



TPC Benchmark™ H Full Disclosure Report

Sun Microsystems Sun Fire™ X4100 Server Using ParAccel Analytic Database™

Submitted for Review
Report Date: Oct 29, 2007

TPC Benchmark H Full Disclosure Report

First Printing

© 2006 Sun Microsystems, Inc.

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

All rights reserved. This product and related documentation are protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or related documentation may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the United States Government is subject to the restrictions set forth in DFARS 252.227-7013 (c)(1)(ii) and FAR 52.227-19, Rights in Technical Data and Computer Software (October 1988).

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

TRADEMARKS

Sun, Sun Microsystems, the Sun logo, Sun Fire V440 Server, SMCC, the SMCC logo, SunSoft, the SunSoft logo, Solaris, SunOS, OpenWindows, DeskSet, ONC, and NFS are trademarks or registered trademarks of Sun Microsystems, Inc. All other product names mentioned herein are the trademarks of their respective owners.

All SPARC trademarks, including the SCD Compliant Logo, are trademarks or registered trademarks of SPARC International, Inc. SPARCstation, SPARCserver, SPARCengine, SPARCworks, and SPARCcompiler are licensed exclusively to Sun Microsystems, Inc. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK™ and Sun™ Graphical User Interfaces were developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

TPC-H Benchmark™ is a trademark of the Transaction Processing Performance Council.

ParAccel Analytic Database™ is a registered trademark of ParAccel, Inc.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.

Sun Microsystems, Inc., believes that the information in this document is accurate as of its publication date. The information in this document is subject to change without notice. Sun Microsystems, Inc., assumes no responsibility for any errors that may appear in this document.

The pricing information in this document is believed to accurately reflect prices in effect on Report Date: Oct 29, 2007. However, Sun Microsystems and ParAccel, Inc. provide no warranty on the pricing information in this document.

The performance information in this document is for guidance only. System performance is highly dependent on many factors including system hardware, system and user software, and user application characteristics. Customer applications must be carefully evaluated before estimating performance. Sun Microsystems, Inc., does not warrant or represent that a user can or will achieve a similar performance. No warranty on system performance or price/performance is expressed or implied in this document.



PARACCEL

Sun Fire™ X4100 Server with ParAccel Analytic Database™

TPC-H Rev. 2.6.0

Report Date: Oct 29, 2007
Revision Date: Nov. 8, 2007

Total System Cost

Composite Query per Hour Metric

Price/Performance

\$624,770.12

198,578.1
QphH@300GB

\$3.15
per QphH@300GB

Database Manager

Operating System

Other Software

Availability Date

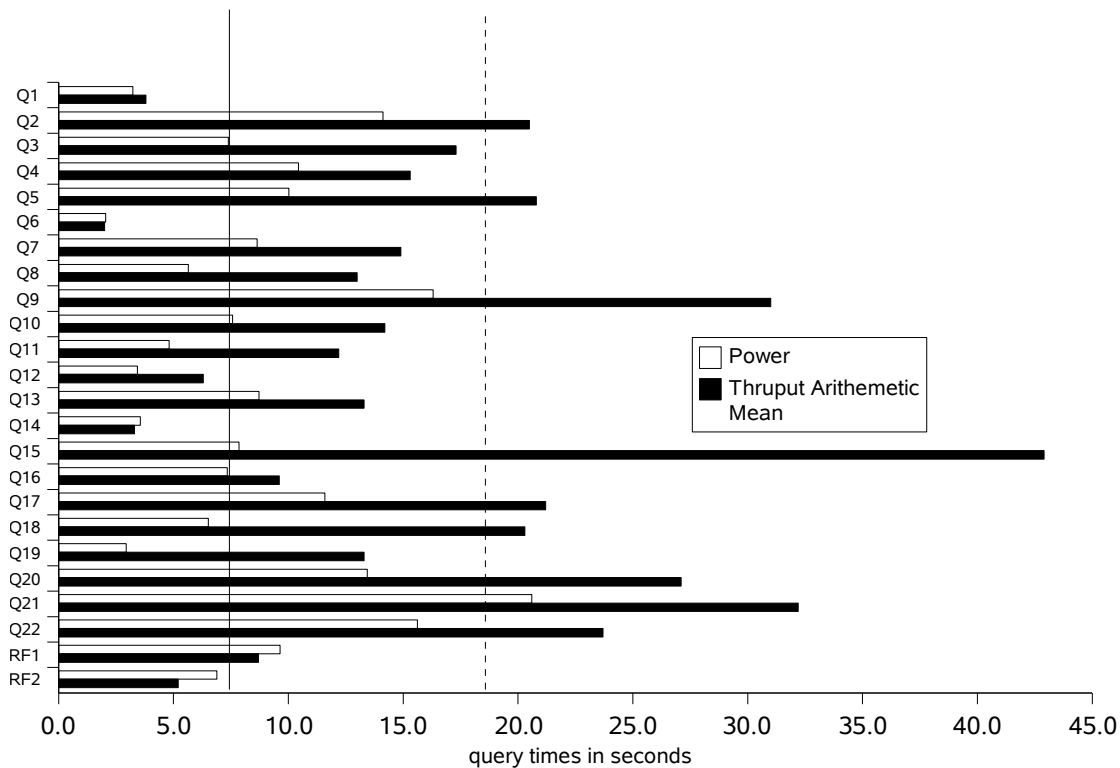
300GB

**ParAccel Analytic
Database™**

**Linux Red Hat
RHEL 4.4
Enterprise Server**

October 29, 2007

geometric mean = 7.4 arithmetic mean = 18.6



RAID (Base tables): Y

RAID (Base tables and auxiliary data structures): Y

RAID (All): Y

Database Load Time = 0Hrs15Min05sec

Load Includes Backup: N

Total Storage/Database Size=27.19

System Configuration:

30 x SunFire X4100 Server, each server with

2 AMD Opteron 2.8GHz processors (each processor is 1 chip, 2 cores, 2 threads)

16 GB memory

2 x 146 GB (10K RPM) internal SAS disks

Total Storage: 8,158 GB (in this calculation 1 GB is defined as 1024 * 1024 * 1024) bytes



Sun Fire™ X4100 Server with ParAccel Analytic Database™

TPC-H Rev. 2.6.0

Report Date: Oct 29, 2007

Revision Date: Nov. 8, 2007

Description	Part Number	Source	Unit Price	Qty	Ext. Price	3 Yr. maint
Server Hardware						
SunFire X4100 M2 x64 Server						
2 X AMD Opteron Model 2220 (2.8GHz) processor						
4 X 2GB DDR2-667 Memory						
2 X 146 GB HDD						
	A86-FPZ2BH8GKBA	2	6,590.00	30	197,700.00	
4 GB Memory Kit DDR2-667 DIMMS (2 X 2GB) for SunFire X4100 M2	4226A-Z	2	750.00	60	45,000.00	
Server Discount (14%)		2			-33,978.00	
SunFire X4100 M2 Server upgrade to 3 years.						
24x7 hardware only support with 4 hour response	W9D-A87-24-3H	2	792.00	30		23,760.00
Support Discount (10%)		2				-2,376.00
Cisco Catalyst 3750-24TS 24-port switch	WS-C3750-24TS-E	2	5,990.00	4	23,960.00	
3750 Discount		2	-2,239.47	4	-8957.88	
Cisco Catalyst 3750-24TS 24-port switch 1 year premium service	CON-OSP-375024TE	2	1,006.00	12		12,072.00
3750 Support Discount		2	-111.78	12		-1341.36
Ethernet 100Base-TX cable	A3L791-05-GRN	2	5.99	60	359.40	
Cable discount		2	-1.13	60	-67.80	
APC Smart-UPS 3000VA USB	592679	3	1,182.99	6	7,097.94	
1 Year PC/Peripheral Extended Service \$701-\$1200	54439	3	53.99	18		971.82
Server Hardware Subtotal					231,113.66	33,086.46
External Storage						
None						
Server Software						
ParAccel Analytic Database, per GB data	PAR-ADB	1	1,000.00	300	300,000.00	
ParAccel Support \$10 per Gb data per year	PAR-SUP-B	1	10.00	300		9,000.00
ParAccel Discount 5%		1			-15,000.00	-450.00
RHEL 4.4 24x7 Support per node 3 years (up to 2 sockets)		2	3,702.00	30		111,060.00
RHEL Discount		2	-1,468.00	30		-44,040.00
Server Software Subtotal					285,000.00	75,570.00
Total					516,113.66	108,656.46
3 Yr. Cost					624,770.12	
QphH @ 300GB					198,578.1	
\$/QphH @ 300GB					\$3.15	

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

Sources

1. ParAccel, Inc.
2. Continental Resources, Inc.
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™ X4100 Server
with ParAccel Analytic
Database™**

TPC-H Rev. 2.6.0

Report Date: Oct 29, 2007

Revision Date: Nov. 8, 2007

Numerical Quantities

Measurement Results:

Database Scale Factor	= 300GB
Total Data Storage / Database Size	= 27.19
Start of database load time	= 21:29:16
End of database load time	= 21:44:21
Database Load Time	= 0Hrs15Min05sec
Query Streams for Throughput Test	= 6
TPC-H Power	= 145,219.3
TPC-H Throughput	= 271,542.9
TPC-H Composite Query-per-Hour Rating (QphH@300GB)	= 198,578.1
Total System Price Over 3 Years	= \$624,770.12
TPC-H Price/Performance Metric (\$/QphH@300GB)	= \$3.15

Measurement Intervals:

Measurement Interval in Throughput Test (Ts)	= 525 seconds
--	---------------

Duration of Stream Execution:

Stream ID	Seed	Start Date	Start Time	End Date	End Time	Duration
Stream 0	1021214421	2007-10-21	22:08:47	2007-10-21	22:12:15	3mins:28secs
Stream 1	1021214422	2007-10-21	22:12:15	2007-10-21	22:19:40	7mins:25secs
Stream 2	1021214423	2007-10-21	22:12:15	2007-10-21	22:18:49	6mins:34secs
Stream 3	1021214424	2007-10-21	22:12:15	2007-10-21	22:19:45	7mins:10secs
Stream 4	1021214425	2007-10-21	22:12:15	2007-10-21	22:18:32	6mins:17secs
Stream 5	1021214426	2007-10-21	22:12:15	2007-10-21	22:18:28	6mins:13secs
Stream 6	1021214427	2007-10-21	22:12:15	2007-10-21	22:19:36	7mins:21secs
Refresh		2007-10-21	22:19:40	2007-10-21	22:21:00	1mins:20secs



**Sun Fire™ X4100 Server
with ParAccel Analytic
Database™**

TPC-H Rev. 2.6.0

Report Date: Oct 29, 2007

Revision Date: Nov. 8, 2007

TPC-H Timing Intervals (in seconds)

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Stream 00	3.2	14.1	7.4	10.4	10.0	2.0	8.6	5.6	16.3	7.6	4.8	3.4
Stream 01	2.8	17.1	19.5	8.8	22.4	3.8	19.8	9.9	22.5	14.6	12.7	7.5
Stream 02	3.9	31.3	19.8	12.6	19.8	1.5	14.3	14.4	26.2	10.6	12.5	5.3
Stream 03	5.5	23.4	8.7	18.3	39.8	1.1	15.3	17.0	27.0	15.6	12.9	7.0
Stream 04	3.8	25.1	23.5	15.0	19.6	1.9	15.0	14.9	58.1	7.3	15.2	7.0
Stream 05	3.7	20.6	23.1	26.8	16.4	1.0	19.1	12.5	30.0	13.4	12.8	5.0
Stream 06	4.0	11.5	19.4	15.0	17.5	2.6	12.4	16.8	36.9	30.2	14.3	9.0
Minimum	2.8	11.5	7.4	8.8	10.0	1.0	8.6	5.6	16.3	7.3	4.8	3.4
Average	3.8	20.5	17.3	15.3	20.8	2.0	14.9	13.0	31.0	14.2	12.2	6.3
Maximum	5.5	31.3	23.5	26.8	39.8	3.8	19.8	17.0	58.1	30.2	15.2	9.0

	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream 00	8.7	3.6	7.9	7.3	11.6	6.5	2.9	13.4	20.6	15.6	9.6	6.9
Stream 01	11.9	2.3	82.5	9.5	24.4	10.5	2.9	78.2	42.3	18.3	8.8	5.0
Stream 02	17.0	3.1	72.1	16.3	29.7	6.5	7.6	19.4	28.0	21.9	8.6	3.4
Stream 03	12.7	1.9	26.7	7.0	25.3	11.1	70.5	18.6	33.2	31.2	7.8	5.5
Stream 04	14.6	4.8	26.3	7.9	19.0	6.9	3.8	22.8	40.3	23.9	10.2	6.5
Stream 05	14.6	3.4	41.4	10.9	22.4	10.6	3.1	15.9	44.8	20.4	6.7	2.7
Stream 06	13.5	3.9	43.5	8.5	16.1	90.0	2.4	21.4	16.5	34.4	9.0	6.5
Minimum	8.7	1.9	7.9	7.0	11.6	6.5	2.4	13.4	16.5	15.6	6.7	2.7
Average	13.3	3.3	42.9	9.6	21.2	20.3	13.3	27.1	32.2	23.7	8.7	5.2
Maximum	17.0	4.8	82.5	16.3	29.7	90.0	70.5	78.2	44.8	34.4	10.2	6.9

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
Physical Organization.....	15
2.2. Horizontal Partitioning.....	15
2.3. 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	17
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.....	22
5.4. RAID Feature.....	23
5.5. Modifications to the DBGEN.....	23
5.6. Database Load Time.....	23
5.7. Data Storage Ratio.....	23
5.8. Database Load Mechanism Details and Illustration.....	24
6. Clause 5 Performance Metrics and Execution Rules.....	25

6.1. System Activity Between Load and Performance Tests.....	25
6.2. Steps in the Power Test.....	25
6.3. Timing Intervals for Each Query and Refresh Functions.....	25
6.4. Number of Streams for the Throughput Test.....	25
6.5. Start and End Date/Times for Each Query Stream.....	25
6.6. Total Elapsed Time of the Measurement Interval.....	26
6.7. Refresh Function Start Date/Time and Finish Date/Time.....	26
6.8. Timing Intervals for Each Query and Each Refresh Function for Each Stream.....	26
6.9. Performance Metrics.....	27
6.10. The Performance Metric and Numerical Quantities from Both Runs.....	27
6.11. System Activity Between Performance Tests.....	27
7. Clause 6 SUT and Driver Implementation.....	28
7.1. Driver	28
7.2. Implementation-Specific Layer.....	28
7.3. Profile-Directed Optimization.....	28
8. Clause 7 Pricing.....	29
8.1. Hardware and Software Used.....	29
8.2. Total Three Year Price	29
8.3. Availability Date.....	29
9. Auditor's Information and Attestation Letter.....	30
Appendix A. ParAccel and Linux Parameters.....	31
Appendix B. Programs and Scripts.....	31
Appendix C. Query Text and Query Output.....	40
Appendix D. Seed and Query Substitution Parameters.....	55
Appendix E. Implementation-Specific Layer/Driver Code.....	56
Appendix F. Misc database scripts.....	71
Appendix G. Pricing information.....	77

- The database was properly scaled to 1000GB 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 6 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. Three system crash tests were performed. One for a compute node (every node in the cluster), one for the leader node (one node in the cluster), and one with the full 5-node load protected by each UPS. The 5-node crash test included the leader node. The third crash test was executed on later DBMS binaries. Based on the successful execution of all the other ACID tests with the measured binaries, it is my opinion that the durability requirements have been demonstrated.

Respectfully Yours,

A handwritten signature in black ink, appearing to read "François Raab", with a long horizontal flourish extending to the right.

François Raab, President

TPC Benchmark H Overview

The TPC Benchmark™ 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*

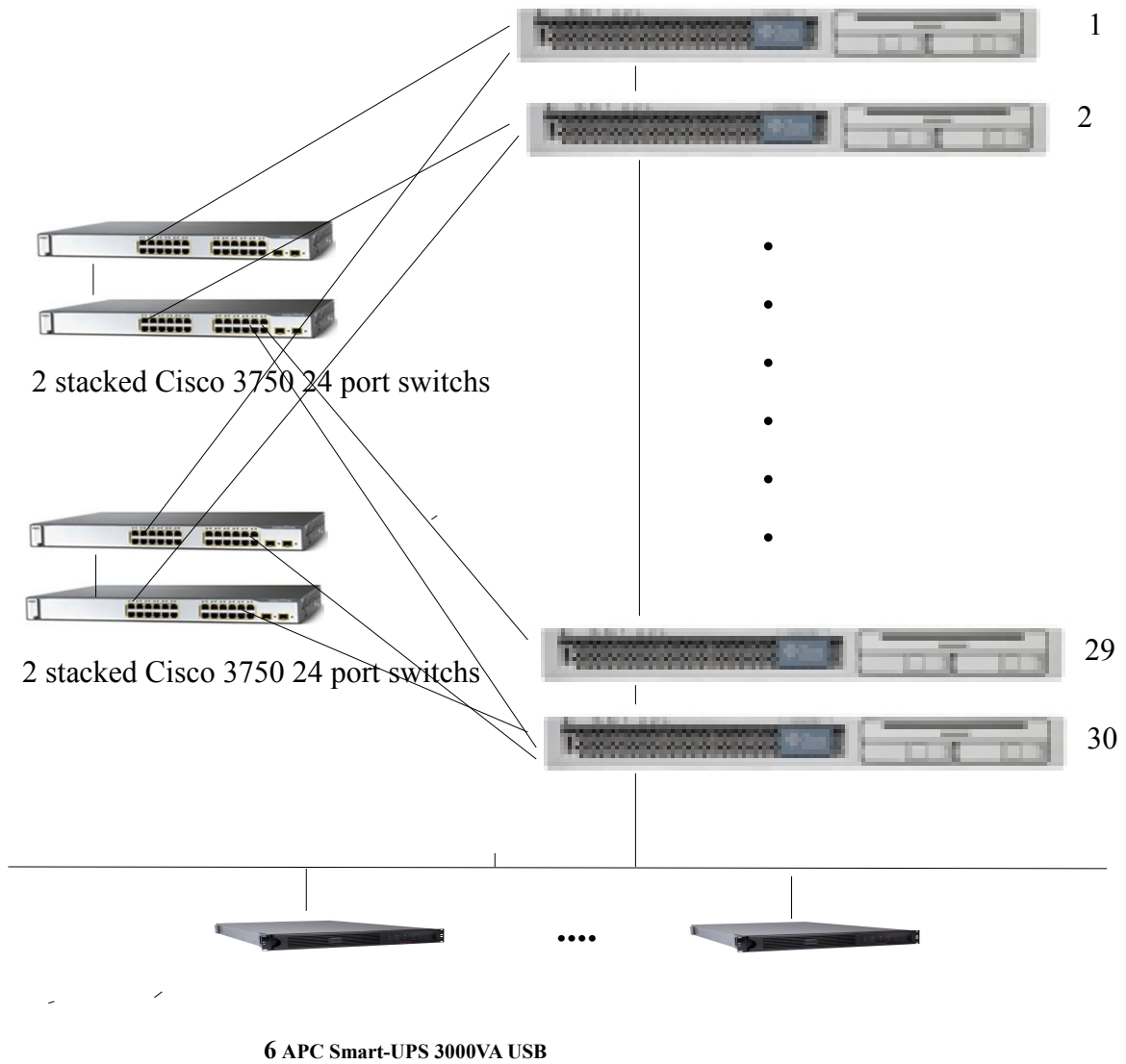
Appendix A contains the Linux and ParAccel Analytic Database™ parameters used in this benchmark.

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:

- | | | |
|--|-------------|---|
| 30 SunFire X4100 Servers (each with): | Plus | 4 Cisco 3750 24 Port Switches (Stacked 2 at a time) |
| ● 2 X 2.8 GHz AMD Optron Processors | and | 4 APC Smart-UPS 3000VA USB |
| ● 16 GB Memory | | |
| ● 2 X 143 GB Internal disks | | |



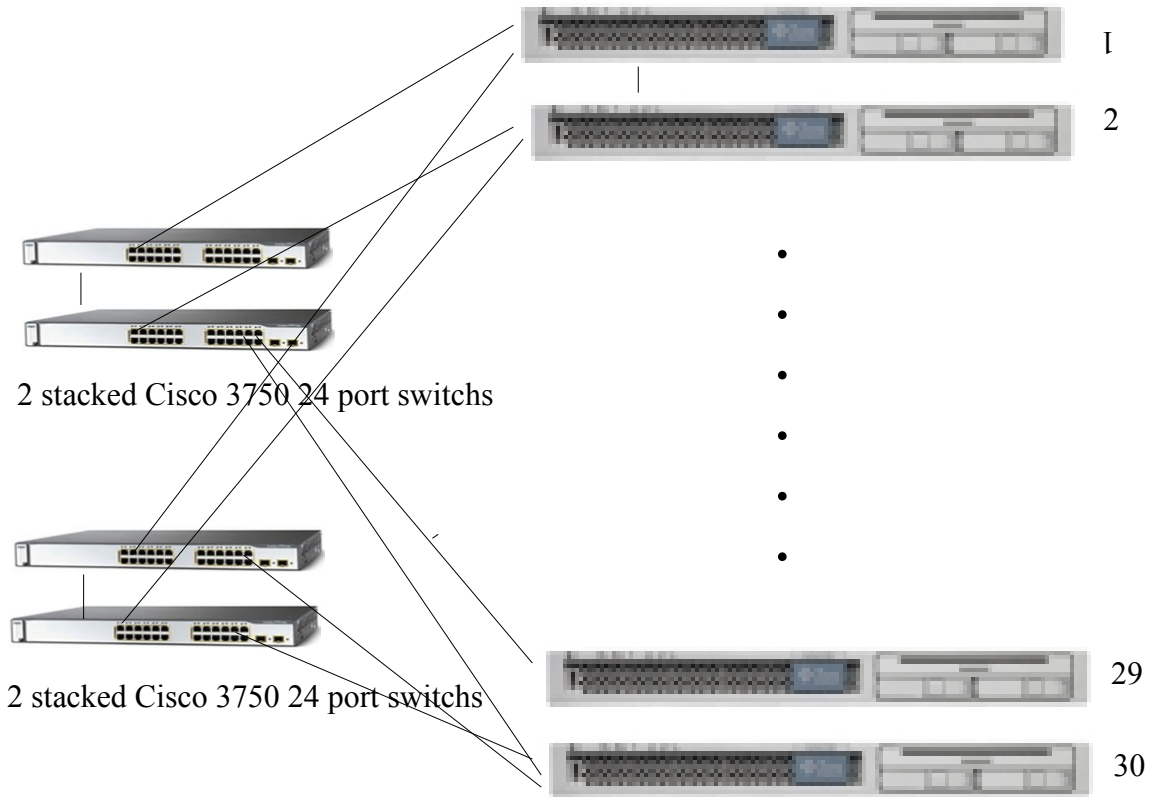
The measured configuration is as follows:

30 SunFire X4100 Servers (each with):

And

4 Cisco 3750 24 Port Switches (Stacked 2 at a time)

- 2 X 2.8 Ghz AMD Opteron Porcessors
- 16 GB Memory
- 2 X 143 GB Internal disks



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.

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.

2.2. 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.

2.3. 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 2.6.0 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 2.6.0 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 Q7, Q8 and Q9, the "date_part" function is used to extract part of a date (e.g., "year").
- 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 nine 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 $T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE + (\Delta T1*(T1.L_EXTENDEDPRICE/T1.L_QUANTITY))$

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 X4100 servers. The system continued to work as if nothing had happened, as is to be expected when 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.

Three system crash tests were performed. One for a compute node (every node in the cluster), one for the leader node (one node in the cluster), and one with the full 5-node load protected by each UPS. The 5-node crash test included the leader node.

In each case power was cut off to the appropriate node, or nodes. When power was restored, the 5-node cluster automatically rebooted. 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).

Each cluster node was protected by a UPS, which results in limiting the single point of failure of memory 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.

<i>Table</i>	<i>Rows</i>
<i>Lineitem</i>	<i>1,799,989,091</i>
<i>Orders</i>	<i>450,000,000</i>
<i>Partsupp</i>	<i>240,000,000</i>
<i>Part</i>	<i>60,000,000</i>
<i>Customer</i>	<i>45,000,000</i>
<i>Supplier</i>	<i>3,000,000</i>
<i>Nation</i>	<i>25</i>
<i>Region</i>	<i>5</i>

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 1 RAID 1 volume per server. The volume was constructed using the system RAID Controller.

The following table shows all the disk slices.

Partition	Use	Size (in GB)
sda1	Boot partition	0.1
sda2	Root; data	142
sda3	swap	2

- Secondary swap area was created in /swap/swapfile with size = 16GB

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 (or the first column if no primary key is specified) 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 2.6.0 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 =0Hrs15Min05sec

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:

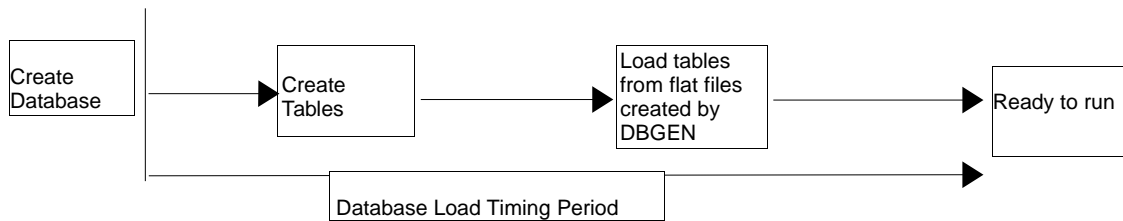
Disk Type	# of Disks	GB* per disk	Total Disk Space in GB**
Internal	60	146	8158.39
		Total Space	8158.39
		Data Storage Ratio	27.19

* 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 rebooted 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.

6 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:

S t r e a m I D	R e f r e s h F u n c t i o n	S t a r t D a t e	S t a r t T i m e	E n d D a t e	E n d T i m e
s t r e a m 0 1	RF1	2007-10-21	22:19:40	2007-10-21	22:19:48
s t r e a m 0 1	RF2	2007-10-21	22:19:48	2007-10-21	22:19:53
s t r e a m 0 2	RF1	2007-10-21	22:19:53	2007-10-21	22:20:02
s t r e a m 0 2	RF2	2007-10-21	22:20:02	2007-10-21	22:20:05
s t r e a m 0 3	RF1	2007-10-21	22:20:05	2007-10-21	22:20:13
s t r e a m 0 3	RF2	2007-10-21	22:20:13	2007-10-21	22:20:19
s t r e a m 0 4	RF1	2007-10-21	22:20:19	2007-10-21	22:20:29
s t r e a m 0 4	RF2	2007-10-21	22:20:29	2007-10-21	22:20:35
s t r e a m 0 5	RF1	2007-10-21	22:20:35	2007-10-21	22:20:42
s t r e a m 0 5	RF2	2007-10-21	22:20:42	2007-10-21	22:20:45
S t r e a m 0 6	RF1	2007-10-21	22:20:45	2007-10-21	22:20:54
S t r e a m 0 6	RF2	2007-10-21	22:20:54	2007-10-21	22:21:00

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	QppH@ 300GB	QthH@ 300GB	QphH@ 300GB
Run 1	145,219.3	271,542.9	198,578.1
Run 2	192,978.9	275,212.4	230,456.5
% Difference	24.75%	1.33%	13.83%

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 ntest shellsript.

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

The scripts that perform the query streams within the power and throughput tests are generated by the gen_streams script (which in turn invokes QGEN to generate the actual query stream files).

The Power Test is performed when ntest executes update_power.sql, which in turn runs the refresh functions and the power stream queries.

The Throughput Test is performed when ntest concurrently executes the 6 query stream scripts, stream[1-6].sql, together with the script update_throughput6.sql. The latter starts the refresh stream.

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 Postgres -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 \$624,770.12

. 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 Continental Resources 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 October 29, 2007.

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 Linux Parameters

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

ParAccel Server Configuration Parameters

(altered from default)

not applicable

Linux Parameters

(altered from default)

/boot/grub/grub.conf

Added `acpi=off` to kernel boot parameters
Added `ramdisk_size=64000` to kernel boot parameters

The following kernel parameters were changed during the run time:

```
kernel.msgmni=1000
kernel.sem = 128000 1048576 32 3072
vm.dirty_writeback_centisecs = 2000
net.core.rmem_max = 8388608
net.core.rmem_default = 8388608
```

Appendix B. Programs and Scripts

```
=====
load_tpch.sh
=====
```

```
#!/bin/bash
source $HOME/.bashrc
```

```
psql dev -f create_tpch_schema.sql
```

```
echo XXXXXXXXXXXXXXXX load data
XXXXXXXXXXXXXXXXXXXXX
psql dev -c "copy customer from
'/tpch_data/customer.tbl' delimiter as '|' "
psql dev -c "copy lineitem from
'/tpch_data/lineitem.tbl' delimiter as '|' "
psql dev -c "copy orders from
'/tpch_data/orders.tbl' delimiter as '|' "
psql dev -c "copy part from
'/tpch_data/part.tbl' delimiter as '|' "
psql dev -c "copy partsupp from
'/tpch_data/partsupp.tbl' delimiter as '|' "
psql dev -c "copy supplier from
'/tpch_data/supplier.tbl' delimiter as '|' "
psql dev -c "copy nation from
'/tpch_data/nation.tbl' delimiter as '|' "
psql dev -c "copy region from
'/tpch_data/region.tbl' delimiter as '|' "
```

```
echo XXXXXXXXXXXXXXXX analyze all
XXXXXXXXXXXXXXXXXXXXX
psql dev -c "analyze"
```

```
=====
create_tpch_schema.sql
=====
```

create_tpch_schema.sql

```
BEGIN;
```

```
create table part (
  p_partkey int4 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,
  PRIMARY KEY (p_partkey)
);
```

```
create table region (
  r_regionkey int4 not null encode always8,
  r_name char(25) not null encode bytedict,
  r_comment varchar(152) not null,
  PRIMARY KEY (r_regionkey)
);
```

```
create table 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,
  PRIMARY KEY (n_nationkey),
  FOREIGN KEY (n_regionkey) REFERENCES region
(r_regionkey)
);
```

```
create table 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,
  PRIMARY KEY (s_suppkey),
  FOREIGN KEY (s_nationkey) REFERENCES nation
(n_nationkey)
);
```

```
create table partsupp (
  ps_partkey int4 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,
  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 int4 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,
PRIMARY KEY (c_custkey),
FOREIGN KEY (c_nationkey) REFERENCES nation
(n_nationkey)
);

```

```

create table orders (
o_orderkey int4 not null encode delta,
o_custkey int4 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,
PRIMARY KEY (o_orderkey),
FOREIGN KEY (o_custkey) REFERENCES customer
(o_custkey)
);

```

```

create table lineitem (
l_orderkey int4 not null encode delta,
l_partkey int4 not null,
l_suppkey int4 not null,
l_linenumbers 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,
FOREIGN KEY (l_orderkey) REFERENCES orders
(o_orderkey),
FOREIGN KEY (l_partkey,l_suppkey) REFERENCES
partsupp (ps_partkey,ps_suppkey)
);

```

```
COMMIT;
```

```
=====
createdb.sh
=====
```

```
#!/bin/sh
```

```
createdb tpch
```

```
=====
create_rf1.sql
=====
```

```

create or replace function rf1(id int) returns int as
$procedure$
declare
streamid integer;
str_sql varchar(500);
begin
streamid := id;

str_sql = 'copy orders from
\'/tpch_data/orders.tbl.u' || streamid ||
'\ ' with delimiter '\|\'';
execute str_sql;

str_sql = 'copy lineitem from
\'/tpch_data/lineitem.tbl.u' || streamid ||
'\ ' with delimiter '\|\'';

```

```
execute str_sql;
```

```

return streamid;
end;
$procedure$
language plpgsql;

```

```
=====
create_rf2.sql
=====
```

```

create or replace function rf2(id int) returns int as
$procedure$
declare
streamid integer;
str_sql varchar(500);
begin
streamid := id;

create table delete_ids (
d_orderkey int4 not null encode delta
);

str_sql = 'copy delete_ids from
\'/tpch_data/delete.' || streamid ||
'\ ' with delimiter '\|\'';
execute str_sql;

analyze delete_ids;

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);

drop table delete_ids;

return streamid;
end;
$procedure$
language plpgsql;

```

```
=====
load_region.sql
=====
```

```

select 'LOAD REGION START - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimestamp ;
copy region from '/tpch_data/region.tbl';
select 'LOAD REGION STOP - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimestamp ;

```

```
=====
load_nation.sql
=====
```

```

select 'LOAD NATION START - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimestamp ;
copy nation from '/tpch_data/nation.tbl';
select 'LOAD NATION STOP - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimestamp ;

```

```
=====
load_customer.sql
=====
```

```

select 'LOAD CUSTOMER START - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimestamp ;
copy customer from '/tpch_data/customer.tbl' WITH
PARALLEL;
select 'LOAD CUSTOMER STOP - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimestamp ;

```



```

=====
load_part.sql
=====

select 'LOAD PART START - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimeStamp ;
copy part from '/tpch_data/part.tbl' WITH PARALLEL;
select 'LOAD PART STOP - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimeStamp ;

```

```

=====
load_supplier.sql
=====

select 'LOAD SUPPLIER START - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimeStamp ;
copy supplier from '/tpch_data/supplier.tbl' WITH
PARALLEL;
select 'LOAD SUPPLIER STOP - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimeStamp ;

```

```

=====
load_partsupp.sql
=====

select 'LOAD PARTSUPP START - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimeStamp ;
copy partsupp from '/tpch_data/partsupp.tbl' WITH
PARALLEL;
select 'LOAD PARTSUPP STOP - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimeStamp ;

```

```

=====
load_orders.sql
=====

select 'LOAD ORDERS START - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimeStamp ;
copy orders from '/tpch_data/orders.tbl' WITH
PARALLEL;
select 'LOAD ORDERS STOP - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimeStamp ;

```

```

=====
load_lineitem.sql
=====

select 'LOAD LINEITEM START - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimeStamp ;
copy lineitem from '/tpch_data/lineitem.tbl' WITH
PARALLEL;
select 'LOAD LINEITEM STOP - ',
date_part('epoch',LOCALTIMESTAMP(3)), localtimeStamp ;

```

```

=====
create_tables_int.sh
=====
BEGIN;

-- Create the tpch tables with optimal compression.

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

create table 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 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 partsupp (
  ps_partkey int4 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 customer (
  c_custkey int4 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 orders (
  o_orderkey int4 not null encode delta,
  o_custkey int4 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 lineitem (
  l_orderkey int4 not null encode delta,
  l_partkey int4 not null,
  l_suppkey int4 not null,
  l_linenum 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
);

COMMIT;

```

```
#!/bin/bash
echo "-----"
echo "Running the 22 TPC-H queries"
echo "-----"
for query in 1 2 3 4 5 6 7 8 9 10 11 12 13
14 15 16 17 18 19 20 21 22; do psql dev -f
q$query.sql > q$query.out; done
echo
"-----"
echo "Finished running the 22 TPC-H
queries"
echo
"-----"
```

ACID Test Execution Code

acid_trans.sh

```
#!/bin/bash
#
# 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 dev <<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 ROUND( ROUND(L_EXTENDEDPRI * (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_EXTENDEDPRI = L_EXTENDEDPRI + ROUND(
ROUND(L_EXTENDEDPRI/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 ROUND( ROUND(L_EXTENDEDPRI * (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
```

acid_setup.sh

```
#!/bin/bash
#
# 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 dev <<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 dev <<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 dev <<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

=====
create_history.dll
=====

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
);

=====
atomicity test
=====

=====
acid_atomicity.sh
=====

#!/bin/bash
#
# 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 dev <<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 dev <<EOF12
select
L_EXTENDEDPRI 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 dev <<EOF13
select
L_EXTENDEDPRI 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 dev <<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 dev <<EOF22

```

```

select
    L_EXTENDEDPRI 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 dev <<EOF23
select
    L_EXTENDEDPRI 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 dev <<EOF24
select *
    from HISTORY;
\q
EOF24

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

=====
consistency test
=====

acid_consistency_main.sh
=====

#!/bin/bash
#
# 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
# 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
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 dev <<EOF10
    select O_ORDERKEY,
        O_TOTALPRICE,
        sum(round(round(L_EXTENDEDPRI*
            (1-L_DISCOUNT),2)*(1+L_TAX),2)),
            round(O_TOTALPRICE -
sum(round(round(L_EXTENDEDPRI*
            (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 &
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 dev <<EOF10
    select O_ORDERKEY,
        O_TOTALPRICE,
        sum(round(round(L_EXTENDEDPRI*
            (1-L_DISCOUNT),2)*(1+L_TAX),2)),
            round(O_TOTALPRICE -
sum(round(round(L_EXTENDEDPRI*
            (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_loop.sh

```

#!/bin/bash
#
# 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 linenummer 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`

```

isolation tests

acid_isolation_main1.sh

```

#!/bin/bash
#
# 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

```

#!/bin/bash
#
# 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

```

#!/bin/bash
#
# 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

```

#!/bin/bash
#
# 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

```

#!/bin/bash
#
# 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 dev <<EOF1
select L_PARTKEY
      from LINEITEM
      where L_ORDERKEY = $O_KEY
            and L_LINENUMBER = $L_KEY
\q
EOF1`

S_KEY=`psql -q -t dev <<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

```

#!/bin/bash
#
# 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

```

#!/bin/bash
#
# 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 dev <<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

```

iso5_query.sh

```

#!/bin/bash
#
# 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 dev <<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 dev <<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 dev <<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
=====
```

```
#!/bin/bash
#
# 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 dev <<EOF1
select L_EXTENDEDPRI, SLEEP(L_LINENUMBER*4)
  from LINEITEM
  where L_ORDERKEY = $O_KEY
        and L_LINENUMBER < 4
        \q
EOF1`
echo ISO6 Query -- L_EXTENDEDPRI = $PS_DATA

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

```
=====
durability test
=====
```

```
=====
acid_durability_main.sh
=====
```

```
#!/bin/bash
#
# TPC-H ACID Durability Main
#
# Copyright 2007 ParAccel, Inc
#
NUMOFKEYS=25
echo Durability Test -- Starts `date`
echo
./acid_setup.sh $NUMOFKEYS dur_orderkeys1.txt
./acid_setup.sh $NUMOFKEYS dur_orderkeys2.txt
./acid_setup.sh $NUMOFKEYS dur_orderkeys3.txt
./acid_setup.sh $NUMOFKEYS dur_orderkeys4.txt
./acid_setup.sh $NUMOFKEYS dur_orderkeys5.txt
./acid_setup.sh $NUMOFKEYS dur_orderkeys6.txt
./acid_setup.sh $NUMOFKEYS dur_orderkeys7.txt
./acid_setup.sh $NUMOFKEYS dur_orderkeys8.txt
./acid_setup.sh $NUMOFKEYS dur_orderkeys9.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 &
wait
echo Durability Test -- Transactions Completed
echo
# Get count from the history table
HIST_COUNT=`psql -q -t dev <<EOF10
select count(*)
  from HISTORY;
\q
EOF10`
```

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

```
=====
acid_durability_loop.sh
=====
```

```
#!/bin/bash
#
# 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 linenummer 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_verify.sh

```

#!/bin/bash
#
# TPC-H ACID Durability Main
#
# Copyright 2007 ParAccel, Inc
#
# Get count from the history table
HIST_COUNT=`psql -q -t dev <<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

# loop through smaple keys for verification
while read O_KEY LKEY DELTA
do
    psql -q -t dev <<EOF10
    select O_ORDERKEY,
           O_TOTALPRICE,
           sum(round(round(L_EXTENDEDPRI*
(1-L_DISCOUNT),2)*(1+L_TAX),2)),
           round(O_TOTALPRICE -
sum(round(round(L_EXTENDEDPRI*
(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

```

Disk Configuration Details

Each server had 2 internal disks configured from the BIOS as a RAID1 device using the on-board LSI RAID controller. This operation precedes the installation of Red Hat Linux. Once Red Hat is installed, most of the RAID1 device is used for a file system mounted on /home.

Appendix C. Query Text and Query Output

qualification query 1

```

-- using default substitutions
select 'Stream 0 Query 1 START - ',
date_part('epoch',LOCALTIMESTAMP(1));
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 '90 days' as date)
group by
    l_returnflag,
    l_linestatus

order by
    l_returnflag,
    l_linestatus;
select 'Stream 0 Query 1 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

Fri Oct 26 12:20:40 PDT 2007
 l_returnflag | l_linestatus | sum_qty |
sum_base_price | sum_disc_price | sum_charge
| avg_qty | avg_price | avg_disc | count_order
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
A | F | 37734107.00 |
56586554400.73 | 53758256749.600 | 55909064213.178 |
25.52 | 38273.12 | 0.04 | 1478493
N | F | 991417.00 |
1487504710.38 | 1413082157.986 | 1469649196.694 |
25.51 | 38284.46 | 0.05 | 38854
N | O | 74429437.00 |
111631834353.84 | 106051845806.199 | 110297934442.245
| 25.50 | 38249.32 | 0.04 | 2918531
R | F | 37719753.00 |
56568041380.90 | 53741292299.136 | 55889618108.977 |
25.50 | 38250.85 | 0.05 | 1478870
(4 rows)

```


6726 | e packages are around the special ideas.
special, pending foxes us|
9624.78 | Supplier#000009658 | ROMANIA
| 189657 | Manufacturer#1
oE9uBgEfSS4opIcepXyAYM,x | 29-748-876-
2014 | ronic asymptotes wake bravely final|
9612.94 | Supplier#000003228 | ROMANIA
| 120715 | Manufacturer#2
KDdpNKN3cWu7ZSrbdqp7AfSLxx,qWB | 29-325-784-
8187 | warhorses. quickly even deposits sublata
daringly ironic instructions. slyly blithe t|
9612.94 | Supplier#000003228 | ROMANIA
| 198189 | Manufacturer#4
KDdpNKN3cWu7ZSrbdqp7AfSLxx,qWB | 29-325-784-
8187 | warhorses. quickly even deposits sublata
daringly ironic instructions. slyly blithe t|
9571.83 | Supplier#000004305 | ROMANIA
| 179270 | Manufacturer#2
qNHZ7WmCzygWMPRDO9Ps | 29-973-481-
1831 | kly carefully express asymptotes. furiou|
9558.10 | Supplier#000003532 | UNITED
KINGDOM | 88515 | Manufacturer#4
| EOeuiiOn2l0VpTlGguufFDFsbNlp0lhpxHp | 33-152-
301-2164 | foxes. quickly even excuses use. slyly
special foxes nag bl|
9492.79 | Supplier#000005975 | GERMANY
| 25974 | Manufacturer#5
S6mIiTx82z7lV | 17-992-579-
4839 | arefully pending accounts. blithely regular
excuses boost carefully carefully ironic p|
9461.05 | Supplier#000002536 | UNITED
KINGDOM | 20033 | Manufacturer#1
| 8mmGbyzaU 7ZS2wJumTibypncu9pNkDc4FYA | 33-556-
973-5522 | . slyly regular deposits wake slyly.
furiously regular warthogs are.|
9453.01 | Supplier#000000802 | ROMANIA
| 175767 | Manufacturer#1
,6HYXB4uaHITmtMBj4Ak57Pd | 29-342-882-
6463 | gular frets. permanently special multipliers
believe blithely alongs|
9408.65 | Supplier#000007772 | UNITED
KINGDOM | 117771 | Manufacturer#4
| AiC5YAH,gnu0i7 | 33-152-
491-1126 | nag against the final requests. furiously
unusual packages cajole blit|
9359.61 | Supplier#000004856 | ROMANIA
| 62349 | Manufacturer#5
yh1 | 29-334-870-9731 | y
ironic theodolites. blithely sile|
9357.45 | Supplier#000006188 | UNITED
KINGDOM | 138648 | Manufacturer#1
| g801,ssP8wpTk4Hm | 33-583-
607-1633 | ously always regular packages. fluffily
even accounts beneath the furiously final pack|
9352.04 | Supplier#000003439 | GERMANY
| 170921 | Manufacturer#4
qYPDgoiBGhCYxjgC | 17-128-996-
4650 | according to the carefully bold ideas|
9312.97 | Supplier#000007807 | RUSSIA
| 90279 | Manufacturer#5
oGYMPCk9XHGB2PBfKRnHA | 32-673-872-
5854 | ecial packages among the pending, even requests
use regula|
9312.97 | Supplier#000007807 | RUSSIA
| 100276 | Manufacturer#5
oGYMPCk9XHGB2PBfKRnHA | 32-673-872-
5854 | ecial packages among the pending, even requests
use regula|
9280.27 | Supplier#000007194 | ROMANIA
| 47193 | Manufacturer#3
zhRUQkBSrFYxIAXTfInj vyGRQjeK | 29-318-454-
2133 | o beans haggle after the furiously unusual
deposits. carefully silent dolphins cajole carefully|
9274.80 | Supplier#000008854 | RUSSIA
| 76346 | Manufacturer#3
lxhLoOUM7I3mZlMknerw OSqdbb4QbGa | 32-524-148-
5221 | y. courts do wake slyly. carefully ironic
platelets haggle above the slyly regular the|

9249.35 | Supplier#000003973 | FRANCE
| 26466 | Manufacturer#1
d18GiDsL6Wm2IsGXM,RZf1jCsgZAOjNYVThTRP4 | 16-722-866-
1658 | uests are furiously. regular tithes through the
regular, final accounts cajole furiously above the q|
9249.35 | Supplier#000003973 | FRANCE
| 33972 | Manufacturer#1
d18GiDsL6Wm2IsGXM,RZf1jCsgZAOjNYVThTRP4 | 16-722-866-
1658 | uests are furiously. regular tithes through the
regular, final accounts cajole furiously above the q|
9208.70 | Supplier#000007769 | ROMANIA
| 40256 | Manufacturer#5
Ht7xS | 29-964-424-9649 | lites
was quickly above the furiously ironic requests. slyly
even foxes against the blithely bold |
9201.47 | Supplier#000009690 | UNITED
KINGDOM | 67183 | Manufacturer#5
| CB BnUTlmi5zdeE17R7 | 33-121-
267-9529 | e even, even foxes. blithely ironic
packages cajole regular packages. slyly final ide|
9192.10 | Supplier#000000115 | UNITED
KINGDOM | 85098 | Manufacturer#3
| nJ 2t0f7Ve,wLl,6WzGBJLNBUCKlsv | 33-597-
248-1220 | es across the carefully express accounts
boost caref|
9189.98 | Supplier#000001226 | GERMANY
| 21225 | Manufacturer#4
qsLCqSvLyZfuXIpjz | 17-725-903-
1381 | deposits. blithely bold excuses about the
slyly bold forges wake |
9128.97 | Supplier#000004311 | RUSSIA
| 146768 | Manufacturer#5
I8IjnXd7NSJRS594RxsRR0 | 32-155-440-
7120 | refully. blithely unusual asymptotes haggle |
9104.83 | Supplier#000008520 | GERMANY
| 150974 | Manufacturer#4
RqRVDgD0ER | 17-728-804-1793 | ly
J9 b4lvR2,3 | about the blithely ironic depths. slyly final
theodolites among the fluffily bold ideas print|
9101.00 | Supplier#000005791 | ROMANIA
| 128254 | Manufacturer#5
zub2zCV,jhHPQqi,P2INAJElzI n6cOEOXFG | 29-549-251-
5384 | ts. notornis detect blithely above the
carefully bold requests. blithely even package|
9094.57 | Supplier#000004582 | RUSSIA
| 39575 | Manufacturer#1
WB0XkCSG3r,mnQ n,h9VIxjrr9ARHFvKgMdf | 32-587-577-
1351 | jole. regular accounts sleep blithely frets.
final pinto beans play furiously past the |
8996.87 | Supplier#000004702 | FRANCE
| 102191 | Manufacturer#5
16-811-269-8946 | ickly final packages along the
express plat|
8996.14 | Supplier#000009814 | ROMANIA
| 139813 | Manufacturer#2
af005pg831PU4IDVmEylXZVqYZqzSDlYLAmR | 29-995-571-
8781 | dependencies boost quickly across the
furiously pending requests! unusual dolphins play sl|
8968.42 | Supplier#000010000 | ROMANIA
| 119999 | Manufacturer#5
aTGLEusCiL4F | 29-578-432-2146 | ly
PDBdv665XBjHPyCOB0i | regular foxes boost slyly. quickly special waters
boost carefully ironi|
8936.82 | Supplier#000007043 | UNITED
KINGDOM | 109512 | Manufacturer#1
| FVajceZInZdbJE6Z9XsRUxxUEpiwHDROXi,1Rz | 33-784-
177-8208 | efully regular courts. furiously|
8929.42 | Supplier#000008770 | FRANCE
| 173735 | Manufacturer#4
R7cG26TtXrHAP9 HckhfRi | 16-242-746-
9248 | cajole furiously unusual requests. quickly
stealthy requests are.|
8920.59 | Supplier#000003967 | ROMANIA
| 26460 | Manufacturer#1
eHoAXe62SY9 | 29-194-731-3944 | aters. express, pending
instructions sleep. brave, r|
8920.59 | Supplier#000003967 | ROMANIA
| 173966 | Manufacturer#2 | eHoAXe62SY9

29-194-731-3944 aters. express, pending instructions sleep. brave, r 8913.96 Supplier#000004603 UNITED KINGDOM 137063 Manufacturer#2 OUzlvMUR7n,utLxmPNeyKsf3T24OXskxB5 33-789- 255-7342 haggle slyly above the furiously regular pinto beans. even 8877.82 Supplier#000007967 FRANCE 167966 Manufacturer#5 A3pilBARM4nx6R,qrwFoRPU 16-442-147- 9345 ously foxes. express, ironic requests im 8862.24 Supplier#000003323 ROMANIA 73322 Manufacturer#3 lYcsC9FwBqk3ItL 29-736-951- 3710 ly pending ideas sleep about the furiously unu 8841.59 Supplier#000005750 ROMANIA 100729 Manufacturer#5 Erx3lAgu0g62iaHF9x50uMH4EgeN9hEG 29-344-502- 5481 gainst the pinto beans. fluffily unusual dependencies affix slyly even deposits. 8781.71 Supplier#000003121 ROMANIA 13120 Manufacturer#5 wNqTogx238ZYCamFb,50v,bj 4IbNFW9Bvw1xP 29-707-291- 5144 s wake quickly ironic ideas 8754.24 Supplier#000009407 UNITED KINGDOM 179406 Manufacturer#4 CHRCbkaWcf5B 33-903- 970-9604 e ironic requests. carefully even foxes above the furious 8691.06 Supplier#000004429 UNITED KINGDOM 126892 Manufacturer#2 k,BQms5UhoAF1B2Asi,fLib 33-964- 337-5038 efully express deposits kindle after the deposits. final 8655.99 Supplier#000006330 RUSSIA 193810 Manufacturer#2 UozlaENr0ytKe2w6CeIEWFWh iO3S8Rae70u 32-561-198- 3705 symptotes use about the express dolphins. requests use after the express platelets. final, ex 8638.36 Supplier#000002920 RUSSIA 75398 Manufacturer#1 Je2a8bszf3L 32-122-621-7549 ly quickly ironic requests. even requests without t 8638.36 Supplier#000002920 RUSSIA 170402 Manufacturer#3 Je2a8bszf3L 32-122-621-7549 ly quickly ironic requests. even requests without t 8607.69 Supplier#000006003 UNITED KINGDOM 76002 Manufacturer#2 EH9wADcEiuenM0NR08zDwMidw,52Y2RyILEiA 33-416- 807-5206 ar, pending accounts. pending depende 8569.52 Supplier#000005936 RUSSIA 5935 Manufacturer#5 jXaNz6vwnEWJ2ksLZJpjtgt0bY2a3AU 32-644-251- 7916 . regular foxes nag carefully atop the regular, silent deposits. quickly regular packages 8564.12 Supplier#000000033 GERMANY 110032 Manufacturer#1 gfeKpYw3400L0SDyWXA6YalQmqlw6YB9f3R 17-138-897- 9374 n sauternes along the regular asymptotes are regularly along the 8553.82 Supplier#000003979 ROMANIA 143978 Manufacturer#4 BfmVhCAnCMY3jzpJUmY4CNws9 HzpdQR7INJU 29-124-646- 4897 ic requests wake against the blithely unusual accounts. fluffily r 8517.23 Supplier#000009529 RUSSIA 37025 Manufacturer#5 e44R8o7JAIS9iMcr 32-565-297- 8775 ove the even courts. furiously special platelets 8517.23 Supplier#000009529 RUSSIA 59528 Manufacturer#2 e44R8o7JAIS9iMcr 32-565-297- 8775 ove the even courts. furiously special platelets 8503.70 Supplier#000006830 RUSSIA 44325 Manufacturer#4	BC4WFCYRUZyaIghU 4S 32-147-878- 5069 padés cajole. furious packages among the carefully express excuses boost furiously across th 8457.09 Supplier#000009456 UNITED KINGDOM 19455 Manufacturer#1 7SBhZs8gPlcJjT0Qf433YBk 33-858- 440-4349 cing requests along the furiously unusual deposits promise among the furiously unus 8441.40 Supplier#000003817 FRANCE 141302 Manufacturer#2 hU3fz3xL78 16-339-356-5115 ely even ideas. ideas wake slyly furiously unusual instructions. pinto beans sleep ag 8432.89 Supplier#000003990 RUSSIA 191470 Manufacturer#1 wehBBp1RQbfxAYDASS75msywnsKHRVdkrvNe6m 32-839-509- 9301 ep furiously. packages should have to haggle slyly across the deposits. furiously regu 8431.40 Supplier#000002675 ROMANIA 5174 Manufacturer#1 HJFStOu9R5NGPOegKhgbzBdyvrG2yh8w 29-474-643- 1443 ithely express pinto beans. blithely even foxes haggle. furiously regular theodol 8407.04 Supplier#000005406 RUSSIA 162889 Manufacturer#4 j7 gYF5RW8DC5UrjKC 32-626-152- 4621 r the blithely regular packages. slyly ironic theodoli 8386.08 Supplier#000008518 FRANCE 36014 Manufacturer#3 2jqzqqAVe9crMVGP,n9nTsQXuNLTYUyJjEDcqWV 16-618-780- 7481 blithely bold pains are carefully platelets. finally regular pinto beans sleep carefully special 8376.52 Supplier#000005306 UNITED KINGDOM 190267 Manufacturer#5 9t8Y8 QqSIsoADPt6NLdk,TP5zyRx41oBulgoGc9 33-632- 514-7931 ly final accounts sleep special, regular requests. furiously regular 8348.74 Supplier#000008851 FRANCE 66344 Manufacturer#4 nWxi7GwEbjhw1 16-796-240- 2472 boldly final deposits. regular, even instructions detect slyly. fluffily unusual pinto bea 8338.58 Supplier#000007269 FRANCE 17268 Manufacturer#4 ZwhJSwABUoiB04,3 16-267-277- 4365 iously final accounts. even pinto beans cajole slyly regular 8328.46 Supplier#000001744 ROMANIA 69237 Manufacturer#5 oLo3fv64q2,FKHa3p,qHnS7Yzv,ps8 29-330-728- 5873 ep carefully-- even, careful packages are slyly along t 8307.93 Supplier#000003142 GERMANY 18139 Manufacturer#1 dqblvV8dCNAorGlJ 17-595-447- 6026 olites wake furiously regular decoys. final requests nod 8231.61 Supplier#000009558 RUSSIA 192000 Manufacturer#2 mcdgen,yTliJDHDS5fv 32-762-137- 5858 foxes according to the furi 8152.61 Supplier#000002731 ROMANIA 15227 Manufacturer#4 nluXJCuY1tu 29-805-463-2030 special requests. even, regular warhorses affix among the final gr 8109.09 Supplier#000009186 FRANCE 99185 Manufacturer#1 wgfosrVPexl9pEXWyaqlBMDYYf 16-668-570- 1402 tions haggle slyly about the sil 8102.62 Supplier#000003347 UNITED KINGDOM 18344 Manufacturer#5 m CtXS2S16i 33-454- 274-8532 egrate with the slyly bold instructions. special foxes haggle silently among the 8046.07 Supplier#000008780 FRANCE 191227 Manufacturer#3 Aczue0UK9osj ,Lx0Jmh 16-473-215- 6395 onic platelets cajole after the regular
--	--

```

instructions. permanently bold excuses|
  8042.09 | Supplier#000003245 | RUSSIA
| 135705 | Manufacturer#4
Dh8Ikg39onrbOL4DyTfGw8a9oKUX3d9Y | 32-836-132-
8872 | osits. packages cajole slyly. furiously regular
deposits cajole slyly. q|
  8042.09 | Supplier#000003245 | RUSSIA
| 150729 | Manufacturer#1
Dh8Ikg39onrbOL4DyTfGw8a9oKUX3d9Y | 32-836-132-
8872 | osits. packages cajole slyly. furiously regular
deposits cajole slyly. q|
  7992.40 | Supplier#000006108 | FRANCE
| 118574 | Manufacturer#1
8tBydnTDwUqfBfFV413 | 16-974-998-
8937 | ironic ideas? fluffily even instructions wake.
blithel|
  7980.65 | Supplier#000001288 | FRANCE
| 13784 | Manufacturer#4
| 16-646-464-8247 | ully bold courts. escapades nag
slyly. furiously fluffy theodo|
  7950.37 | Supplier#000008101 | GERMANY
| 33094 | Manufacturer#5
kkYvL6IuvojJgTNG IKkaXQDYgx8ILohj | 17-627-663-
8014 | arefully unusual requests x-ray above the
quickly final deposits. |
  7937.93 | Supplier#000009012 | ROMANIA
| 83995 | Manufacturer#2
| 29-250-925-
9690 | to the blithely ironic deposits nag sly|
  7914.45 | Supplier#000001013 | RUSSIA
| 125988 | Manufacturer#2
riRcntps4KEDtYScjpmIWeYF6mNnR | 32-194-698-
3365 | busily bold packages are dolphi|
  7912.91 | Supplier#000004211 | GERMANY
| 159180 | Manufacturer#5
2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG | 17-266-947-
7315 | ay furiously regular platelets. cou|
  7912.91 | Supplier#000004211 | GERMANY
| 184210 | Manufacturer#4
2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG | 17-266-947-
7315 | ay furiously regular platelets. cou|
  7894.56 | Supplier#000007981 | GERMANY
| 85472 | Manufacturer#4
NSJ96vMROAbexP | 17-963-404-
3760 | ic platelets affix after the furiously|
  7887.08 | Supplier#000009792 | GERMANY
| 164759 | Manufacturer#3
Y28ITVeYriT3kIGdV2K8fSZ V2UqT5H10tz | 17-988-938-
4296 | ckly around the carefully fluffy theodolites.
slyly ironic pack|
  7871.50 | Supplier#000007206 | RUSSIA
| 104695 | Manufacturer#1
fNCnrVmvJjE95sgWZzvW | 32-432-452-
7731 | ironic requests. furiously final theodolites
cajole. final, express packages sleep. quickly reg|
  7852.45 | Supplier#000005864 | RUSSIA
| 8363 | Manufacturer#4
WCNfBPZeSXh3h,c | 32-454-883-
3821 | usly unusual pinto beans. brave ideas sleep
carefully quickly ironi|
  7850.66 | Supplier#000001518 | UNITED
KINGDOM | 86501 | Manufacturer#1
| ONda3YJiHKJOC | 33-730-
383-3892 | ifts haggle fluffily pending pai|
  7843.52 | Supplier#000006683 | FRANCE
| 11680 | Manufacturer#4
2Z0JGkiv01Y00oCFwUGfviIbhzcDy | 16-464-517-
8943 | express, final pinto beans x-ray slyly
asymptotes. unusual, unusual|
(100 rows)

```

qualification query 3

```

-- using default substitutions
select 'Stream 0 Query 3 START - ',

```

```

date_part('epoch',LOCALTIMESTAMP(1));
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 '1995-03-15'
  and l_shipdate > date '1995-03-15'
group by
  l_orderkey,
  o_orderdate,
  o_shippriority
order by
  revenue desc,
  o_orderdate
limit 10;
select 'Stream 0 Query 3 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

```

Fri Oct 26 12:20:49 PDT 2007

```

  l_orderkey | revenue | o_orderdate |
  o_shippriority
+-----+-----+-----+
0          2456423 | 406181.008 | 1995-03-05 |
0          3459808 | 405838.697 | 1995-03-04 |
0          492164 | 390324.058 | 1995-02-19 |
0          1188320 | 384537.934 | 1995-03-09 |
0          2435712 | 378673.054 | 1995-02-26 |
0          4878020 | 378376.794 | 1995-03-12 |
0          5521732 | 375153.920 | 1995-03-13 |
0          2628192 | 373133.307 | 1995-02-22 |
0          993600 | 371407.458 | 1995-03-05 |
0          2300070 | 367371.144 | 1995-03-13 |
(10 rows)

```

qualification query 4

```

-- using default substitutions
select 'Stream 0 Query 4 START - ',
date_part('epoch',LOCALTIMESTAMP(1));
select
  o_orderpriority,
  count(*) as order_count
from
  orders
where
  o_orderdate >= date '1993-07-01'
  and o_orderdate < cast(date '1993-07-01' +
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;
select 'Stream 0 Query 4 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

```

```

Fri Oct 26 12:20:53 PDT 2007
o_orderpriority | order_count
-----+-----
1-URGENT        |      10594
2-HIGH          |      10476
3-MEDIUM       |      10410
4-NOT SPECIFIED|      10556
5-LOW          |      10487
(5 rows)

```

qualification query 5

```

-- using default substitutions
select 'Stream 0 Query 5 START - ',
date_part('epoch',LOCALTIMESTAMP(1));
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 = 'ASIA'
    and o_orderdate >= date '1994-01-01'
    and o_orderdate < cast(date '1994-01-01' +
interval '1 year' as date)
group by
    n_name
order by
    revenue desc;
select 'Stream 0 Query 5 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

```

```

Fri Oct 26 12:20:58 PDT 2007
n_name | revenue
-----+-----
INDONESIA | 55502040.783
VIETNAM | 55295086.579
CHINA | 53724493.874
INDIA | 52035511.632
JAPAN | 45410175.353
(5 rows)

```

qualification query 6

```

-- using default substitutions
select 'Stream 0 Query 6 START - ',
date_part('epoch',LOCALTIMESTAMP(1));
select

```

```

sum(l_extendedprice * l_discount) as revenue
from
    lineitem
where
    l_shipdate >= date '1994-01-01'
    and l_shipdate < cast(date '1994-01-01' +
interval '1 year' as date)
    and l_discount between .06 - 0.01 and .06 +
0.01
    and l_quantity < 24;
select 'Stream 0 Query 6 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

```

```

Fri Oct 26 12:21:01 PDT 2007
revenue
-----
123141046.397
(1 row)

```

qualification query 7

```

-- using default substitutions
select 'Stream 0 Query 7 START - ',
date_part('epoch',LOCALTIMESTAMP(1));
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 = 'FRANCE'
                and n2.n_name = 'GERMANY')
                or (n1.n_name =
'GERMANY' and n2.n_name = 'FRANCE')
            )
            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;
select 'Stream 0 Query 7 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

```

```

Fri Oct 26 12:21:02 PDT 2007
supp_nation | cust_nation

```

```

| one_year | revenue
-----+-----+-----
FRANCE    | 1995 | 54639732.326 | GERMANY
FRANCE    | 1996 | 54633082.905 | GERMANY
GERMANY   | 1995 | 52531746.305 | FRANCE
GERMANY   | 1996 | 52520548.643 | FRANCE
(4 rows)

```

qualification query 8

```

-- using default substitutions
select 'Stream 0 Query 8 START - ',
date_part('epoch',LOCALTIMESTAMP(1));
select
    o_year,
    sum(case
        when nation = 'BRAZIL' 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 = 'AMERICA'
            and s_nationkey = n2.n_nationkey
            and o_orderdate between date
'1995-01-01' and date '1996-12-31'
            and p_type = 'ECONOMY ANODIZED
STEEL'
    ) as all_nations
group by
    o_year
order by
    o_year;
select 'Stream 0 Query 8 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

```

```

Fri Oct 26 12:21:06 PDT 2007
o_year | mkt_share
-----+-----
1995   | 0.034
1996   | 0.041
(2 rows)

```

qualification query 9

```

-- using default substitutions
select 'Stream 0 Query 9 START - ',
date_part('epoch',LOCALTIMESTAMP(1));
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 '%green%'
    ) as profit
group by
    nation,
    o_year
order by
    nation,
    o_year desc;
select 'Stream 0 Query 9 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

```

```

Fri Oct 26 12:21:10 PDT 2007
nation | o_year | sum_profit
-----+-----+-----
ALGERIA | 1998 | 31342866.889
ALGERIA | 1997 | 57138192.394
ALGERIA | 1996 | 56140139.529
ALGERIA | 1995 | 53051469.075
ALGERIA | 1994 | 53867581.552
ALGERIA | 1993 | 54942717.541
ALGERIA | 1992 | 54628034.118
ARGENTINA | 1998 | 30211185.393
ARGENTINA | 1997 | 50805741.197
ARGENTINA | 1996 | 51923745.958
ARGENTINA | 1995 | 49298625.208
ARGENTINA | 1994 | 50835609.558
ARGENTINA | 1993 | 51646078.639
ARGENTINA | 1992 | 50410314.437
BRAZIL | 1998 | 27217924.097
BRAZIL | 1997 | 48378668.692
BRAZIL | 1996 | 50482869.810
BRAZIL | 1995 | 47623383.090
BRAZIL | 1994 | 47840165.201
BRAZIL | 1993 | 49054693.496
BRAZIL | 1992 | 48667638.538
CANADA | 1998 | 30379833.429
CANADA | 1997 | 50465051.759
CANADA | 1996 | 52560500.816
CANADA | 1995 | 52375332.217
CANADA | 1994 | 52600364.071
CANADA | 1993 | 52644503.504
CANADA | 1992 | 53932871.108
CHINA | 1998 | 31075465.835
CHINA | 1997 | 50551873.863
CHINA | 1996 | 51039293.301
CHINA | 1995 | 49287534.089
CHINA | 1994 | 50851089.509
CHINA | 1993 | 54229629.248

```

CHINA	1992	52400528.817	MOROCCO	1998	26732115.291
EGYPT	1998	29054433.066	MOROCCO	1997	45637303.728
EGYPT	1997	50627610.886	MOROCCO	1996	45558221.225
EGYPT	1996	49542212.297	MOROCCO	1995	47851318.369
EGYPT	1995	48311549.804	MOROCCO	1994	46272172.435
EGYPT	1994	49790644.183	MOROCCO	1993	46764325.682
EGYPT	1993	48904292.455	MOROCCO	1992	48122783.054
EGYPT	1992	49434932.102	MOZAMBIQUE	1998	30712391.666
ETHIOPIA	1998	28040716.967	MOZAMBIQUE	1997	50316528.195
ETHIOPIA	1997	47455009.312	MOZAMBIQUE	1996	51640319.703
ETHIOPIA	1996	46491097.047	MOZAMBIQUE	1995	50693773.954
ETHIOPIA	1995	46804448.799	MOZAMBIQUE	1994	49253277.074
ETHIOPIA	1994	48516143.387	MOZAMBIQUE	1993	49153015.990
ETHIOPIA	1993	46551891.053	MOZAMBIQUE	1992	48247551.307
ETHIOPIA	1992	44934648.149	PERU	1998	29326101.993
FRANCE	1998	32226407.492	PERU	1997	49753779.886
FRANCE	1997	47121485.329	PERU	1996	50935169.755
FRANCE	1996	47263134.945	PERU	1995	53309882.825
FRANCE	1995	47275997.042	PERU	1994	50643531.261
FRANCE	1994	47067208.801	PERU	1993	51584621.444
FRANCE	1993	51163369.541	PERU	1992	47523898.551
FRANCE	1992	47846234.804	ROMANIA	1998	30368667.103
GERMANY	1998	28624942.322	ROMANIA	1997	50365683.299
GERMANY	1997	49309074.315	ROMANIA	1996	49598998.485
GERMANY	1996	49918682.592	ROMANIA	1995	47537642.362
GERMANY	1995	52650718.170	ROMANIA	1994	51455282.460
GERMANY	1994	50346899.880	ROMANIA	1993	50407136.362
GERMANY	1993	50991895.248	ROMANIA	1992	48185384.595
GERMANY	1992	48274125.547	RUSSIA	1998	28322383.695
INDIA	1998	29943144.024	RUSSIA	1997	50106684.652
INDIA	1997	50665452.666	RUSSIA	1996	51753341.874
INDIA	1996	50283091.725	RUSSIA	1995	49215819.824
INDIA	1995	50006774.082	RUSSIA	1994	52205665.908
INDIA	1994	48995190.183	RUSSIA	1993	51860229.474
INDIA	1993	50286902.317	RUSSIA	1992	53251676.597
INDIA	1992	50850328.817	SAUDI ARABIA	1998	31541259.465
INDONESIA	1998	27672339.686	SAUDI ARABIA	1997	52438750.237
INDONESIA	1997	50512145.161	SAUDI ARABIA	1996	52543737.226
INDONESIA	1996	51653059.532	SAUDI ARABIA	1995	52938695.945
INDONESIA	1995	51508779.076	SAUDI ARABIA	1994	51389601.398
INDONESIA	1994	52817949.774	SAUDI ARABIA	1993	52937508.286
INDONESIA	1993	47959994.412	SAUDI ARABIA	1992	54843459.052
INDONESIA	1992	51776604.479	UNITED KINGDOM	1998	28494873.700
IRAN	1998	29065735.920	UNITED KINGDOM	1997	49381810.388
IRAN	1997	50042062.499	UNITED KINGDOM	1996	51386853.452
IRAN	1996	50926652.629	UNITED KINGDOM	1995	51509586.290
IRAN	1995	51249667.084	UNITED KINGDOM	1994	48086499.223
IRAN	1994	50337085.302	UNITED KINGDOM	1993	49166826.669
IRAN	1993	51730762.912	UNITED KINGDOM	1992	49349121.564
IRAN	1992	49955856.006	UNITED STATES	1998	25126238.670
IRAQ	1998	31624550.653	UNITED STATES	1997	50077305.890
IRAQ	1997	55121748.421	UNITED STATES	1996	48048648.965
IRAQ	1996	55897663.192	UNITED STATES	1995	48809031.903
IRAQ	1995	54815471.949	UNITED STATES	1994	49296746.647
IRAQ	1994	54408515.530	UNITED STATES	1993	48029946.281
IRAQ	1993	53633167.365	UNITED STATES	1992	48671943.965
IRAQ	1992	55891938.725	VIETNAM	1998	30442735.719
JAPAN	1998	27934179.368	VIETNAM	1997	50309179.241
JAPAN	1997	44517162.072	VIETNAM	1996	50488160.843
JAPAN	1996	42545605.606	VIETNAM	1995	49658284.077
JAPAN	1995	43749355.885	VIETNAM	1994	50596056.729
JAPAN	1994	44840242.553	VIETNAM	1993	50953918.602
JAPAN	1993	44660015.010	VIETNAM	1992	49613837.757
JAPAN	1992	45410248.603			
JORDAN	1998	26901488.310	(175 rows)		
JORDAN	1997	45471877.874			
JORDAN	1996	46794325.296			
JORDAN	1995	45178828.074			
JORDAN	1994	45333636.002			
JORDAN	1993	47971495.572			
JORDAN	1992	44717238.686			
KENYA	1998	28597614.034			
KENYA	1997	47949733.212			
KENYA	1996	46886924.097			
KENYA	1995	46072338.223			
KENYA	1994	45772060.694			
KENYA	1993	46308727.733			
KENYA	1992	47257780.303			

=====
qualification query 10
=====

```
-- using default substitutions
select 'Stream 0 Query 10 START - ',
date_part('epoch',LOCALTIMESTAMP(1));
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 '1993-10-01'
  and o_orderdate < cast(date '1993-10-01' +
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;
select 'Stream 0 Query 10 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

```

```

Fri Oct 26 12:21:14 PDT 2007
 c_custkey |      c_name      | revenue |
 c_acctbal |      n_name      |         |
 c_address  |      c_phone     |         |
 c_comment  |                  |         |
-----+-----+-----
+-----+-----+-----
+-----+-----+-----
+-----+-----+-----
+-----+-----+-----
-----
57040 | Customer#000057040 | 734235.242 |
632.87 | JAPAN              |           | Eioyzjf4pp
| 22-895-641-3466 | sits. slyly regular requests sleep
alongside of the regular inst|
143347 | Customer#000143347 | 721002.690 |
2557.47 | EGYPT              |           | laReFYv,Kw4
| 14-742-935-3718 | ggle carefully enticing requests.
final deposits use bold, bold pinto beans. ironic,
idle re|
60838 | Customer#000060838 | 679127.302 |
2454.77 | BRAZIL             |           |
64EaJ5vMAHWJlBOxJklpNc2RjWE | 12-913-494-
9813 | need to boost against the slyly regular
account|
101998 | Customer#000101998 | 637029.565 |
3790.89 | UNITED KINGDOM    |           |
01c9CILnNtfoQYmZj | 33-593-865-
6378 | ress foxes wake slyly after the bold excuses.
ironic platelets are furiously carefully bold
theodolites|
125341 | Customer#000125341 | 633508.083 |
4983.51 | GERMANY           |           |
S29ODD6bceU8QsueEJznkNaK | 17-582-695-
5962 | arefully even depths. blithely even excuses
sleep furiously. foxes use except the dependencies.
ca|
25501 | Customer#000025501 | 620269.784 |
7725.04 | ETHIOPIA          |           |
W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ | 15-874-808-
6793 | he pending instructions wake carefully at the
pinto beans. regular, final instructions along the
slyly fina|

```

```

115831 | Customer#000115831 | 596423.863 |
5098.10 | FRANCE            |           | rFeBbEEyk dl
ne7zV5fDrmiqlOk09wV7pxqCgIc | 16-715-386-3788 | l
somas sleep. furiously final deposits wake blithely
regular pinto b|
84223 | Customer#000084223 | 594998.019 |
528.65 | UNITED KINGDOM    |           | nAVZCs6BaWap
rrM27N 2qBnzc5WBauxbA | 33-442-824-8191 | slyly
final deposits haggle regular, pending dependencies.
pending escapades wake |
54289 | Customer#000054289 | 585603.388 |
5583.02 | IRAN              |           | vXCxoCsU0Bad5JQI
,oobkZ | 20-834-292-4707 | ely
special foxes are quickly finally ironic p|
39922 | Customer#000039922 | 584878.111 |
7321.11 | GERMANY           |           |
Zgy4s5012GKN4pLDPBU8m342gIw6R | 17-147-757-
8036 | y final requests. furiously final foxes cajole
blithely special platelets. f|
6226 | Customer#000006226 | 576783.758 |
2230.09 | UNITED KINGDOM    |           |
8gPu8,NPGkfyQQ0hcIYUGPIBwc,ybP5g, | 33-657-701-
3391 | ending platelets along the express deposits
cajole carefully final |
922 | Customer#000000922 | 576767.529 |
3869.25 | GERMANY           |           |
Az9RFaut7NkPnc5zSD2PwHgVwr4jRzq | 17-945-916-
9648 | luffily fluffy deposits. packages c|
147946 | Customer#000147946 | 576455.128 |
2030.13 | ALGERIA           |           |
iANyZHjghyy7Ajah0pTrYyhJ | 10-886-956-
3143 | ithely ironic deposits haggle blithely ironic
requests. quickly regu|
115640 | Customer#000115640 | 569341.190 |
6436.10 | ARGENTINA         |           | Vtgfia9qI
7EpHgecUlX | 11-411-543-4901 | ost
slyly along the patterns; pinto be|
73606 | Customer#000073606 | 568656.856 |
1785.67 | JAPAN            |           |
xuR0Tro5yChdFOCrjkd2ol | 22-437-653-
6966 | he furiously regular ideas. slowly|
110246 | Customer#000110246 | 566842.978 |
7763.35 | VIETNAM          |           | 7KzflgX
MDOq7sOkI | 31-943-426-9837 |
egular deposits serve blithely above the fl|
142549 | Customer#000142549 | 563537.233 |
5085.99 | INDONESIA        |           |
ChqEoK430ysjdHbtKCP6dKqjNyyvvi9 | 19-955-562-
2398 | sleep pending courts. ironic deposits against
the carefully unusual platelets cajole carefully
express accounts.|
146149 | Customer#000146149 | 557254.984 |
1791.55 | ROMANIA           |           | s87fvzFQpU
| 29-744-164-6487 | of the slyly silent accounts.
quickly final accounts across the |
52528 | Customer#000052528 | 556397.348 |
551.79 | ARGENTINA         |           | NFztyTOR10UOJ
| 11-208-192-3205 | deposits hinder. blithely pending
asymptotes breach slyly regular re|
23431 | Customer#000023431 | 554269.533 |
3381.86 | ROMANIA           |           |
HgiV0phqhaIa9aydNoIlb | 29-915-458-
2654 | nusual, even instructions: furiously stealthy
n|
(20 rows)

```

qualification query 11

```

-- using default substitutions
select 'Stream 0 Query 11 START - ',
date_part('epoch',LOCALTIMESTAMP(1));
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 = 'GERMANY'
group by
    ps_partkey having
        sum(ps_supplycost * ps_availqty) > (
            select
                sum(ps_supplycost *
                    ps_availqty) * 0.0001000000
            from
                partsupp,
                supplier,
                nation
            where
                ps_suppkey = s_suppkey
                and s_nationkey =
                    n_nationkey
                and n_name = 'GERMANY'
        )
order by
    value desc;
select 'Stream 0 Query 11 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

```

```

Fri Oct 26 12:21:17 PDT 2007
ps_partkey | value
-----+-----
129760 | 17538456.86
166726 | 16503353.92
191287 | 16474801.97
161758 | 16101755.54
34452 | 15983844.72
139035 | 15907078.34
9403 | 15451755.62
154358 | 15212937.88
38823 | 15064802.86
85606 | 15053957.15
33354 | 14408297.40

```

lines deleted

```

113808 | 7893353.88
27901 | 7892952.00
128820 | 7892882.72
25891 | 7890511.20
122819 | 7888881.02
154731 | 7888301.33
101674 | 7879324.60
51968 | 7879102.21
72073 | 7877736.11
5182 | 7874521.73
(1048 rows)

```

qualification query 12

```

-- using default substitutions
select 'Stream 0 Query 12 START - ',
date_part('epoch',LOCALTIMESTAMP(1));
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 ('MAIL', 'SHIP')
    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;
select 'Stream 0 Query 12 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

```

Fri Oct 26 12:21:20 PDT 2007

l_shipmode	high_line_count	low_line_count
MAIL	6202	9324
SHIP	6200	9262

(2 rows)

qualification query 13

```

-- using default substitutions
select 'Stream 0 Query 13 START - ',
date_part('epoch',LOCALTIMESTAMP(1));
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
            '%special%requests%'
        group by
            c_custkey
    ) as c_orders (c_custkey, c_count)
group by
    c_count
order by
    custdist desc,
    c_count desc;
select 'Stream 0 Query 13 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

```

Fri Oct 26 12:21:22 PDT 2007

c_count	custdist
0	50005
9	6641
10	6532
11	6014
8	5937


```

'%Customer%Complaints%'
)
group by
    p_brand,
    p_type,
    p_size
order by
    supplier_cnt desc,
    p_brand,
    p_type,
    p_size;
select 'Stream 0 Query 16 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

```

```

Fri Oct 26 12:21:30 PDT 2007
p_brand | p_type | p_size |
supplier_cnt
-----+-----+-----+
Brand#41 | MEDIUM BRUSHED TIN | 3 |
28
Brand#54 | STANDARD BRUSHED COPPER | 14 |
27
Brand#11 | STANDARD BRUSHED TIN | 23 |
24
Brand#11 | STANDARD BURNISHED BRASS | 36 |
24
Brand#15 | MEDIUM ANODIZED NICKEL | 3 |
24
Brand#15 | SMALL ANODIZED BRASS | 45 |
24
Brand#15 | SMALL BURNISHED NICKEL | 19 |
24
Brand#21 | MEDIUM ANODIZED COPPER | 3 |
24
Brand#22 | SMALL BRUSHED NICKEL | 3 |
24
Brand#22 | SMALL BURNISHED BRASS | 19 |
24

```

lines deleted

```

Brand#33 | SMALL ANODIZED BRASS | 9 |
3
Brand#35 | MEDIUM ANODIZED TIN | 19 |
3
Brand#51 | SMALL PLATED BRASS | 23 |
3
Brand#52 | MEDIUM BRUSHED BRASS | 45 |
3
Brand#53 | MEDIUM BRUSHED TIN | 45 |
3
Brand#54 | ECONOMY POLISHED BRASS | 9 |
3
Brand#55 | PROMO PLATED BRASS | 19 |
3
Brand#55 | STANDARD PLATED TIN | 49 |
3
(18314 rows)

```

qualification query 17

```

-- using default substitutions
select 'Stream 0 Query 17 START - ',
date_part('epoch',LOCALTIMESTAMP(1));
select
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#23'

```

```

and p_container = 'MED BOX'
and l_quantity < (
    select
        0.2 * avg(l_quantity)
    from
        lineitem
    where
        l_partkey = p_partkey
);
select 'Stream 0 Query 17 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

```

```

Fri Oct 26 12:21:36 PDT 2007
avg_yearly
-----
348406.054
(1 row)

```

qualification query 18

```

-- using default substitutions
select 'Stream 0 Query 18 START - ',
date_part('epoch',LOCALTIMESTAMP(1));
select
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice,
    sum(l_quantity)
from
    customer,
    orders,
    lineitem
where
    o_orderkey in (
        select
            l_orderkey
        from
            lineitem
        group by
            l_orderkey having
                sum(l_quantity) > 300
    )
    and c_custkey = o_custkey
    and o_orderkey = l_orderkey
group by
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice
order by
    o_totalprice desc,
    o_orderdate
limit 100;
select 'Stream 0 Query 18 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

```

```

Fri Oct 26 12:21:41 PDT 2007
c_name | c_custkey | o_orderkey |
o_orderdate | o_totalprice | sum
-----+-----+-----+-----+-----+
Customer#000128120 | 128120 | 4722021 | 1994-
04-07 | 544089.09 | 323.00
Customer#000144617 | 144617 | 3043270 | 1997-
02-12 | 530604.44 | 317.00
Customer#000013940 | 13940 | 2232932 | 1997-
04-13 | 522720.61 | 304.00
Customer#000066790 | 66790 | 2199712 | 1996-
09-30 | 515531.82 | 327.00

```

Customer#000046435	46435	4745607	1997-
07-03	508047.99	309.00	
Customer#000015272	15272	3883783	1993-
07-28	500241.33	302.00	
Customer#000146608	146608	3342468	1994-
06-12	499794.58	303.00	
Customer#000096103	96103	5984582	1992-
03-16	494398.79	312.00	
Customer#000024341	24341	1474818	1992-
11-15	491348.26	302.00	
Customer#000137446	137446	5489475	1997-
05-23	487763.25	311.00	
Customer#000107590	107590	4267751	1994-
11-04	485141.38	301.00	
Customer#000050008	50008	2366755	1996-
12-09	483891.26	302.00	
Customer#000015619	15619	3767271	1996-
08-07	480083.96	318.00	
Customer#000077260	77260	1436544	1992-
09-12	479499.43	307.00	
Customer#000109379	109379	5746311	1996-
10-10	478064.11	302.00	
Customer#000054602	54602	5832321	1997-
02-09	471220.08	307.00	
Customer#000105995	105995	2096705	1994-
07-03	469692.58	307.00	
Customer#000148885	148885	2942469	1992-
05-31	469630.44	313.00	
Customer#000114586	114586	551136	1993-
05-19	469605.59	308.00	
Customer#000105260	105260	5296167	1996-
09-06	469360.57	303.00	
Customer#000147197	147197	1263015	1997-
02-02	467149.67	320.00	
Customer#000064483	64483	2745894	1996-
07-04	466991.35	304.00	
Customer#000136573	136573	2761378	1996-
05-31	461282.73	301.00	
Customer#000016384	16384	502886	1994-
04-12	458378.92	312.00	
Customer#000117919	117919	2869152	1996-
06-20	456815.92	317.00	
Customer#000012251	12251	735366	1993-
11-24	455107.26	309.00	
Customer#000120098	120098	1971680	1995-
06-14	453451.23	308.00	
Customer#000066098	66098	5007490	1992-
08-07	453436.16	304.00	
Customer#000117076	117076	4290656	1997-
02-05	449545.85	301.00	
Customer#000129379	129379	4720454	1997-
06-07	448665.79	303.00	
Customer#000126865	126865	4702759	1994-
11-07	447606.65	320.00	
Customer#000088876	88876	983201	1993-
12-30	446717.46	304.00	
Customer#000036619	36619	4806726	1995-
01-17	446704.09	328.00	
Customer#000141823	141823	2806245	1996-
12-29	446269.12	310.00	
Customer#000053029	53029	2662214	1993-
08-13	446144.49	302.00	
Customer#000018188	18188	3037414	1995-
01-25	443807.22	308.00	
Customer#000066533	66533	29158	1995-
10-21	443576.50	305.00	
Customer#000037729	37729	4134341	1995-
06-29	441082.97	309.00	
Customer#000003566	3566	2329187	1998-
01-04	439803.36	304.00	
Customer#000045538	45538	4527553	1994-
05-22	436275.31	305.00	
Customer#000081581	81581	4739650	1995-
11-04	435405.90	305.00	
Customer#000119989	119989	1544643	1997-
09-20	434568.25	320.00	
Customer#000003680	3680	3861123	1998-
07-03	433525.97	301.00	

Customer#000113131	113131	967334	1995-
12-15	432957.75	301.00	
Customer#000141098	141098	565574	1995-
09-24	430986.69	301.00	
Customer#000093392	93392	5200102	1997-
01-22	425487.51	304.00	
Customer#000015631	15631	1845057	1994-
05-12	419879.59	302.00	
Customer#000112987	112987	4439686	1996-
09-17	418161.49	305.00	
Customer#000012599	12599	4259524	1998-
02-12	415200.61	304.00	
Customer#000105410	105410	4478371	1996-
03-05	412754.51	302.00	
Customer#000149842	149842	5156581	1994-
05-30	411329.35	302.00	
Customer#000010129	10129	5849444	1994-
03-21	409129.85	309.00	
Customer#000069904	69904	1742403	1996-
10-19	408513.00	305.00	
Customer#000017746	17746	6882	1997-
04-09	408446.93	303.00	
Customer#000013072	13072	1481925	1998-
03-15	399195.47	301.00	
Customer#000082441	82441	857959	1994-
02-07	382579.74	305.00	
Customer#000088703	88703	2995076	1994-
01-30	363812.12	302.00	

(57 rows)

qualification query 19

```

-- using default substitutions
select 'Stream 0 Query 19 START - ',
date_part('epoch',LOCALTIMESTAMP(1));
select
    sum(l_extendedprice* (1 - l_discount)) as
revenue
from
    lineitem,
part
where
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#12'
        and p_container in ('SM CASE', 'SM
BOX', 'SM PACK', 'SM PKG')
        and l_quantity >= 1 and l_quantity <= 1
+ 10
        and p_size between 1 and 5
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN
PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#23'
        and p_container in ('MED BAG', 'MED
BOX', 'MED PKG', 'MED PACK')
        and l_quantity >= 10 and l_quantity <=
10 + 10
        and p_size between 1 and 10
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN
PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#34'
        and p_container in ('LG CASE', 'LG
BOX', 'LG PACK', 'LG PKG')
        and l_quantity >= 20 and l_quantity <=
20 + 10

```

```

        and p_size between 1 and 15
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN
PERSON'
    );
select 'Stream 0 Query 19 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

```

Fri Oct 26 12:21:44 PDT 2007

```

revenue
-----
3083843.026
(1 row)

```

qualification query 20

```

-- using default substitutions
select 'Stream 0 Query 20 START - ',
date_part('epoch',LOCALTIMESTAMP(1));
select
    s_name,
    s_address
from
    supplier,
    nation
where
    s_suppkey in (
        select
            ps_suppkey
        from
            partsupp
        where
            ps_partkey in (
                select
                    p_partkey
                from
                    part
                where
                    p_name like
'forest%'
            )
        and ps_availqty > (
            select
                0.5 *
sum(l_quantity)
            from
                lineitem
            where
                l_partkey =
ps_partkey
                and l_suppkey =
ps_suppkey
                and l_shipdate >=
date '1994-01-01'
                and l_shipdate <
cast(date '1994-01-01' + interval '1 year' as date)
            )
        and s_nationkey = n_nationkey
        and n_name = 'CANADA'
order by
    s_name;
select 'Stream 0 Query 20 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

```

Fri Oct 26 12:21:45 PDT 2007

```

s_name | s_address
-----+-----
Supplier#000000020 | iybAE,RmTymrZVYaFZva2SH,j
Supplier#000000091 |
YV45D7TkfdQan00Z7q9QxkyGUapU1oOWU6q3

```

```

Supplier#000000197 |
YC2Acon6kjY3zj3Fbxs2k4Vdf7X0cd2F
Supplier#000000226 | 83qOdu2EYRdPQAQhEtn GRZED
Supplier#000000285 |
Br7elnntlyxrw6ImgpJ7YdhFDjuBf
Supplier#000000378 | FfbhyCxWvcPr08ltp9
Supplier#000000402 |
i9Sw4DoyMhzhKXCH9By,AYSgmD
Supplier#000000530 | 0qwCMwobKY
OcmLyfRXlagA8ukENJv,
Supplier#000000688 | D
fw5ocppmZpYBBIPI718hCihLDZ5KhKX

```

lines deleted

```

Supplier#000009278 | RqYTzgxj93CLX
OmcYfCENoefD
Supplier#000009327 | uoqMdf7e7Gj9dbQ53
Supplier#000009430 | igRqmneFt
Supplier#000009567 |
r4Wfx4c3xsEAjcgJ71HHZByornl D9vrztXlv4
Supplier#000009601 | 51m637bO,Rw5DnHWFUVvLacRx9
Supplier#000009709 |
rRnCbHYgDg19PZYnyWKVYSUW0vKg
Supplier#000009753 | wLhVEcRmd7PkJF4FBnGK7Z
Supplier#000009796 | z,y4Idmr15DOvPUqYG
Supplier#000009799 | 4wNjxGa4OKWl
Supplier#000009811 | E3iuyq7UnZxU7oPZIE2Gu6
Supplier#000009812 |
APFRMy3lCbGfGa53n5t9DxzFPQPgnjrGt32
Supplier#000009862 | rJzweWeN58
Supplier#000009868 | ROjGgx5gvtkmmUUoeyy7v
Supplier#000009869 |
ucLqxzrpBTRMewGSM29t0rNTM30g1Tu3Xgg3mKag
Supplier#000009899 | 7XdpAhrzrlt,UQFZE
Supplier#000009974 |
7wJ,J5DKcxSU4KplcQLpbcAvB5AsvKT
(204 rows)

```

qualification query 21

```

-- using default substitutions
select 'Stream 0 Query 21 START - ',
date_part('epoch',LOCALTIMESTAMP(1));
select
    s_name,
    count(*) as numwait
from
    supplier,
    lineitem l1,
    orders,
    nation
where
    s_suppkey = l1.l_suppkey
    and o_orderkey = l1.l_orderkey
    and o_orderstatus = 'F'
    and l1.l_receiptdate > l1.l_commitdate
    and exists (
        select
            *
        from
            lineitem l2
        where
            l2.l_orderkey = l1.l_orderkey
            and l2.l_suppkey <> l1.l_suppkey
    )
    and not exists (
        select
            *

```

```

from
  lineitem l3
where
  l3.l_orderkey = l1.l_orderkey
  and l3.l_suppkey <> l1.l_suppkey
  and l3.l_receiptdate >
l3.l_commitdate
)
  and s_nationkey = n_nationkey
  and n_name = 'SAUDI ARABIA'
group by
  s_name
order by
  numwait desc,
  s_name
limit 100;
select 'Stream 0 Query 21 STOP - ',
date_part('epoch',LOCALTIMESTAMP(1));

```

Fri Oct 26 12:21:51 PDT 2007

s_name	numwait
Supplier#000002829	20
Supplier#000005808	18
Supplier#000000262	17
Supplier#000000496	17
Supplier#0000002160	17
Supplier#0000002301	17
Supplier#0000002540	17
Supplier#0000003063	17
Supplier#00000005178	17
Supplier#000008331	17
Supplier#000002005	16
Supplier#0000002095	16
Supplier#0000005799	16
Supplier#0000005842	16
Supplier#000006450	16
Supplier#0000006939	16
Supplier#000009200	16
Supplier#000009727	16
Supplier#0000000486	15
Supplier#000000565	15
Supplier#000001046	15
Supplier#000001047	15
Supplier#000001161	15
Supplier#000001336	15
Supplier#000001435	15
Supplier#0000003075	15
Supplier#000003335	15
Supplier#0000005649	15
Supplier#000006027	15
Supplier#000006795	15
Supplier#000006800	15
Supplier#000006824	15
Supplier#0000007131	15
Supplier#000007382	15
Supplier#000008913	15
Supplier#000009787	15
Supplier#000000633	14
Supplier#000001960	14
Supplier#000002323	14
Supplier#000002490	14
Supplier#000002993	14
Supplier#000003101	14
Supplier#000004489	14
Supplier#000005435	14
Supplier#000005583	14
Supplier#000005774	14
Supplier#0000007579	14
Supplier#000008180	14
Supplier#000008695	14
Supplier#000009224	14
Supplier#000000357	13
Supplier#000000436	13
Supplier#000000610	13
Supplier#000000788	13
Supplier#000000889	13

Supplier#000001062	13
Supplier#000001498	13
Supplier#000002056	13
Supplier#000002312	13
Supplier#000002344	13
Supplier#000002596	13
Supplier#000002615	13
Supplier#000002978	13
Supplier#000003048	13
Supplier#000003234	13
Supplier#000003727	13
Supplier#000003806	13
Supplier#000004472	13
Supplier#000005236	13
Supplier#000005906	13
Supplier#000006241	13
Supplier#000006326	13
Supplier#000006384	13
Supplier#000006394	13
Supplier#000006624	13
Supplier#000006629	13
Supplier#000006682	13
Supplier#000006737	13
Supplier#000006825	13
Supplier#000007021	13
Supplier#000007417	13
Supplier#000007497	13
Supplier#000007602	13
Supplier#000008134	13
Supplier#000008234	13
Supplier#000009435	13
Supplier#000009436	13
Supplier#000009564	13
Supplier#000009896	13
Supplier#000000379	12
Supplier#000000673	12
Supplier#000000762	12
Supplier#000000811	12
Supplier#000000821	12
Supplier#000001337	12
Supplier#000001916	12
Supplier#000001925	12
Supplier#000002039	12
Supplier#000002357	12
Supplier#000002483	12

qualification query 22

```

-- using default substitutions
select 'Stream 0 Query 22 START - ',
date_part('epoch',LOCALTIMESTAMP(1));
select
  centrycode,
  count(*) as numcust,
  sum(c_acctbal) as totacctbal
from
  (
    select
      substring(c_phone from 1 for 2)
    as centrycode,
      c_acctbal
    from
      customer
    where
      substring(c_phone from 1 for 2)
in
  ('13', '31', '23', '29',
'30', '18', '17')
  and c_acctbal > (
    select
      avg(c_acctbal)
    from
      customer

```

```

        where
            c_acctbal > 0.00
        and
        substring(c_phone from 1 for 2) in
            ('13',
            '31', '23', '29', '30', '18', '17')
        )
        and not exists (
            select
                *
            from
                orders
            where
                o_custkey =
                    c_custkey
        )
    ) as custsale
group by
    centrycode
order by
    centrycode;
select 'Stream 0 Query 22 STOP - ',
date_part('epoch', LOCALTIMESTAMP(1));

```

```

Fri Oct 26 12:21:57 PDT 2007
  centrycode | numcust | totacctbal
-----|-----|-----
    13      |    888  | 6737713.99
    17      |    861  | 6460573.72
    18      |    964  | 7236687.40
    23      |    892  | 6701457.95
    29      |    948  | 7158866.63
    30      |    909  | 6808436.13
    31      |    922  | 6806670.18
(7 rows)

```

Appendix D. Seed and Query Substitution Parameters

This Appendix contains Seed values and substitution parameters for each stream

Seed Values

```

stream0 1021214421
stream1 1021214422
stream2 1021214423
stream3 1021214424
stream4 1021214425
stream5 1021214426
stream6 1021214427

```

Query Parameters

stream0: 1021214421

```

=====
14 1996-01-01
2 31 BRASS MIDDLE EAST
9 medium
20 peach 1997-01-01 GERMANY

```

```

6 1995-01-01 0.08 25
17 Brand#44 WRAP JAR
18 315
8 ROMANIA EUROPE LARGE BURNISHED NICKEL
21 SAUDI ARABIA
13 express accounts
3 MACHINERY 1995-03-28
22 17 16 15 22 23 21 18
16 Brand#44 ECONOMY POLISHED 45 35
41 4 33 9 26 11
4 1997-05-01
11 MOZAMBIQUE 0.0000003333
15 1994-01-01
1 69
10 1993-02-01
19 Brand#54 Brand#52 Brand#21 6
17 25
5 EUROPE 1995-01-01
7 ROMANIA JORDAN
12 FOB MAIL 1994-01-01

```

stream1: 1021214422

```

=====
21 JAPAN
3 BUILDING 1995-03-13
18 312
5 MIDDLE EAST 1995-01-01
11 EGYPT 0.0000003333
7 IRAQ INDONESIA
6 1995-01-01 0.05 24
20 blush 1995-01-01 RUSSIA
17 Brand#41 WRAP CAN
12 MAIL SHIP 1996-01-01
16 Brand#24 SMALL ANODIZED 6 35 25
46 12 13 5 4
15 1996-01-01
13 express accounts
10 1993-11-01
2 19 TIN ASIA
8 IRAQ MIDDLE EAST MEDIUM BRUSHED NICKEL
14 1996-01-01
19 Brand#51 Brand#35 Brand#11 1
18 21
9 lemon
22 13 14 15 12 28 17 11
1 77
4 1995-02-01

```

stream2: 1021214423

```

=====
6 1996-01-01 0.03 24
17 Brand#43 SM CASE
14 1997-01-01
16 Brand#14 LARGE BURNISHED 27 26
13 18 34 43 40 5
19 Brand#13 Brand#13 Brand#15 6
19 28
10 1994-08-01
9 indian
2 7 COPPER MIDDLE EAST
15 1994-01-01
8 CANADA AMERICA MEDIUM PLATED NICKEL
5 AFRICA 1996-01-01
22 33 13 31 24 25 27 11
12 TRUCK SHIP 1996-01-01
7 CANADA IRAN
13 express deposits
18 314
1 85
4 1997-08-01
20 magenta 1993-01-01 JAPAN
3 MACHINERY 1995-03-30
11 PERU 0.0000003333
21 EGYPT

```

=====
stream3: 1021214424
=====

```
8 SAUDI ARABIA MIDDLE EAST MEDIUM ANODIZED
NICKEL
5 AMERICA 1996-01-01
4 1995-05-01
6 1996-01-01 0.08 25
17 Brand#45 SM JAR
7 SAUDI ARABIA INDIA
1 93
18 315
22 21 32 26 27 25 20 15
14 1997-01-01
9 ghost
10 1993-05-01
15 1997-01-01
11 ETHIOPIA 0.0000003333
20 thistle 1997-01-01 BRAZIL
2 44 STEEL ASIA
21 VIETNAM
19 Brand#15 Brand#51 Brand#14 1
20 24
13 express deposits
16 Brand#44 PROMO POLISHED 11 15 14
13 23 12 47 48
12 RAIL SHIP 1997-01-01
3 BUILDING 1995-03-15
```

=====
stream4: 1021214425
=====

```
5 ASIA 1996-01-01
21 JORDAN
14 1997-01-01
19 Brand#12 Brand#34 Brand#53 7
10 20
15 1994-01-01
17 Brand#52 SM CAN
12 REG AIR SHIP 1997-01-01
6 1996-01-01 0.05 24
4 1993-02-01
9 drab
8 JAPAN ASIA SMALL POLISHED NICKEL
16 Brand#25 SMALL BRUSHED 14 1 28
5 9 30 31 18
11 CHINA 0.0000003333
2 32 BRASS AFRICA
10 1994-03-01
18 313
1 101
13 express deposits
7 JAPAN FRANCE
22 20 18 15 25 33 23 14
3 HOUSEHOLD 1995-03-01
20 goldenrod 1995-01-01 MOZAMBIQUE
```

=====
stream5: 1021214426
=====

```
21 FRANCE
15 1997-01-01
4 1995-09-01
6 1996-01-01 0.03 24
7 EGYPT ETHIOPIA
16 Brand#15 ECONOMY BURNISHED 12 11
13 14 46 20 35 8
19 Brand#24 Brand#22 Brand#52 2
11 28
18 314
14 1997-01-01
22 16 32 18 21 13 27 22
```

```
11 FRANCE 0.0000003333
13 special deposits
3 BUILDING 1995-03-17
1 109
2 20 NICKEL ASIA
5 EUROPE 1996-01-01
8 EGYPT MIDDLE EAST SMALL BURNISHED BRASS
20 rose 1994-01-01 FRANCE
12 SHIP REG AIR 1997-01-01
17 Brand#54 LG CASE
10 1994-12-01
9 cream
```

=====
stream6: 1021214427
=====

```
10 1993-09-01
3 HOUSEHOLD 1995-03-03
15 1995-01-01
13 special deposits
6 1996-01-01 0.08 25
8 VIETNAM ASIA STANDARD BRUSHED BRASS
9 chartreuse
7 VIETNAM CANADA
4 1993-06-01
11 ROMANIA 0.0000003333
22 28 30 13 20 21 19 11
18 312
12 FOB REG AIR 1997-01-01
1 117
5 AFRICA 1996-01-01
16 Brand#45 STANDARD PLATED 22 10
9 46 24 12 8 17
2 8 COPPER AFRICA
14 1998-01-01
19 Brand#21 Brand#55 Brand#52 7
12 24
20 cornflower 1997-01-01 VIETNAM
17 Brand#51 LG JAR
21 UNITED KINGDOM
```

Appendix E. Implementation-Specific Layer/Driver Code

=====
ntest
=====

```
#!/bin/bash

source /home/paracel/.bashrc

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

test_timestamp=`date +%m_%d_%H_%M_%S`

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 "
    exit
fi

export
```



```

audit_file_dir="$HOME/run/paraccel_stage/results/"
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 thurefresh
run without any refreshes "
  echo " - thurefresh do all the throughput
refreshes without query streams"
  echo " - all do full audit run: load
+ 2 power-thurefresh cycles:"
  echo " run1 <onerefresh 1 + stream0 +
onerefresh 2 + thrustreams + thurefresh> + 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: $0 qi
scale_factor "
  echo " note: database must
be running; if not restart it first"
  echo
  exit
  fi
  query_num=${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 "
  num_query_streams input_seed "
  echo
  exit
  fi
  sf=$2
  nqs=$3
  let input_seed=$4 ;;
thurefresh) if (( $# < 4 || $# > 4 ))
then
  echo
  echo "usage: $0 thurefresh
scale_factor "
  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 "
  num_query_streams "
  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
if [[ ($scope = all ) && -z $real_scope && -z
$real_scope2 ]]
then
  case $sf in
    .1) aminqs=1
        ;;
    1) aminqs=2
        ;;
    10) aminqs=3
        ;;
    30) aminqs=4
        ;;
    100) aminqs=5
        ;;
    300) aminqs=6
        ;;
    1000) aminqs=7
        ;;
    3000) aminqs=8
        ;;
    10000) aminqs=9
        ;;
    * )
        echo
        echo "ERROR: scale factor (=$sf) must
be one of (.1,1,10,30,100,300,1000,3000,10000)"
        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
    ;;
*)
  ;;
esac

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

if [[ $scope = thrurefresh || $scope = all
]]
then
  # generate update_throughputX.sql
  i=2
  run=1
  ((max_streams=nqs + 1))
  while((run<=2))
  do
    echo "MAX = $max_streams"
    upd_tput_fname="update_throughput_run_${ru
n}_${nqs}.sql"
    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 "exec rf1 ${i};" >>
$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 "exec rf2 ${i};" >>
$upd_tput_fname
      echo "select 'Stream ${stream_number}
RF2 STOP - ', date_part('epoch',LOCALTIMESTAMP(3)),
localtimestamp;" >> $upd_tput_fname
      ((i=i+1))
    done
  done

```

```

        ((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 = threfresh || $scope = run1 || $scope
= all || $scope = mall ]]
then
    run=1
    while((run<=2))
    do
        if [[ -f
update_throughput_run_${run}_${nqs}.sql ]]
        then
            echo
        else
            echo
            echo "ERROR:
update_throughput_run_${run}_${nqs}.sql 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 compliant minimum

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

if ((sf >= 1000))
then
    big=big # make l_orderkey and o_orderkey
    bigint
else
    big= # otherwise int
fi

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

echo " "
case $scope in
    q* )
        ;;
    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:
" ;;
        thstreams )
            echo "ONLY DOING ONE THROUGHPUT TEST
WITHOUT REFRESHES with:" ;;
        threfresh )
            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,threfresh,run1)"
            echo
            exit ;;
esac

echo " "

        echo "        scale factor = $sf "

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

if [[ $scope = load || $scope = all ]]
then
    if [[ $big = big ]]
    then
        echo "        using orderkeys of type unsigned
bigint"
    else
        echo "        using orderkeys of type unsigned
int "
    fi
    echo
fi

if [[ $scope != load && $scope != refresh1 && $scope !=
refresh2 ]]
then
    if [[ $input_seed == 1 ]]
    then
        seed=$input_seed
        echo "        using newly generated 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
    fi

```

```

else
    echo "      using $input_seed as seed"
    seed=$input_seed
fi
fi

echo "      Using the following nodes"
echo "      -----"
cqi show | grep -i node #replace this with cqi -s
when sysmgr is ready

echo -e "Are these OK? (type y or n) \c"
read ans

if [[ $ans = n || $ans = no ]]
then
    echo
    echo change one or more of
    echo " options.sql, tpch.cfg,
create_database.sql, create_tables_${big}int or
/etc/system"
    echo and try again
    echo
    exit
fi

# ----- start real execution
-----
if [[ $scope = all ]]
then
    rm -f ${audit_file_dir}* 2>/dev/null
fi

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
    do_one_query $query_num $sf
    echo
    exit
fi

echo
echo

if [[ $scope = load ]]
then

    echo " "
    echo " "
    echo " "

    if [[ -e cud.ooo ]]
    then
        rm cud.ooo
    fi

    echo "      Creating tpch database: `date`"
    echo " "
    createdb.sh > cud.ooo
    echo " "
    #This installs the stored procedures for refresh
1 and 2
    echo "Installing stored procs for RF1 and RF2."
    echo " "
    psql -d tpch -f create_rf1.sql -o create_rf1.out
    psql -d tpch -f create_rf2.sql -o create_rf2.out
    if [[ -s cud.ooo ]] # cud.ooo exists and non-
empty
    then
        echo "      tpch database created: `date`"
    else
        echo

```

```

echo ERROR: create_database.sh failed
echo

```

```

exit
fi
echo " "
echo " "

```

```

fi

```

```

# create db_spaces and create the 2 stored procs
which control the refresh operations
if [[ $scope = load ]]
then

```

```

psql -d tpch -f time_wait.sql -o tpchw.ooo
if [[ -s tpchw.ooo ]]
then
    echo "      executed tpch_wait.sql"
else
    echo
    echo ERROR: tpch_wait.sql failed
    echo
    exit
fi

```

```

echo " "
fi

```

```

if [[ $scope = load ]]
then

```

```

sleep 5

if [[ -e ct.ooo ]]
then
    rm ct.ooo
fi

```

```

psql -d tpch -f create_tables_${big}int.sql -o
ct.ooo
if [[ -s ct.ooo ]]
then

```

```

    echo "      executed
create_tables_${big}int.sql"
    else
        echo
        echo ERROR: create_tables_${big}int.sql
failed
        echo
        exit
    fi

```

```

sleep 5

```

```

#
# Load the database
#
echo " "

```

```

starttime=`date "+%Y-%m-%d %H:%M:%S"`
echo "      Load started $starttime " | tee
start_load.out
psql -d tpch -a -f load_lineitem.sql >
load_lineitem.out &
loadlpid=$!
echo " "
echo "      - lineitem load started: `date` "
echo " "
psql -d tpch -a -f load_region.sql >
load_region.out
echo "      - region load completed: `date` "
psql -d tpch -a -f load_nation.sql >
load_nation.out
echo "      - nation load completed: `date` "
psql -d tpch -a -f load_customer.sql >
load_customer.out
echo "      - customer load completed: `date` "
"
psql -d tpch -a -f load_part.sql > load_part.out
echo "      - part load completed: `date` "

```

```

    psql -d tpch -a -f load_supplier.sql >
load_supplier.out
    echo "          - supplier load completed: `date`
"
    psql -d tpch -a -f load_partsupp.sql >
load_partsupp.out
    echo "          - partsupp load completed: `date`
"
    psql -d tpch -a -f load_orders.sql >
load_orders.out
    echo "          - orders load completed: `date` "
    wait $loadlpid
    echo "          - lineitem load completed: `date`
"
    echo
    echo "      Building statistics.."
    echo
    echo ANALYZE START at `date` > analyze.out
    psql -d tpch -a -c 'analyze' >> analyze.out
    echo ANALYZE STOP at `date` >> analyze.out
    echo
    echo "      Done building statistics `date` "
    echo
    endyear=`date +%Y`\
    enddate=`date +%m%d`\
    endtime=`date +%H%M%S`\
    endstamp=`date "+%Y-%m-%d %H:%M:%S"`
    echo "      Load Finished $endstamp" | tee
end_load.out
    echo "      Seed time is:
$endyear$enddate$endtime"
    echo
    show_asiq_cpu end_load > end_load_cpu.out
    slt=`tr -s ' ' <start_load.out | cut -d' ' -f 4,5`
    ,
    elt=`tr -s ' ' <end_load.out | cut -d' ' -f 4,5`
    echo "CALLING CALCULATE_LOAD.. FROM NTEST"
    lt=`calculate_load_time.bash "$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 $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 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
echo "Performing Pre-Test clean-up."
echo
if rm stream*.out 2>/tmp/junk
then
    echo "Clean-up: Removed stream*.out"
else
    echo "Clean-up No stream files to remove"
fi
echo
if rm update_power.out 2>/tmp/junk
then
    echo "Clean-up: Removed update_power.out"
else
    echo "Clean-up: No update_power.out file to remove"
fi
echo
if rm update_throughput.out 2>/tmp/junk
then
    echo "Clean-up: Removed update_throughput.out"
else
    echo "Clean-up: No update_throughput.out file to
remove"
fi
echo
echo
if [[ ( $scope = refresh1 || $scope = refresh2 ||
$scope = stream0 || $scope = all ) ]]
then
    if [[ $scope = all ]]
    then
        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

if [[ ($scope = threrefresh || $scope = all || $scope
= thurstreams) ]]
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

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

            if [[ $scope != thurstreams ]] # thurstreams
does a threrefresh without refreshes
            then
                echo "      Start Refresh Stream for set $nqs
`date` "

                psql -d tpch -f
update_throughput_run_1_${nqs}.sql >>
update_throughput.out

                echo
                echo "      Refresh stream has completed
`date` "
                fi
                throughput_interval=$(tpch_throughput_interval.ba
sh $nqs)
                tpch_throughput_interval.bash $nqs >
throughput_interval.out
                throughput=$(tpch_stats.pl "threrefresh" $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 =
thurstreams ||
                $scope = threrefresh || $scope = all ]]

```

```

then
    dayHr=`date +%m%d%H`
    echo
    echo
    if [[ -z $real_scope2 ]] # non_null means
lthrurefresh
    then
lrun1
        if [[ -z $real_scope ]] # non_null means
            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_"$test_timestamp

#rpt_file_name="mrun1_${scope}_${sf}g_${nqs}s_${dayHr}
_${xenv}_ln_${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}_ln.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_"$test_timestamp
            echo "Producing ${rpt_file_name}"
            tpch_report.new $scope $sf
\
                run1_${scope}_${sf}g_${nqs}s_
${dayHr}_${xenv}_ln.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 >
${audit_file_dir}mls${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 >
${audit_file_dir}mls${i}q${qn}.trunc.out
                    fi
                    ((q=q+1))
                done
                ((i=i+1))
                done
                mv update_power.out $audit_file_dir"mls00rf.out"
#change to move
                mv update_throughput.out
$audit_file_dir"mls01rf.out" #change to move
                mv load_*.out $audit_file_dir
                mv analyze.out $audit_file_dir

                echo "Moved *.out files to audit naming standard
in ../paracel_stage/results"
                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 "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 refresh1.sql > rfl.out
        echo
        echo " End refresh 1 `date` "
        #create the refresh file
        cat rfl.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 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

if [[ ($scope = thurerefresh || $scope = all || $scope
= thurstreams) ]]
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

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

    if [[ $scope != thurstreams ]] # thurstreams
does a thurerefresh without refreshes
then
        echo "      Start Refresh Stream for set $nqs
`date` "

        psql -d tpch -f update_throughput${nqs}.sql
>> update_throughput.out

        echo
        echo "      Refresh stream has completed
`date` "
        fi
        throughput_interval=$(tpch_throughput_interval.ba
sh $nqs)
        tpch_throughput_interval.bash $nqs >
throughput_interval.out
        throughput=$(tpch_stats.pl "thurerefresh" $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 =
thurstreams ||
$scope = thurerefresh || $scope = all ]]
then
    dayHr=`date +%m%d%H`
    echo

```



```

echo
if [[ -z $real_scope2 ]] # non_null means
lthreupdate
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 = threupdate || $scope =
all ]]
then
composite=$throughput
fi
rpt_file_name="run2_`${test_timestamp}`
echo "Producing `${rpt_file_name}`"
echo "Running tpch_report
#####
tpch_report.new $scope $sf
\
run1_${scope}_${sf}g_${nqs}
s_${dayHr}_${xenv}_ln.r \
$seed $nqs $nd $ds $sc
\
> ${rpt_file_name}
mv ${rpt_file_name} $audit_file_dir
else
echo " "
fi
else
# lthreupdate
composite=$throughput
rpt_file_name="run2_`${test_timestamp}`
echo "Producing `${rpt_file_name}`"
tpch_report.new $scope $sf
\
run1_${scope}_${sf}g_${nqs}s_
${dayHr}_${xenv}_ln.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 >
${audit_file_dir}m2s${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 >
${audit_file_dir}m2s${i}q${qn}.trunc.out
fi
((q=q+1))
done
((i=i+1))
echo create query output files here - dude
done

mv update_power.out $audit_file_dir"m2s00rf.out"
#change to move
mv update_throughput.out
$audit_file_dir"m2s01rf.out" #change to move

echo "Moved *.out files to audit naming standard
in ../paracel_stage/results"
echo
echo "FINISHED Run2 `date`"
echo
fi
exit

=====
tpch_stats.pl
=====

#!/usr/bin/perl -w
use strict;

my @time_array;
my @scratch;
my $query;
my $start_time;
my $end_time;
my $file;
my $throughput;
my $throughput_interval;

# Validate input arguments
if (!$ARGV[3])
{
print "\n";
print "Usage: tpch_throughput.pl [Scope (power or
throughput)] [Scale Factor] [Number of Streams]
[Return Value]\n";
print "\n";
exit;
}

# Specifying "none" for a return value will allow
verbose printouts (Not returning a value to a caller)

# Set variables based on input arguments

my $scope = $ARGV[0];
my $scale_factor = $ARGV[1];
my $num_streams = $ARGV[2];
my $return_value = $ARGV[3];

# Power tests only run one stream. Don't rely on
command line input for this.
if ($scope eq "power")
{
$num_streams = 1;
}

# This method calculates the geometric mean
sub GeometricMean()

```

```

{
  my $sum = $_[0];
  # Create a string that returns the geometric sum
[Note: input is already Log(q1) + Log(q2)..]
  my $line = "10**($sum/($num_streams*22)+2)";
  #my $line = "$sum/($num_streams*22)+2)";
  # Calculate the geometric mean
  my $geometric_mean = `my_calc.pl "$line"`;
  # Return the geometric mean, rounded to tenths of a
second
  $geometric_mean = sprintf("%.1f",$geometric_mean);
  return $geometric_mean
}

# This method calculates the arithmetic mean
sub ArithmeticMean()
{
  my $sum = $_[0];
  # Calculate the arithmetic mean
  my $line = "$sum/($num_streams*22)+2)";
  my $arithmetic_mean = `my_calc.pl "$line"`;
  # Return the arithmetic mean, rounded to tenths of
a second
  $arithmetic_mean =
sprintf("%.1f",$arithmetic_mean);
  return $arithmetic_mean;
}

# This method calculates TPC-H Power
sub TPCHPower()
{
  my $sum = $_[0];
  # Calculate the arithmetic mean
  my $line = "3600*$scale_factor**(-($sum)/24)";
  # my $line = "3600*$scale_factor**(-
($sum)/($num_streams * 24))";
  my $power = `my_calc.pl "$line"`;
  # Return the tpc-h power, rounded to tenths of a
second
  $power = sprintf("%.1f",$power);
  return $power;
}

# This method calculates throughput
sub TPCHThroughput()
{
  my $throughput_interval = $_[0];
  # create a string whose value is the throughput
metric
  my $line =
"$num_streams*22*3600*$scale_factor/$throughput_interv
al";
  #evaluate the throughput metric
  my $th=`my_calc.pl "$line"`;
  # $throughput=`round_to_tenths.pl $th`;
  $throughput = sprintf("%.1f",$th);
  return $throughput;
}

# This is a print delegate that allows for
conditionals
# If this script is returning a value, printing is
silenced
sub MyPrint
{
  my $message = $_[0];
  if ($return_value eq "none")
  {
    print $message;
  }
}

#This method sorts a two dimensional array and prints
the results
sub SortPrintArray()
{
  #my $list1_title = $_[0];
  #my $list2_title = $_[2];
  #my $list3_title = $_[4];

  my @biglist = $_[0];
  print $biglist[0] [5];
  my @a;
  my @b;
  my @sorted = map { $_[0] }
                sort { $a->[0] cmp $b->[0] }
                map { [ $_[0], $_[1] ] } @biglist;
  print "--- Sorted -----\\n";
  print $sorted[0] [0];
  print "\\n";
  print "-----\\n";
}

# Create log string for 22 queries, n streams
sub ParseFile
{
  #Input args:
  my $file = $_[0];
  my $stream_num = $_[1];
  my $success = 0; #later this bool answers: Did the
file open?

  my $offset = 1;
  my $raw_sum = 0;
  my $log_sum = 0;
  my $line = "";
  my $rows = 0;
  my $exectime;
  my $rounded_exectime;

  # array used to sort output
  my @queries;
  my @exec_times;
  my @perf_results;

  #Adjust parse offset, if necessary
  open(FILE,$file) || exit;
  while(<FILE>)
  {
    chomp($_);
    my @scratch = split(/\\s+/, $_);
    if ( $_ =~ /START/)
    {
      #Read the stream number from the file if this
is a throughput update
      if ($file =~ /update_throughput/)
      {
        $stream_num = $scratch[2];
        $offset = 0;
      }
      $query = $scratch[3 + $offset];
      $start_time = $scratch[7 + $offset];
    }
    if ( $_ =~ /STOP/)
    {
      $end_time = $scratch[7 + $offset];
      $exectime = $end_time - $start_time;
      $raw_sum = $raw_sum + $exectime;
      if ($rows == 0)
      {
        $line = "Log($exectime)";
      }
      else
      {
        $line = "$line+Log($exectime)";
      }
      # round to first decimal place
      $rounded_exectime =
sprintf("%.1f",$exectime);
      if ($scope eq "throughput" || $scope eq
"tpworf"
          || $scope eq "trefresh")
      {
        MyPrint(" Stream $stream_num: Query $query
$exectime
seconds\\n");
      }
      if ($scope eq "power" || $scope eq "stream0"
          || $scope eq "prefresh" || $scope eq "refresh")
      {
        MyPrint(" Query $query $exectime

```

```

seconds \n");
    }
    push (@queries, $query);
    push (@exec_times, $exectime);
    $rows++;
  }
}
my $element;
my @output = (@queries,@exec_times);
#&SortPrintArray(@output);
my $line_length = length($line);
my @return;
if ($line_length > 0)
{
  # Calculate query sum
  $log_sum = $log_sum + `my_calc.pl "$line"`;
  @return = ($raw_sum, $log_sum);
}
else
{
  @return = (-1,-1);
}
return @return;
}

sub Refreshes()
{
  my $file_name = $_[0];
  my $offset = $_[1];
  # Run refreshes
  my @result = ParseFile($file_name, $offset);
  my $raw_sum = $result[0];
  my $log_sum = $result[1];
  return @result;
}

sub Power()
{
  my $raw_sum = 0;
  my $log_sum = 0;
  my @result;
  # Set the number of streams to one (power test)
  # $num_streams = 1;
  MyPrint("TPCH Power Test Response Times\n");
  MyPrint("-----\n");
  # Run power test
  my $file = "stream0.out";
  @result = ParseFile($file, 0);
  if ($result[0] == -1)
  {
    print "File: $file failed to open\n";
    exit;
  }
  $raw_sum = $raw_sum + $result[0];
  $log_sum = $log_sum + $result[1];
  MyPrint("-----\n");
  MyPrint("Power -- Total Time Elapsed:
$raw_sum\n");
  MyPrint("-----\n");
  if ($scope eq "power")
  {
    MyPrint("Refresh Streams (Scope:
$scope)\n");
    MyPrint("-----\n");
    @result = &Refreshes("update_power.out", 1);
    $raw_sum = $raw_sum + $result[0];
    $log_sum = $log_sum + $result[1];
    my $refresh_sum = $result[0];
    MyPrint ("Total (RF1 and RF2):
$refresh_sum seconds\n");
    MyPrint("-----\n");
    MyPrint("Power + Refreshes:      Total Time
Elapsed:      $raw_sum\n");
  }
}

```

```

MyPrint("-----\n");
}
# Print the geometric mean, rounded to tenths of
a second
my $geometric_mean = &GeometricMean($log_sum);
MyPrint("Power Test Geometric Mean:
$geometric_mean\n");
#Print the arithmetic mean, rounded to tenths of
a second
my $arithmetic_mean = &ArithmeticMean($raw_sum);
MyPrint("Power Test Arithmetic Mean:
$arithmetic_mean\n");
MyPrint("-----\n");
my $power = &TPCHPower($log_sum);
my @return = ($power, $arithmetic_mean,
$geometric_mean);
return @return;
}

sub Throughput()
{
  my $raw_sum = 0;
  my $log_sum = 0;
  my $stream_num;
  my @result;
  MyPrint("\n");
  MyPrint("TPCH Throughput Test Response Times\n");
  MyPrint("-----\n");
  # Run throughput test
  for (my $count = 0; $count<$num_streams;
$count++)
  {
    $stream_num = $count + 1;
    $file = "stream".($stream_num)."out";
    @result = ParseFile($file,$stream_num,1);
    $raw_sum = $raw_sum + $result[0];
    $log_sum = $log_sum + $result[1];
    if ($result[0] == -1)
    {
      print "File: $file failed to open\n";
      exit;
    }
    MyPrint("-----\n");
    MyPrint("Throughput -- Time Elapsed: Stream
$stream_num: $result[0]\n");
    MyPrint("-----\n");
  }
  my $ti_file = "throughput_interval.out";
  #The ntest script generated an output file
  containing the throughput interval.
  #Open and read that file.
  open(INTERVAL,"throughput_interval.out") || die
  "Can't open file ".$ti_file."\n";
  $throughput_interval = <INTERVAL>;
  MyPrint("-----\n");
  MyPrint("Throughput -- Total Time Elapsed:
$num_streams Streams: $throughput_interval");
  MyPrint("-----\n");
  $throughput =
&TPCHThroughput($throughput_interval);
  if ($scope eq "thrurefresh")
  {
    MyPrint("Refresh Streams (Scope:
$scope)\n");
    MyPrint("-----\n");
    # Run refreshes
    @result =
&Refreshes("update_throughput.out", 0);
    $raw_sum = $raw_sum + $result[0];
  }
}

```

```

        $log_sum = $log_sum + $result[1];
        my $refresh_sum = $result[0];
        MyPrint (" Refresh Total:
$refresh_sum      seconds\n");
        MyPrint("-----
-----\n");
        chomp($refresh_sum);
        my $test_total = $throughput_interval +
$refresh_sum;
        MyPrint (" Total Elapsed Time (Thr. + Ref.):
$test_total      seconds\n");
        MyPrint("-----
-----\n");
    }
    # Print the geometric mean, rounded to tenths of
a second
    my $geometric_mean = &GeometricMean($log_sum);
    MyPrint("Throughput Test Geometric Mean:
$geometric_mean\n");
    #Print the arithmetic mean, rounded to tenths of
a second
    my $arithmetic_mean = &ArithmeticMean($raw_sum);
    MyPrint("Throughput Test Arithmetic Mean:
$arithmetic_mean\n");
    my $power = &TPCHPower($log_sum);
    my @return = ($power, $arithmetic_mean,
$geometric_mean);
    return @return;
}

# Control report output based on scope
my @return;
if ($scope eq "power" || $scope eq "stream0")
{
    @return = &Power;
}
elseif ($scope eq "thrurefresh" || $scope eq
"thrustreams")
{
    @return = &Throughput;
}
elseif ($scope eq "refresh1" || $scope eq "refresh2"
)
{
    my $refresh_sum = 0;
    MyPrint("Refresh Streams (Scope: $scope)\n");
    MyPrint("-----
-----\n");
    for (my $i = 1;$i < 3;$i++)
    {
        @return = &Refreshes("rf$i.out");
        $refresh_sum = $refresh_sum + $return[0];
#add the raw sums of the two refreshes together
    }
    print "Total (RF1 and RF2):      $refresh_sum
seconds\n";
    MyPrint("-----
-----\n");
}
elseif ($scope eq "thrurefresh")
{
    @return = &Refreshes("update_throughput.out");
}
elseif ($scope eq "refresh1" || $scope eq "refresh2"
)
{
    @return = &Refreshes("update_power.out");
}
elseif ($scope eq "checkfiles")
{
    if ($num_streams == 1)
    {
        if(-e "update_power.out")
        {
            $scope = "power";
        }
        else
        {
            $scope = "stream0";
        }
    }
    @return = &return = &power;
}
my $stream_count = 0;
for (my $s_counter=0; $s_counter<$num_streams+1;
$s_counter++)
{
    if (-e "stream$s_counter.out")
    {
        $stream_count++;
    }
}
if ($stream_count > 0)
{
    if ($stream_count < $num_streams)
    {
        MyPrint("Stream Count: {$stream_count} is
less than the specified number of streams:
{$num_streams}\n");
        MyPrint("Will attempt to generate stats for
the existing streams\n");
    }
    if(-e "update_throughput.out")
    {
        $scope = "throughput";
        &Throughput;
    }
    else
    {
        $scope = "tworf";
        &Throughput;
    }
}
}
else { print("Invalid Scope name: $scope \n");
exit; }
#If the caller requested a return value, deliver it
and exit
use Switch;
switch ($return_value)
{
    case["power"]{print"$return[0]";}
    case["arith_mean"]{print"$return[1]";}
    case["geo_mean"]{print"$return[2]";}
    case["throughput"]{print"$throughput";}
    case["throughput_interval"]{print"$throughput_in
terval";}
    else {}
    exit;
}

=====
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

=====
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 ParAccel 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 "$_ [debug secs_from_epoch.pl]\n";
s/-/:/g;
s/ /:/;
# print "$_ [debug secs_from_epoch.pl]\n";

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

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

=====
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

arithmetic 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 coefficient function: c(n,i)

notes:
1. We don't completely parse expressions, so that
some expressions cannot be
correctly 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
}

```

```

=====
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];
printf "%.1f\n", $_;

=====
gen_streams
=====

#!/bin/bash
echo " Gen_Streams got called"

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

DSS_QUERY=/home/paracel/run/dbgen
SCRIPTS_DIR=/home/paracel/run/scripts

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

i=0

cd $DSS_QUERY

while ((i<=ns))
do
qgen -a -p $i -r $seed1$seed2 -i
$DSS_QUERY/begin.sql -t $DSS_QUERY/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

```

Appendix F. Misc database scripts

The dbtables-syb.sql script was run to validate the correctness of the database after the database load. Three other scripts were used to extract basic information about tables and indexes from the database dew_cat1.sql, dew_cat2.sql, dew_cat3.sql.

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.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 int4 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,
  PRIMARY KEY (p_partkey)
);

```

```

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,
  PRIMARY KEY (r_regionkey)
);

```

```

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,
  PRIMARY KEY (n_nationkey),
  FOREIGN KEY (n_regionkey) REFERENCES region
(r_regionkey)
);

```

```

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,
  PRIMARY KEY (s_suppkey),
  FOREIGN KEY (s_nationkey) REFERENCES nation
(n_nationkey)
);

```

```

create table temp_partsupp (
  ps_partkey int4 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,
  PRIMARY KEY (ps_partkey, ps_suppkey),
  FOREIGN KEY (ps_partkey) REFERENCES part
(p_partkey),
  FOREIGN KEY (ps_suppkey) REFERENCES supplier
(s_suppkey)
);

```

```

create table temp_customer (
  c_custkey int4 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,
  PRIMARY KEY (c_custkey),
  FOREIGN KEY (c_nationkey) REFERENCES nation
(n_nationkey)
);

```

```

create table temp_orders (
  o_orderkey int4 not null encode delta,
  o_custkey int4 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,
  PRIMARY KEY (o_orderkey),
  FOREIGN KEY (o_custkey) REFERENCES customer
(c_custkey)
);

```

```

create table temp_lineitem (
  l_orderkey int4 not null encode delta,
  l_partkey int4 not null,
  l_suppkey int4 not null,
  l_linenum 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,
FOREIGN KEY (l_orderkey) REFERENCES orders
(o_orderkey),
FOREIGN KEY (l_partkey,l_suppkey) REFERENCES
partsupp (ps_partkey,ps_suppkey)
);

```

```

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 = -2147483646;

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 = -2147483646)
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 = -2147483646;

COMMIT;

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

```

```

UPDATE TEMP_NATION                                WHERE P_PARTKEY = 2147483647;
SET N_NATIONKEY = 2147483647;

INSERT INTO NATION
SELECT * FROM TEMP_NATION;

SELECT * FROM NATION
WHERE N_NATIONKEY = 2147483647
OR N_NATIONKEY = 1;

DELETE FROM NATION
WHERE N_NATIONKEY = 2147483647;

COMMIT;

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

UPDATE TEMP_REGION
SET R_REGIONKEY = 2147483647;

INSERT INTO REGION
SELECT * FROM TEMP_REGION;

SELECT * FROM REGION
WHERE R_REGIONKEY = 2147483647
OR R_REGIONKEY = 1;

DELETE FROM REGION
WHERE R_REGIONKEY = 2147483647;

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',
'P_TYPE VARCHAR.....2.....5E', -2147483646,
'PCONTAINRE', 1234567890.12,
'PART COMMENT FIELD 23E');

SELECT * FROM PART

DELETE FROM PART
WHERE P_PARTKEY = 2147483647;

--
=====
INSERT INTO PARTSUPP
(P_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 FIELDS 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,
getdate(),
'ORDER PRIORTY5E', 'FIXED TEXT 15E',
-2147483646,
'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,
-2147483646,
-1234567890.12,
-1234567890.12,
-1234567890.12,
-1234567890.12,
'Q',
'R',
getdate(),
getdate(),
getdate(),
'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 = -2147483646;

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

--
=====
INSERT INTO NATION
(N_NATIONKEY, N_NAME, N_REGIONKEY, N_COMMENT)
VALUES
(2147483647,
'ZE REPUBLIC D MAKEBELIEVE',
2147483647,
'A NATION COMMENT FOR FIELD SIZE 152 NO E');

SELECT * FROM NATION
WHERE N_NATIONKEY = 2147483647
AND N_REGIONKEY = 2147483647;

DELETE FROM NATION
WHERE N_NATIONKEY = 2147483647
AND N_REGIONKEY = 2147483647;

--
=====
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 = 2147483647;

DELETE FROM REGION
WHERE R_REGIONKEY = 2147483647;

-- 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 and Continental Resources, Inc 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	Extended Price	Extended Support Fees 3 YEARS	
1	PAR-ADB	ParAccel Analytic Database Per GB	1,000	300	5.00%	\$285,000	
2	PAR-SUPP-B	3 yr 24/7 ParAccel Support Per GB	30	300	5.00%		\$8,550.00
3							
4		5% Discount					
5							
6							
7							
8							
9							
10							
11							
12							

Quote Date: 10/29/07
Valid thru:

Total \$293,550
 License + 3 year support

Payment terms : Net 30 Days



Sales Contact

Adam Apps
 (732) 563-0900 Main
 (732) 748-3627 Direct
 (781) 687-6423 Fax
 aapps@conres.com E-mail

QUOTATION Paracel

Quote #: 20067019
 Page #: 1
 Customer #: 32719
 Quote Date: 10/24/2007
 Cust P.O. #: n/a

Quote valid for 30 Days

Payment Terms: Net 30
F.O.B.: Hudson, N.H.

LINE #	MANUFACTURER	MFG MODEL #	QTY	DESCRIPTION	LIST PRICE	EXT. LIST	SALE PRICE	EXTEND PRICE
1	SUN	A86-FPZ2BH8GKBA	30	Sun Fire X4100 M2 X64 Server, 2x AMD Opteron Model 2220, (2.8 GHz) processor, 4 x 2GB DDR	\$6,590.00	\$197,700.00	\$5,667.40	\$170,022.00
2	SUN	4226A-Z	60	4 GB Memory Kit DDR2-667 DIMMs (2x2GB) for Sun Fire X4100 M2	\$750.00	\$45,000.00	\$645.00	\$38,700.00
3	SUN	WD9-A87-24-3H	30	Sun Fire X4100 M2 Server upgrade to 3 yrs. 7x24 hardware only support with 4 hour response	\$792.00	\$23,760.00	\$712.80	\$21,384.00
4	SUN	X311L	60	NORTH AMERICAN/ASIAN POWER CORD	\$0.00	\$0.00	\$0.00	\$0.00
5	CISCO	WS-C3750-24TS-E	4	Cisco Catalyst 3750-24TS 24 port switch	\$5,990.00	\$23,960.00	\$3,750.53	\$15,002.11
6	CISCO	CON-OSP-375024TE	12	Cisco Catalyst 3750-24TS 24 port switch 1 year Onsite 24x7x4 service	\$1,006.00	\$12,072.00	\$894.22	\$10,730.67
7	BELKIN	A3L791-05-GRN	60	Belkin Cat5e Patch Cable	\$5.99	\$359.40	\$4.86	\$291.79
8	REDHAT	MCT0798-C	30	RHEL ES PREMIUM 3YR	\$3,702.00	\$111,060.00	\$2,234.00	\$67,020.00

TOTAL \$323,150.56

**Applicable tax, freight and insurance charges will be included on your invoice.*



Call Me for a formal GE Finance Lease Proposal (732) 748-3627
 Estimated Lease Rate:

Lease the items on this quote for \$10341 / month

/month based on the items listed on this quotation and a 36 month fair market value lease payment at the current rate of interest. Actual rate may vary.

Terms: Subject to Credit Approval; and subject to credit review.

Above Sales Quotation is subjected to CR's general terms and conditions outlined at www.conres.com
 Pricing subject to change. Delivery is conveyed F.O.B. shipping point. Title on Sales passes upon payment in full. Title on all rental or leased equipment remains with Continental Resources, Inc., and or Continental Leasing Co., Inc. Risk of loss is FOB shipping point. Payment of Freight Insurance modifies Risk of Loss to FOB destination. Taxes, Freight and Insurance be included in above quote. Return rights are restricted to vendor or manufacturers policy in existence at time of return. Third party leases must be identified before shipment and Lessor must be judged credit worthy by CRI. All payments are in United States International Terms and Conditions.

License and Maintenance Agreements: If customer agrees to purchase any items that carry a license or maintenance agreement and if invoice(s) for these product(s) is (are) not paid within approved credit terms, Continental Resources, Inc. reserves the right to, and customer grants permission to, revoke the agreement(s). All payment terms are in United States Dollars. All international credit granted is governed by The United Nations Convention on Contract International Sale of Goods (CISG) and all letters of credit are governed by the Uniform Customs and Practice for Documentary Credits (1 Revision International Chamber of Commerce Publication No. 500).

For our unabridged terms and conditions please visit: http://conres.com/corporate/leasing/terms_96.html
 Note: If you are receiving this quote via email attachment, and do not have Adobe Reader, you can download it from our Web site at: http://www.conres.com/corporate/New_s.html