Tax may change if this quote is amended.

Follow this link to view/purchase the item(s) in this quote:
http://www.cdw.com/r.asp?n=17368&cdwqn=PBV8656

CDW Corporation
The Right Technology. Right Away. (tm)

Agilysys
...formerly Innovativ

204 Fernwood Avenue, Edison, NJ 08837
Phone: (732) 417-1162 Fax: (732) 225-8351
www.agilysys.com

To: AT&T Corp

QUOTE QT056436

Please reference this quote # on your purchase order.

Date: 6/11/09
Issued By: Dan Pearson
Phone #: (720) 214-1538
Email: dan.pearson@agilysys.com
Project:

Thank you for your inquiry. We are pleased to quote as follows: **Contract No: 20040818.3.C** Please fax all purchase orders to: (732) 225-8351

Master ID	Salesperson	Customer ID	Payment Terms	Purchase Order
65096	I. Temin/ S. Moran * Ph: (732) 692-1981 * attreps@innovativ.com	AT&T-07	NET 45	

Line	Item / Description	Qty	List Price	Discount	Unit Price	Ext. Price
1	B24-FSZ2-64G-HMH Sun Fire X4540 server, 2x AMD Opteron Model quad-core 2356, 2.3GHz Processor,16x4GB Mem, 48 SATA 3.5 500GB, 7200RPM HDD, 2x PSU, Service Processor, 4x 10/100/1000 Ethernet ports, 4x USB 2.0 ports, 3x PCI-E slots, VGA port, 1xflash boot disk slot, no p	43	$41,995.00	18.00%	$34,435.90	$1,480,743.70
2	X311L Localized Power Cord Kit North American/Asian This Product is Hazard Class Y, RoHS compliant.	86				
	SUN MAINTENANCE 3 YEAR TERM Gold Support - Phone and Online Technical Support: 24/7, Live transfer. HW Service Coverage: 8-8, M-F; On-site within 4 hrs. SunVIP Interop Support. Knowledgebase Access. Solaris Releases. On-demand Solaris Updates. Online System Admin Resources. Sun Alerts and Notification Services. SunSpectrum eLearning Library. Discounts: Warranty Upgrade 30%, Ar-50958:Eu 64%					
3	GOLD SUPPORT GOLD SUPPORT B24-FSZ2-64G-HMH 2 QuadCore 2356,64 GBMem, 24TB	43	$2,766.60		$2,766.60	$118,963.80

Please Note: Quote expires 30 days from date of Quote. This quote is provided subject to the above referenced Master Agreement on file Some conditions or additions may be necessary before this quote is final, which may include but is not limited to Insurance, Frieght, and applicable Sales Tax.

Systems will not be integrated unless requested and quoted.

Returns must be in original unopened packaging and factory seals must be intact. All returns must be made within 15 days and have an RMA# assigned by Agilysys. The RMA number must be clearly marked on the shipping label ONLY. Writing on boxes will invalidate the return. **No discontinued, promotional, special order, remanufactured or specially configured/integrated products may be returned to Agilysys NJ, Inc. for customer credit**.

Total $1,599,707.50

(excluding tax and Freight)

Please read the following: The delivery of services from Sun Microsystems (Sun) is governed by (1) the contract terms set out at http://www.sun.com/sales/salesterms/. **unless you have a service agreement with Sun that would govern the delivery of services; and (2) the appropriate service listing(s) set out at** http://www.sun.com/servicelist/.