Sun microsystems

THE NETWORK IS THE COMPUTER™

TPC Benchmark™ H
Full Disclosure Report

Sun Microsystems Sun Fire™ V240 Server
Using Sybase IQ 12.5

Submitted for Review
Report Date: June 23, 2003

TPC Benchmark H Full Disclosure Report

First Printing

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

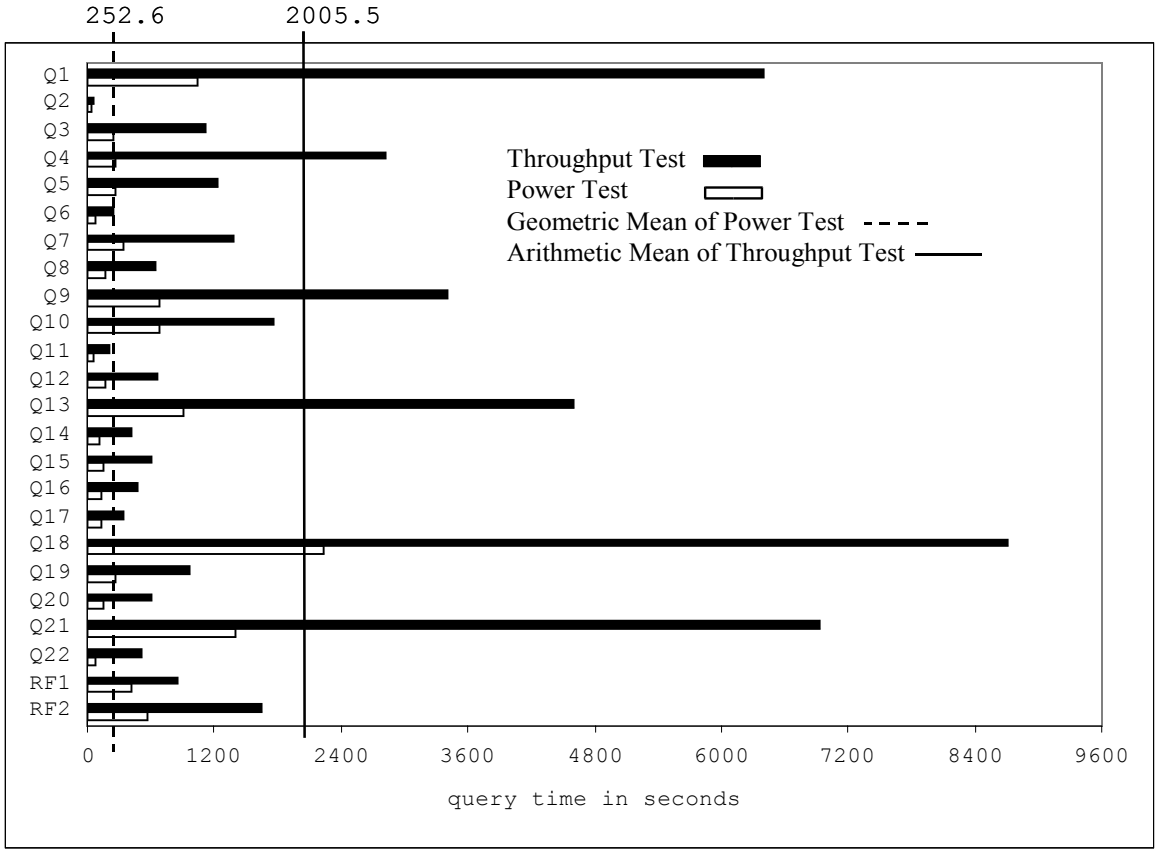
Sun Fire™ V240 Server with Sybase IQ 12.5

TPC-H Rev. 2.0

June 23, 2003

Total System Cost	Composite Query per Hour Metric	Price/Performance
$45,021	**1124.4** QphH@100GB	**$40** $/QphH@100GB

Database Size	Database Manager	Operating System	Other Software	Availability Date
100GB	Sybase IQ 12.5	Solaris 9	Solaris Volume Manager	June 23, 2003



Database Load Time = 6:16	Load Includes Backup: N	Total Data Storage/Database Size=4.07
RAID (Base tables): Y	RAID (Base tables and auxiliary data structures): Y	RAID (All): N

System Configuration: Sun Fire™ V240 Server
Processors: 2 UltraSPARC™ IIIi Cu 1 GHz processors
Memory: 8 GB memory
Disks: 2 internal disks (36.4 GB), 1 SE3310 (10x36.4GB plus 2 spares)
Total Storage: 406.8 GB (in this calculation one GB is defined as 1024*1024*1024 bytes)

Sun Fire™ V240 Server with Sybase IQ 12.5

TPC-H Rev. 2.0

June 23, 2003

Description	Part Number	Source	Unit Price	Qty	Ext. Price	3 Yr. maint.
Server Hardware						
Sun Fire V240 2*1GHz, 2*36.4GB disk	N32-XUB2-9S-204AV2	1	5,781	1	5,781	5,012
2GB Memory Expansion Kit (2X1GB DDR PC2100 ECC) Sunfire V240	X7404A	1	1,243	4	4,972	
2M-SCSI-Cable	X1138A	2	82	2	164	
Dual-Ultra-3-SCSI-HBA	X6758A	2	571	1	571	
Server HardwareSubtotal					11,488	5,012
Storage						
SE3310-436GB-12x36-JBOD 10k rpm	XTA3310R01A0R436	2	8,565	1	8,565	
Storage Subtotal					8,565	
Server Software						
Solaris 9 CD SPARC Edition Slim kit	SOLZS-090C9AYS	1	36	1	36	
Sybase IQ-M Single App Svr - 2 cpu's	12841	3	15,000	1	15,000	
Sybase IQ 3 Years Extended Support 24 x 7	98480	3		1		7,920
Sybase discount (licence and support)		3			-3,000	
Server Software Subtotal					12,036	7,920
			Total		32,089	12,932
Service for all Sun products is from Sun Microsystems, Inc.			**3 Yr. Cost**		45,021	
Service for Sybase products is from Sybase Inc.			**QphH@ 100GB**		1,124.40	
			$/QphH@ 100GB		$40	

Notes (Source):

1. Integrity Networking Systems, Inc.
2. Continental Resources, Inc.
3. Sybase Inc.

Price Quotes in Appendix G

Audited by: Brad Askins, InfoSizing, Inc. (www.sizing.com)

Sun microsystems SYBASE	**Sun Fire™ V240 Server with Sybase IQ 12.5**	**TPC-H Rev. 2.0** **June 23, 2003**

Numerical Quantities

Measurement Results:

Database Scale Factor	= 100GB
Total Data Storage / Database Size	= 4.07
Start of database load time	= 2003-05-31 11:04:32
End of database load time	= 2003-05-31 17:19:59
Database Load Time	= 6:16
Query Streams for Throughput Test	= 5
TPC-H Power	= 1425.1
TPC-H Throughput	= 887.2
TPC-H Composite Query-per-Hour Rating (QphH@100GB)	= 1124.4
Total System Price Over 3 Years	= $45,021
TPC-H Price/Performance Metric ($/QphH@100GB)	= $40

Measurement Intervals:

Measurement Interval in Throughput Test (Ts)	= 44,637 seconds

Duration of Stream Execution:

Stream ID	Seed	Start Date	Start Time	End Date	End Time	Duration
Stream 00	531171959	Jun 1, 2003	08:38:25	Jun 1, 2003	11:34:27	02:56:02
Stream 01	531171960	Jun 1, 2003	11:34:28	Jun 1, 2003	23:28:00	11:53:32
Stream 02	531171961	Jun 1, 2003	11:34:32	Jun 1, 2003	23:58:25	12:23:53
Stream 03	531171962	Jun 1, 2003	11:34:36	Jun 1, 2003	23:52:55	12:18:19
Stream 04	531171963	Jun 1, 2003	11:34:41	Jun 1, 2003	23:57:23	12:22:42
Stream 05	531171964	Jun 1, 2003	11:34:45	Jun 1, 2003	23:53:35	12:18:50
Refresh		Jun 1, 2003	11:34:49	Jun 1, 2003	23:47:39	12:12:50

Sun Fire™ V240 Server with Sybase IQ 12.5

TPC-H Rev. 2.0

June 23, 2003

TPC-H Timing Intervals (in seconds)

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Stream 00	1040.2	30.1	236.9	259	272	67.5	338.8	169.7	680.5	685.9	47.8	161.6
Stream 01	7347	50.7	917.4	2499	885	270.5	1302.8	376.8	2636.1	1559.2	183.3	399.5
Stream 02	6276.7	67.2	1546	2876.9	1221.7	314.5	1177.6	688.2	4444.4	1839.5	326.1	605.6
Stream 03	6303.5	52.4	917.9	3012.5	1178.3	330.5	1794	1169.8	3086.6	1534.1	139.7	818.9
Stream 04	5972.2	63.8	899.7	2638.5	1920	121.9	1322.4	411.1	3573.4	1979.7	126.7	540.5
Stream 05	6138.6	53.9	1335.9	3058.7	961.8	151.1	1360.1	612	3332	1847.1	228.8	936.9
Minimum	5972.2	50.7	899.7	2499	885	121.9	1177.6	376.8	2636.1	1534.1	126.7	399.5
Average	6407.6	57.6	1123.4	2817.1	1233.4	237.7	1391.4	651.6	3414.5	1751.9	200.9	660.3
Maximum	7347	67.2	1546	3058.7	1920	330.5	1794	1169.8	4444.4	1979.7	326.1	936.9

	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream 00	908.2	104.5	154.7	126	131.8	2240.5	257.3	160.6	1407.1	82.8	415.8	571.4
Stream 01	4214.9	325.3	492.4	331	204.1	8962.7	1313	546.4	7650.3	339.5	1115.5	1985.7
Stream 02	4061.4	457.2	499.4	660.3	588.5	9334.8	981.9	748.6	5303.7	603.9	760.5	1560.8
Stream 03	4983	502	591.3	545.2	306.8	8128.2	879.4	1041.8	6497.9	480	685	1646.3
Stream 04	5284.9	347.2	1033.2	338.2	236.3	8907	974.4	200.3	7013.2	650.7	778.2	1614.8
Stream 05	4438.6	443.7	459.8	483.4	346	8223.3	709.8	520.1	8175.6	506.4	923.5	1396.7
Minimum	4061.4	325.3	459.8	331	204.1	8128.2	709.8	200.3	5303.7	339.5	685	1396.7
Average	4596.6	415.1	615.2	471.6	336.3	8711.2	971.7	611.4	6928.1	516.1	852.5	1640.9
Maximum	5284.9	502	1033.2	660.3	588.5	9334.8	1313	1041.8	8175.6	650.7	1115.5	1985.7

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

Benchmark Sponsors:

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

Eric Miles
Sr. VP & General Manager
Sybase Business Intelligence
561 Virginia Road
Concord, MA 01742

June 19, 2003

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

Platform: **Sun Fire V240 Server**

Database Manager: **Sybase IQ 12.5**

Operating System: **Solaris 9**

The results were:

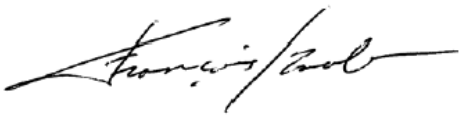
CPU (Speed)	Memory	Disks	**QphH@100GB**
Sun Fire 240			
2 x UltraSPARC IIIi Cu (1 GHz)	8 GB Main	10 x 36.4 GB 2 x 36.4 GB	**1124.4**

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

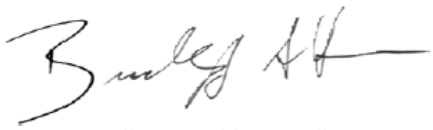
- The database records were defined with the proper layout and size
- The database population was generated using DBGEN
- The database was properly scaled to 100GB and populated accordingly
- The compliance of the database auxiliary data structures was verified
- The database load time was correctly 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 5 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

Respectfully Yours,



François Raab, President



Bradley J. Askins, Auditor

1373 North Franklin Street • Colorado Springs, CO 80903-2527 • Office: 719/473-7555 • Fax: 719/473-7554

Table of Contents

TPC Benchmark H Overview

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

- Examine large volumes of data
- Execute queries with a high degree of complexity
- Give answers to critical business questions

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

- Give answers to real-world business questions
- Simulate generated ad-hoc queries
- Are far more complex than most OLTP transactions
- Include a rich breadth of operators and selectivity constraints
- Generate intensive activity on the part of the database server component of the system under test
- Are executed against a database complying to specific population and scaling requirements
- Are implemented with constraints derived from staying closely synchronized with an on-line production database

1. General Items

1.1 Benchmark Sponsor

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

Sun Microsystems, Inc. and Sybase 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 Solaris and Sybase IQ 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.

Sun Fire™ V240 Server, priced configured with:

- 2 UltraSPARC™ IIIi Cu 1 GHz processors
- 8 GB memory
- 1 PCI Dual Ultra-3 SCSI HBA
- 1 Sun StorEdge 3310 SCSI Array containing 12 x 36.4 GB disk drives
- 2 internal disk 36.4 GB disk
- 2 SCSI cables 2 meters

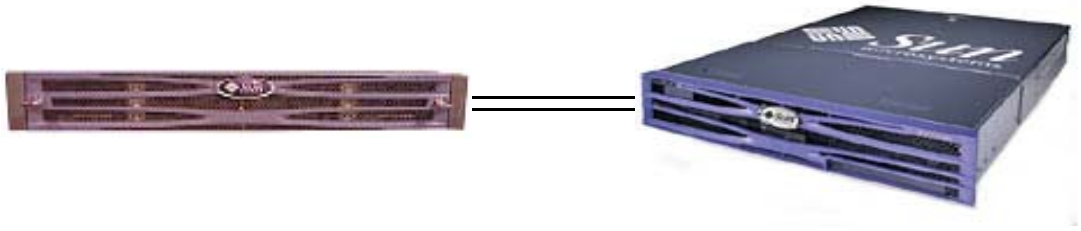
The tested configuration is identical to the priced configuration.

Sun Fire V240 Server

2 x UltraSPARC™ IIIi Cu 1 Ghz

8 GB Memory

2 x 36.4 GB internal disks



StorEdge™ 3310 SCSI Array

1 x (12 x 36.4GB disks - 10 used, 2 spares)

IQ Main Database 149.0 GB RAID 1 (2 x = 298.0 GB)

IQ Temp Database 37.76 GB Non-RAID

See section 5.2 for detailed disk configuration information

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 scripts that create the tables and indexes for the TPC-H database.

2.2 Physical Organization

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

No record clustering or index clustering was used.

2.3 Horizontal Partitioning

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

No horizontal partitioning was used.

2.4 Replication

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

No replication was used.

3. Clause 2 Queries and Refresh Functions

3.1 Query Language

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

SQL was the query language used to implement all queries.

3.2 Verifying Method for Random Number Generation

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

TPC supplied versions 1.3.0 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 1.3.0 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 Q1, Q4, Q5, Q6, Q10, Q12, Q14, Q15 and Q20, the "dateadd" function is used to perform date arithmetic.
- In Q7, Q8 and Q9, the "datepart" function is used to extract part of a date (e.g., "year").
- In Q2, Q3, Q10, Q18 and Q21, the "top" 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 (repeatable read).

3.7 Source Code of Refresh Functions

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

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 test 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 six execution streams.
3. The consistency of the ORDERS and LINEITEM tables was re-verified.

4.4 Isolation

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

4.4.1 Read-Write Conflict with Commit

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

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

4.4.2 Read-Write Conflict with Rollback

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

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

4.4.3 Write-Write Conflict with Commit

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

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

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

4.4.4 Write-Write Conflict with Rollback

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

1. An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. T1 was suspended prior to ROLLBACK.
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 Q1 against the qualification database, was started using a randomly selected DELTA.
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 Q1.

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.

The disks containing TPC-H tables, indexes, catalog file and catalog log file were mirrored. During the durability test a disk containing tables and indexes and a database temp device was removed from its cabinet. The database server haulted when it could no longer access the unmirrored temp device. After the disk was replaced, the database server was restarted and the effects of all committed updates became immediately visible.

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.

The system crash and memory failure tests were combined. Power to the server was turned off during the durability test. When power was restored, the system rebooted and the database was 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).

See section 4.5.2.

5. Clause 4 Scaling and Database Population

5.1 Ending Cardinality of Tables

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

Table	*Rows*
Lineitem	*600,037,902*
Orders	*150,000,000*
Partsupp	*80,000,000*
Part	*20,000,000*
Customer	*15,000,000*
Supplier	*1,000,000*
Nation	*25*
Region	*5*

5.2 Distribution of Tables and Logs Across Media

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

- All tables and indexes were mirrored across 5 volumes on the StorEdge 3310. Each volume was a RAID 1 device created using the Solaris Volume Manager.
- The Sybase IQ Temp dbspaces were configured using 10 raw partitions on the StorEdge 3310. The temp space was not mirrored.

All Data and Indexes were striped across the five RAID 1 volumes using Sybase IQ striping. Mirroring was performed by Solaris Volume Manager. The following two tables show the distribution of tables and logs across media.

The following table shows the file systems for the operating system and sybase software.

Disk Size GB	Partition	Use
33.9	c1t0d0s0	/root
	c1t0d0s1	/swap
33.9	c2t8d0s1	Solaris Volume Manager - metadb
33.9	c2t9d0s1	Solaris Volume Manager – metadb

Each of the 10 disks in the StorEdge 3310 Array was split into two partitions using Solaris format, a larger partition for IQ Main Store and a small partition for IQ Temp Store. Each IQ Main Store device is constructed from two 30GB partitions on two disks using Solaris Volume Manager RAID 1. A smaller 3.9GB partition is created on each of the 10 disks in the StorEdge 3310 Arrays, these smaller un-mirrored raw partition are combined to form the IQ Temp Store. This table shows the devices used by the database server for data storage and temp space. The /sybase2 file system used to store the IQ catalog file and catalog log resides on a small RAID 1 partition on the SE3310.

Database Storage Summary:

IQ Main Storage: 149.0 GB RAID 1 (2 x = 298.0 GB)

IQ Temp Storage: 37.76 GB Non-RAID

Database Device Details:

Disk Size GB	Raw Partition Name	SVM Device Name	Mirror Primary Secondary	Symbolic Link	Database Usage	Database Device Size GB	RAID
33.9	c2t10d0s1	d2	P		/sybase2	1.00	RAID 1
33.9	c2t11d0s1	d3	S		/sybase2	0.00	RAID 1
33.9	c2t8d0s4	d5	P	M01	IQ Main	30.00	RAID 1
33.9	c2t9d0s4	d6	S		IQ Main	0.00	RAID 1
33.9	c2t10d0s4	d7	P	M02	IQ Main	30.00	RAID 1
33.9	c2t11d0s4	d8	S		IQ Main	0.00	RAID 1
33.9	c2t12d0s4	d9	P	M03	IQ Main	29.50	RAID 1
33.9	c3t8d0s4	d10	S		IQ Main	0.00	RAID 1
33.9	c3t9d0s4	d11	P	M04	IQ Main	29.50	RAID 1
33.9	c3t10d0s4	d12	S		IQ Main	0.00	RAID 1
33.9	c3t11d0s4	d13	P	M05	IQ Main	30.00	RAID 1
33.9	c3t12d0s4	d14	S		IQ Main	0.00	RAID 1
33.9	c2t8d0s5			T01	IQ Temp	3.90	None
33.9	c2t9d0s5			T02	IQ Temp	3.90	None
33.9	c2t10d0s5			T03	IQ Temp	3.90	None
33.9	c2t11d0s5			T04	IQ Temp	3.90	None
33.9	c2t12d0s5			T05	IQ Temp	3.90	None
33.9	c3t8d0s5			T06	IQ Temp	3.78	None
33.9	c3t9d0s5			T07	IQ Temp	3.78	None
33.9	c3t10d0s5			T08	IQ Temp	3.40	None
33.9	c3t11d0s5			T09	IQ Temp	3.40	None
33.9	c3t12d0s5			T10	IQ Temp	3.90	None

Additional details can be found in the disk configuration section in Appendix B.

5.3 Database partition/replication mapping

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

Database partitioning/replication was not 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. In addition, the Sybase IQ root .db file and log space were also protected by RAID 1.

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 1.3.0 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 6 hours 16 minutes.

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.

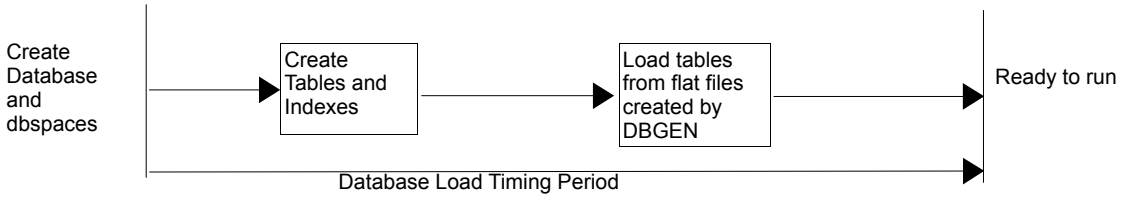
The data storage ratio is computed from the following information:

Disk Type	# Of Disks	Space Per Disk*	Sub-Total Disk Space**
3310 disk array	10	36.4 GB	339.0 GB
internal	2	36.4 GB	67.8 GB
		Total Space	**406.8 GB**
		Data Storage Ratio	**4.07**

* *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.

5.9 Qualification Database Configuration

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

The qualification database used identical scripts to create and load the data with adjustments for the size difference.

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

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 given in the Numerical Quantity Summary earlier in this document.

6.4 Number of Streams for the Throughput Test

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

Five streams 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 throughput test start time and finish time for each stream are contained in the Numerical Quantity Summary earlier in this document.

6.6 Total Elapsed Time of the Measurement Interval

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

The total elapsed time of the throughput test is contained in the Numerical Quantity Summary earlier in this document.

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

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

The start and finish times for each refresh function:

Stream ID	Refresh Function	Start Date	Start Time	End Date	End Time
Stream 01	RF1	Jun 1, 2003	11:34:49	Jun 1, 2003	11:53:24
Stream 01	RF2	Jun 1, 2003	12:51:44	Jun 1, 2003	13:24:50
Stream 02	RF1	Jun 1, 2003	14:23:10	Jun 1, 2003	14:35:51
Stream 02	RF2	Jun 1, 2003	15:34:11	Jun 1, 2003	16:00:12
Stream 03	RF1	Jun 1, 2003	16:58:32	Jun 1, 2003	17:09:57
Stream 03	RF2	Jun 1, 2003	18:08:18	Jun 1, 2003	18:35:44
Stream 04	RF1	Jun 1, 2003	19:34:04	Jun 1, 2003	19:47:03
Stream 04	RF2	Jun 1, 2003	20:45:23	Jun 1, 2003	21:12:18
Stream 05	RF1	Jun 1, 2003	22:10:38	Jun 1, 2003	22:26:02
Stream 05	RF2	Jun 1, 2003	23:24:22	Jun 1, 2003	23:47:39

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 contained in the Numerical Quantity Summary earlier in this document.

6.9 Performance Metrics

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

The performance metrics, and the numbers on which they are based, are contained in the Numerical Quantity Summary earlier in this document.

6.10 The Performance Metric and Numerical Quantities from Both Runs

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

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

Run ID	QppH@100GB	QthH@100GB	QphH@100GB
Run 1	1427.8	892.9	1129.1
Run 2	1425.1	887.2	1124.4
% Difference	-0.19	-0.54	-0.42

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.

The database was not restarted after it was loaded or between the two runs.

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 do_test script found in Appendix E, the scripts that do_test executes can be found in Appendix B.

The Power Test and Throughput Test are performed by the do_test. The query streams are generated by a script called gen_streams.ksh which uses QGEN to generate the query stream files.

The Power Test is invoked within the do_test script by executing the update_power.sql which runs the refresh functions and the stream0.sql which contains the power stream queries.

The Throughput Test is invoked within the do_test script by executing the five query stream scripts stream[1-5].sql files along with simultaneously executing the update_throughput.sql script. The refresh functions execute in parallel with the ad-hoc queries streams trickle feeding the refresh transactions.

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.

dbisqlc is a Sybase database utility that allows SQL to be executed against an Sybase IQ database. The dbisqlc utility is invoked from command-line mode on the SUT. dbisqlc reads an input file with the SQL commands and sends output to standard output. The dbisqlc utility uses information in the .odbc.ini file to connect to the database.

dbtest is a utility used internally at Sybase for testing , it facilitates access and manipulation of data in a Sybase IQ database. dbtest is invoked from command-line mode on the SUT, specifying access to the TPC-H database. It runs an input file containing either the QGEN generated SQL for the queries or SQL for the refresh functions. The dbtest utility uses information in the .odbc.ini file to connect to the database.

7.3 Profile-Directed Optimization

If profile-directed optimization as described in Clause 5.2.9 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.

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 $45,021. 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 by Integrity Networking Systems Inc., Continental Resources Inc.and Sybase 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.

Hardware and Software components will be available June 23, 2003.

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 is included at the front of this report.

Appendix A. Solaris 9 and Sybase IQ 12.5 Parameters

This Appendix contains Solaris kernel parameters and environment variables and Sybase IQ system parameters.

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

Sybase IQ Server Configuration Parameters

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

tpch.cfg

```
-n tpch
-c  32m
-gd all
-gl all
-gm 10
-gp 4096
-ti 4400
-tl 300
-iqmc 4500
-iqtc 2400
-iqmt 600
-iqgovern 12
-iqpartition 2
```

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

Sybase IQ Database Options

(altered from default)

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

options.sql

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

SET OPTION PUBLIC.Allow_Nulls_By_Default='Off';
SET OPTION PUBLIC.Append_Load='On';
SET OPTION PUBLIC.Flatten_Subqueries = 'On';
SET OPTION PUBLIC.Force_No_Scroll_Cursors='On';
SET OPTION PUBLIC.Garray_Fill_Factor_Percent=5;
SET OPTION PUBLIC.Load_Memory_Mb=0;
SET OPTION PUBLIC.Max_Hash_Rows=9000000;
SET OPTION PUBLIC.Max_Iq_Threads_Per_Connection=100;
SET OPTION PUBLIC.Minimize_Storage='On';
SET OPTION PUBLIC.Notify_Modulus=10000000;
SET OPTION PUBLIC.Query_Temp_Space_Limit=0;
SET OPTION PUBLIC.Row_Counts='On';
SET OPTION PUBLIC.Sort_Phase1_Helpers=4;
SET OPTION PUBLIC.Sweeper_Threads_Percent=15;
SET OPTION PUBLIC.Dml_Options10 = 'On';
SET OPTION PUBLIC.Query_Detail='On';
SET OPTION PUBLIC.Query_Plan_After_Run='On';
SET OPTION PUBLIC.Query_Plan_As_Html='On';
SET OPTION PUBLIC.Query_Plan='On';
```

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

Sybase IQ Environment Variables

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

SYBASE="/export/home/sybase"
export SYBASE
SYBASE_OCS="OCS-12_5"
export SYBASE_OCS
SYBASE_JRE="${SYBASE}/shared/jre-1_22"
export SYBASE_JRE
ASDIR="${SYBASE}/ASIQ-12_5"
export ASDIR
PATH="${ASDIR}/bin:${SYBASE}/${SYBASE_OCS}/bin:
${PATH}:/etc:."
export PATH
IQLIB="${ASDIR}/usrlib:${ASDIR}/lib:${SYBASE}
/${SYBASE_OCS}/lib"
LD_LIBRARY_PATH_64="${IQLIB}:$LD_LIBRARY_PATH_64"
export LD_LIBRARY_PATH_64
LD_LIBRARY_PATH="${IQLIB}:$LD_LIBRARY_PATH"
export LD_LIBRARY_PATH
unset IQLIB
LD_PRELOAD=mpss.so.1
export LD_PRELOAD
MPSSHEAP=512K
MPSSSTACK=64K
export MPSSHEAP
export MPSSSTACK
```

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

.odbc.ini

```
================================================
[ODBC Data Sources]
tpch=ASIQ Driver
utility_db=ASIQ Driver

[tpch]
Driver=/export/home/sybase/asiq12/lib/dbodbc7_r.so.1
EngineName=tpch1000
CommLinks=tcpip{host=10.8.5.67;Port=2638}
DatabaseName=tpch
UserID=DBA
Password=SQL
DBG=yes
LOG=/export/home/sybase/tpch1000_odbc.log

[utility_db]
Driver=/export/home/sybase/asiq12/lib/dbodbc7_r.so.1
EngineName=tpch
CommLinks=tcpip{host=10.8.5.67;Port=2638}
DatabaseName=utility_db
UserID=DBA
Password=SQL
DBG=yes
LOG=/export/home/sybase/utility_db_odbc.log
```

Solaris Parameters

(altered from default)

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

/etc/system

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

set pt_cnt=1024
set npty=1024
set sadcnt=2048
set nautopush=1024
set dosynctodr=0
```

Appendix B. Programs and Scripts

==========

check_query1.bash

==========

```
#!/bin/bash
#
# First remove the rf1.lock so that the Query Stream
will start
#
rm -f /export/home/sybase/run/scripts/rf1.lock
#
# Sleep while the rf2.lock file exists
# when the query stream completes it will remove the
rf2.lock
#
while [ -f /export/home/sybase/run/scripts/rf2.lock ]
do
 # Wait for the Query Steam to complete
 # check every 10 seconds
 # echo "Lock File Exists"
 sleep 10
done
# Return Control to the RF stream
```

==========

create_database.sql

==========

```
CREATE DATABASE '/sybase2/tpch.db'
TRANSACTION LOG ON
COLLATION 'ISO_BINENG'
CASE RESPECT
PAGE SIZE 4096
BLANK PADDING ON
JAVA ON
JCONNECT ON
IQ PATH '/sybase2/M01'
IQ PAGE SIZE 524288
TEMPORARY PATH '/sybase2/T01';
```

==========

create_dbspaces.sql

==========

```
create dbspace iq2 as '/sybase2/M02' iq store;
create dbspace iq3 as '/sybase2/M03' iq store;
create dbspace iq4 as '/sybase2/M04' iq store;
create dbspace iq5 as '/sybase2/M05' iq store;
create dbspace iqtmp2 as '/sybase2/T02' iq temporary
store;
create dbspace iqtmp3 as '/sybase2/T03' iq temporary
store;
create dbspace iqtmp4 as '/sybase2/T04' iq temporary
store;
create dbspace iqtmp5 as '/sybase2/T05' iq temporary
store;
create dbspace iqtmp6 as '/sybase2/T06' iq temporary
store;
create dbspace iqtmp7 as '/sybase2/T07' iq temporary
store;
create dbspace iqtmp8 as '/sybase2/T08' iq temporary
store;
create dbspace iqtmp9 as '/sybase2/T09' iq temporary
store;
create dbspace iqtmp10 as '/sybase2/T10' iq temporary
store;
```

==========

create_tables.sql

==========

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


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


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


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


CREATE TABLE partsupp
  (
    ps_partkey             unsigned int,
    ps_suppkey             unsigned int,
    ps_availqty            integer,
    ps_supplycost          double precision,
    ps_comment             varchar(199),
    PRIMARY KEY (ps_partkey, ps_suppkey)
  );
CREATE HG INDEX ps_partkey_hg ON partsupp(ps_partkey)
;
CREATE HG INDEX ps_suppkey_hg ON partsupp(ps_suppkey)
;
```

```
CREATE TABLE customer
  (
    c_custkey             unsigned int,
    c_name                varchar(25),
    c_address             varchar(40),
    c_nationkey           unsigned int,
    c_phone               char(15),
    c_acctbal             double precision,
    c_mktsegment          char(10),
    c_comment             varchar(117),
    PRIMARY KEY(c_custkey)
  );
CREATE HG INDEX c_nationkey_hg ON
customer(c_nationkey) ;


CREATE TABLE orders
  (
    o_orderkey            unsigned int,
    o_custkey             unsigned int,
    o_orderstatus         char(1),
    o_totalprice          double precision,
    o_orderdate           date,
    o_orderpriority       char(15),
    o_clerk               char(15),
    o_shippriority        int,
    o_comment             varchar(79),
    PRIMARY KEY (o_orderkey)
  );
CREATE HG INDEX o_custkey_hg ON orders(o_custkey) ;
CREATE DATE INDEX o_orderdate_date ON
orders(o_orderdate) ;


CREATE TABLE lineitem
  (
    l_orderkey            unsigned int,
    l_partkey             unsigned int,
    l_suppkey             unsigned int,
    l_linenumber          int,
    l_quantity            double precision,
    l_extendedprice       double precision,
    l_discount            double precision,
    l_tax                 double precision,
    l_returnflag          char(1),
    l_linestatus          char(1),
    l_shipdate            date,
    l_commitdate          date,
    l_receiptdate         date,
    l_shipinstruct        char(25),
    l_shipmode            char(10),
    l_comment             varchar(44)
  );

CREATE HG INDEX l_partsupp_hg ON lineitem(l_partkey,
l_suppkey) ;
CREATE HG INDEX l_orderkey_hg ON lineitem(l_orderkey)
;
CREATE HG INDEX l_partkey_hg ON lineitem(l_partkey) ;
CREATE HG INDEX l_suppkey_hg ON lineitem(l_suppkey) ;
CREATE DATE INDEX l_shipdate_date ON
lineitem(l_shipdate) ;
CREATE DATE INDEX l_receiptdate_date ON
lineitem(l_receiptdate);
```

==========

tpch_rf.sql

==========

```
create table refresh_control ( rf1_data_set int not
null, rf2_data_set int not null);
insert into refresh_control values (0,0);
commit;
CREATE PROCEDURE DBA.tpch_rf1 (IN c_directory
varchar(128),
                               IN c_stream varchar(3))
ON EXCEPTION RESUME
BEGIN
  DECLARE delim_asci integer;
  DECLARE c_data_set varchar(3);
  DECLARE i_data_set integer;
  DECLARE c_cmd long varchar;
  DECLARE outfilename varchar(128); -- Debug
  DECLARE outfilename2 varchar(128); -- Debug
  DECLARE c_lf varchar(2);
  DECLARE t_qstart timestamp;
  DECLARE t_qstop timestamp;
  DECLARE n_seconds numeric(12,5);
  DECLARE c_sqlstate CHAR(5);
  SET t_qstart = now(*);
  SET c_lf=char(10);
  SELECT rf1_data_set INTO i_data_set FROM
refresh_control;
  SET c_data_set=CAST(i_data_set+1 AS varchar(3));
  SET c_cmd='load table orders ( '+c_lf;
  SET c_cmd=c_cmd+' o_orderkey
'+char(39)+'|'+char(39)+', '+c_lf;
  SET c_cmd=c_cmd+' o_custkey
'+char(39)+'|'+char(39)+', '+c_lf;
  SET c_cmd=c_cmd+' o_orderstatus
'+char(39)+'|'+char(39)+', '+c_lf;
  SET c_cmd=c_cmd+' o_totalprice
'+char(39)+'|'+char(39)+', '+c_lf;
  SET c_cmd=c_cmd+' o_orderdate date('+char(39)+'YYYY-
MM-DD'+char(39)+'), filler(1), '+c_lf;
  SET c_cmd=c_cmd+' o_orderpriority
'+char(39)+'|'+char(39)+', '+c_lf;
  SET c_cmd=c_cmd+' o_clerk '+char(39)+'|'+char(39)+',
'+c_lf;
  SET c_cmd=c_cmd+' o_shippriority
'+char(39)+'|'+char(39)+','+c_lf;
  SET c_cmd=c_cmd+' o_comment
'+char(39)+'|'+char(39)+' ) '+c_lf;
  SET c_cmd=c_cmd+'from
'+char(39)+c_directory+'orders.tbl.u'+c_data_set+char(
39)+c_lf;
  SET c_cmd=c_cmd+'row delimited by
'+char(39)+'\\x0a'+char(39)+' quotes off escapes off
preview on;';
  EXECUTE IMMEDIATE c_cmd;
  SELECT SQLSTATE INTO c_sqlstate;
  IF c_sqlstate != '00000' THEN
    ROLLBACK;
    RAISERROR 23002 'RF1 failed at Step 1 with
SQLSTATE: ', c_sqlstate;
    RETURN(1);
  END IF;
  SET c_cmd='load table lineitem ( '+c_lf;
  SET c_cmd=c_cmd+' l_orderkey
'+char(39)+'|'+char(39)+', '+c_lf;
  SET c_cmd=c_cmd+' l_partkey
'+char(39)+'|'+char(39)+', '+c_lf;
  SET c_cmd=c_cmd+' l_suppkey
'+char(39)+'|'+char(39)+', '+c_lf;
  SET c_cmd=c_cmd+' l_linenumber
'+char(39)+'|'+char(39)+', '+c_lf;
  SET c_cmd=c_cmd+' l_quantity
'+char(39)+'|'+char(39)+', '+c_lf;
  SET c_cmd=c_cmd+' l_extendedprice
'+char(39)+'|'+char(39)+', '+c_lf;
  SET c_cmd=c_cmd+' l_discount
'+char(39)+'|'+char(39)+', '+c_lf;
```

```
  SET c_cmd=c_cmd+' l_tax '+char(39)+'|'+char(39)+',
'+c_lf;
  SET c_cmd=c_cmd+' l_returnflag
'+char(39)+'|'+char(39)+', '+c_lf;
  SET c_cmd=c_cmd+' l_linestatus
'+char(39)+'|'+char(39)+', '+c_lf;
  SET c_cmd=c_cmd+' l_shipdate date('+char(39)+'YYYY-
MM-DD'+char(39)+'), filler(1), '+c_lf;
  SET c_cmd=c_cmd+' l_commitdate
date('+char(39)+'YYYY-MM-DD'+char(39)+'), filler(1),
'+c_lf;
  SET c_cmd=c_cmd+' l_receiptdate
date('+char(39)+'YYYY-MM-DD'+char(39)+'), filler(1),
'+c_lf;
  SET c_cmd=c_cmd+' l_shipinstruct
'+char(39)+'|'+char(39)+', '+c_lf;
  SET c_cmd=c_cmd+' l_shipmode
'+char(39)+'|'+char(39)+', '+c_lf;
  SET c_cmd=c_cmd+' l_comment
'+char(39)+'|'+char(39)+' )'+c_lf;
  SET c_cmd=c_cmd+'from
'+char(39)+c_directory+'lineitem.tbl.u'+c_data_set+cha
r(39)+c_lf;
  SET c_cmd=c_cmd+'row delimited by
'+char(39)+'\\x0a'+char(39)+c_lf+'quotes off escapes
off preview on;';
  EXECUTE IMMEDIATE c_cmd;
  SELECT SQLSTATE INTO c_sqlstate;
  IF c_sqlstate != '00000' THEN
    rollback;
    RAISERROR 23002 'RF1 failed at Step 2 with
SQLSTATE: ', c_sqlstate;
    RETURN(1);
  END IF;
  UPDATE refresh_control SET
rf1_data_set=cast(c_data_set AS integer);
  COMMIT;
  SET t_qstop = now(*);
  SET n_seconds=cast(datediff(millisecond,t_qstart,
t_qstop) AS numeric(12,5))/1000;
  SET c_cmd='Stream updates Update
update_'+c_stream+'_RF1 LENGTH -- '+cast(n_seconds AS
varchar(20))+ ' seconds' ;
  SELECT c_cmd;
  RETURN(0);
END;
CREATE PROCEDURE DBA.tpch_rf2 (in c_directory
varchar(128),
                               in c_stream varchar(3))
ON exception resume
BEGIN
  DECLARE delim_asci integer;
  DECLARE c_data_set varchar(3);
  DECLARE i_data_set integer;
  DECLARE c_cmd long varchar;
  DECLARE outfilename varchar(128); -- Debug
  DECLARE c_lf varchar(2);
  DECLARE t_qstart timestamp;
  DECLARE t_qstop timestamp;
  DECLARE n_seconds numeric(12,5);
  DECLARE c_sqlstate CHAR(5);
  SET t_qstart = now(*);
  SET c_lf=char(10);
  SELECT rf2_data_set INTO i_data_set FROM
refresh_control;
  SET c_data_set=CAST(i_data_set+1 AS varchar(3));
  CREATE TABLE #delete_table ( d_orderkey UNSIGNED
INT, PRIMARY KEY (d_orderkey) );
  SET c_cmd='load table #delete_table  (d_orderkey
'+char(39)+'\\x0a'+char(39)+') '+c_lf;
  SET c_cmd=c_cmd+'from
'+char(39)+c_directory+'delete.'+c_data_set+char(39)+c
_lf;
  SET c_cmd=c_cmd+'quotes off '+c_lf;
  SET c_cmd=c_cmd+'escapes off; '+c_lf;
  EXECUTE IMMEDIATE c_cmd;
  SELECT SQLSTATE INTO c_sqlstate;
  IF c_sqlstate != '00000' THEN
    ROLLBACK;
    SET c_cmd='RF2 failed at Step 1 with SQLSTATE:
'+c_sqlstate;
    RAISERROR 23002 c_cmd;
    RETURN(1);
  END IF;
  DELETE lineitem FROM lineitem, #delete_table WHERE
l_orderkey = d_orderkey;
  SELECT SQLSTATE INTO c_sqlstate;
  IF c_sqlstate != '00000' THEN
    ROLLBACK;
    SET c_cmd='RF2 failed at Step 2 with SQLSTATE:
'+c_sqlstate;
    RAISERROR 23002 c_cmd;
    RETURN(1);
  END IF;
  DELETE orders FROM orders, #delete_table WHERE
o_orderkey = d_orderkey;
  SELECT SQLSTATE INTO c_sqlstate;
  IF c_sqlstate != '00000' THEN
    ROLLBACK;
    SET c_cmd='RF2 failed at Step 3 with SQLSTATE:
'+c_sqlstate;
    RAISERROR 23002 c_cmd;
    RETURN(1);
  END IF;
  UPDATE refresh_control SET
rf2_data_set=CAST(c_data_set AS integer);
  COMMIT;
  DROP TABLE #delete_table;
  SET t_qstop = now(*);
  SET n_seconds=cast(datediff(millisecond,t_qstart,
t_qstop) as numeric(12,5))/1000;
  SET c_cmd='Stream updates Update
update_'+c_stream+'_RF2 LENGTH -- '+cast(n_seconds as
varchar(20))+ ' seconds' ;
  SELECT c_cmd;
  RETURN(0);
END;
```

==========

load_region.sql

==========

```
LOAD TABLE REGION (
R_REGIONKEY                              '|',
R_NAME                                   '|',
R_COMMENT                                '|'
)
FROM '/sybase_stage/region.tbl'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
```

==========

load_nation.sql

==========

```
LOAD TABLE NATION (
N_NATIONKEY                              '|',
N_NAME                                   '|',
N_REGIONKEY                              '|',
N_COMMENT                                '|'
```

```
)
FROM '/sybase_stage/nation.tbl'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
```

load_customer.sql

```
LOAD TABLE CUSTOMER (
C_CUSTKEY                                '|',
C_NAME                                   '|',
C_ADDRESS                                '|',
C_NATIONKEY                              '|',
C_PHONE                                  '|',
C_ACCTBAL                                '|',
C_MKTSEGMENT                             '|',
C_COMMENT                                '|'
)
FROM '/sybase_stage/customer.tbl.1',
'/sybase_stage/customer.tbl.2',
'/sybase_stage/customer.tbl.3',
'/sybase_stage/customer.tbl.4',
'/sybase_stage/customer.tbl.5',
'/sybase_stage/customer.tbl.6',
'/sybase_stage/customer.tbl.7',
'/sybase_stage/customer.tbl.8'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
```

load_part.sql

```
LOAD TABLE PART (
P_PARTKEY                                '|',
P_NAME                                   '|',
P_MFGR                                   '|',
P_BRAND                                  '|',
P_TYPE                                   '|',
P_SIZE                                   '|',
P_CONTAINER                              '|',
P_RETAILPRICE                            '|',
P_COMMENT                                '|'
)
FROM '/sybase_stage/part.tbl.1',
'/sybase_stage/part.tbl.2',
'/sybase_stage/part.tbl.3',
'/sybase_stage/part.tbl.4',
'/sybase_stage/part.tbl.5',
'/sybase_stage/part.tbl.6',
'/sybase_stage/part.tbl.7',
'/sybase_stage/part.tbl.8'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
```

load_supplier.sql

```
LOAD TABLE SUPPLIER (
S_SUPPKEY                                '|',
S_NAME                                   '|',
S_ADDRESS                                '|',
S_NATIONKEY                              '|',
S_PHONE                                  '|',
S_ACCTBAL                                '|',
S_COMMENT                                '|'
)
FROM '/sybase_stage/supplier.tbl.1',
'/sybase_stage/supplier.tbl.2',
'/sybase_stage/supplier.tbl.3',
'/sybase_stage/supplier.tbl.4',
'/sybase_stage/supplier.tbl.5',
'/sybase_stage/supplier.tbl.6',
'/sybase_stage/supplier.tbl.7',
'/sybase_stage/supplier.tbl.8'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
```

load_partsupp.sql

```
LOAD TABLE PARTSUPP (
PS_PARTKEY                               '|',
PS_SUPPKEY                               '|',
PS_AVAILQTY                              '|',
PS_SUPPLYCOST                            '|',
PS_COMMENT                               '|'
)
FROM '/sybase_stage/partsupp.tbl.1',
     '/sybase_stage/partsupp.tbl.2',
     '/sybase_stage/partsupp.tbl.3',
     '/sybase_stage/partsupp.tbl.4',
     '/sybase_stage/partsupp.tbl.5',
     '/sybase_stage/partsupp.tbl.6',
     '/sybase_stage/partsupp.tbl.7',
     '/sybase_stage/partsupp.tbl.8'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
```

load_orders.sql

```
LOAD TABLE ORDERS (
O_ORDERKEY                                       '|',
O_CUSTKEY                                        '|',
O_ORDERSTATUS                                    '|',
O_TOTALPRICE                                     '|',
O_ORDERDATE                                      '|',
O_ORDERPRIORITY                                  '|',
O_CLERK                                          '|',
O_SHIPPRIORITY                                   '|',
O_COMMENT                                        '|'
)
FROM
'/sybase_stage/orders.tbl.1',
'/sybase_stage/orders.tbl.2',
'/sybase_stage/orders.tbl.3',
'/sybase_stage/orders.tbl.4',
'/sybase_stage/orders.tbl.5',
'/sybase_stage/orders.tbl.6',
'/sybase_stage/orders.tbl.7',
'/sybase_stage/orders.tbl.8',
```

```
'/sybase_stage/orders.tbl.9',
'/sybase_stage/orders.tbl.10',
'/sybase_stage/orders.tbl.11',
'/sybase_stage/orders.tbl.12',
'/sybase_stage/orders.tbl.13',
'/sybase_stage/orders.tbl.14',
'/sybase_stage/orders.tbl.15',
'/sybase_stage/orders.tbl.16',
'/sybase_stage/orders.tbl.17',
'/sybase_stage/orders.tbl.18',
'/sybase_stage/orders.tbl.19',
'/sybase_stage/orders.tbl.20',
'/sybase_stage/orders.tbl.21',
'/sybase_stage/orders.tbl.22',
'/sybase_stage/orders.tbl.23',
'/sybase_stage/orders.tbl.24',
'/sybase_stage/orders.tbl.25',
'/sybase_stage/orders.tbl.26',
'/sybase_stage/orders.tbl.27',
'/sybase_stage/orders.tbl.28',
'/sybase_stage/orders.tbl.29',
'/sybase_stage/orders.tbl.30',
'/sybase_stage/orders.tbl.31',
'/sybase_stage/orders.tbl.32',
'/sybase_stage/orders.tbl.33',
'/sybase_stage/orders.tbl.34',
'/sybase_stage/orders.tbl.35',
'/sybase_stage/orders.tbl.36',
'/sybase_stage/orders.tbl.37',
'/sybase_stage/orders.tbl.38',
'/sybase_stage/orders.tbl.39',
'/sybase_stage/orders.tbl.40'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
```

load_lineitem.sql

```
LOAD TABLE 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                                 '|'
)
FROM
     '/sybase_stage/lineitem.tbl.1',
     '/sybase_stage/lineitem.tbl.2',
     '/sybase_stage/lineitem.tbl.3',
     '/sybase_stage/lineitem.tbl.4',
     '/sybase_stage/lineitem.tbl.5',
     '/sybase_stage/lineitem.tbl.6',
     '/sybase_stage/lineitem.tbl.7',
     '/sybase_stage/lineitem.tbl.8',
     '/sybase_stage/lineitem.tbl.9',
     '/sybase_stage/lineitem.tbl.10',
     '/sybase_stage/lineitem.tbl.11',
     '/sybase_stage/lineitem.tbl.12',
     '/sybase_stage/lineitem.tbl.13',
     '/sybase_stage/lineitem.tbl.14',
     '/sybase_stage/lineitem.tbl.15',
     '/sybase_stage/lineitem.tbl.16',
     '/sybase_stage/lineitem.tbl.17',
     '/sybase_stage/lineitem.tbl.18',
     '/sybase_stage/lineitem.tbl.19',
     '/sybase_stage/lineitem.tbl.20',
     '/sybase_stage/lineitem.tbl.21',
     '/sybase_stage/lineitem.tbl.22',
     '/sybase_stage/lineitem.tbl.23',
     '/sybase_stage/lineitem.tbl.24',
     '/sybase_stage/lineitem.tbl.25',
     '/sybase_stage/lineitem.tbl.26',
     '/sybase_stage/lineitem.tbl.27',
     '/sybase_stage/lineitem.tbl.28',
     '/sybase_stage/lineitem.tbl.29',
     '/sybase_stage/lineitem.tbl.30',
     '/sybase_stage/lineitem.tbl.31',
     '/sybase_stage/lineitem.tbl.32',
     '/sybase_stage/lineitem.tbl.33',
     '/sybase_stage/lineitem.tbl.34',
     '/sybase_stage/lineitem.tbl.35',
     '/sybase_stage/lineitem.tbl.36',
     '/sybase_stage/lineitem.tbl.37',
     '/sybase_stage/lineitem.tbl.38',
     '/sybase_stage/lineitem.tbl.39',
     '/sybase_stage/lineitem.tbl.40'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
checkpoint;
commit;
```

update_power.sql

```
create variable qstart timestamp;
create variable qstop timestamp;
create variable c_sqlstate CHAR(5);
create variable c_path varchar(128);
set c_path='/sybase_stage/';
set qstart=now(*);
select 'Stream 0 RF1 START -- ', qstart ;
call tpch_rf1 (c_path,'0');
set qstop=now(*);
select 'Stream 0 Update RF1 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds';
select 'Stream 0 RF1 FINISH -- ', qstop ;
-- Sleep Until the query stream completes
set qstart = now(*);
select 'Stream 0 RF WAITING -- ', qstart;
xp_cmdshell('/export/home/sybase/run/scripts/check_que
ry1.bash');
set qstart = now(*);
select 'Stream 0 RF CONTINUING -- ', qstart;
set qstart = now(*);
select 'Stream 0 RF2 START -- ', qstart ;
call tpch_rf2 (c_path,'0');
set qstop=now(*);
select 'Stream 0 Update RF2 LENGTH -- ',
cast(datediff(millisecond,qstart,qstop) as
numeric)/1000, ' seconds';
select 'Stream 0 RF2 FINISH -- ', qstop ;
```

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

update_throughput.sql

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

create variable qstart timestamp;
create variable qstop timestamp;
create variable c_sqlstate CHAR(5);
create variable c_path varchar(128);
set qstart = now(*);
set c_path='/sybase_stage/';
select 'Stream updates START -- ', qstart ;
select @@servername, db_name();
call tpch_rf1 (c_path,'1');
commit;
xp_cmdshell('sleep 3500'); -- sleep 3500 seconds
commit;
call tpch_rf2 (c_path,'1');
commit;
xp_cmdshell('sleep 3500'); -- sleep 3500 seconds
commit;
call tpch_rf1 (c_path,'2');
commit;
xp_cmdshell('sleep 3500'); -- sleep 3500 seconds
commit;
call tpch_rf2 (c_path,'2');
commit;
xp_cmdshell('sleep 3500'); -- sleep 3500 seconds
commit;
call tpch_rf1 (c_path,'3');
commit;
xp_cmdshell('sleep 3500'); -- sleep 3500 seconds
commit;
call tpch_rf2 (c_path,'3');
commit;
xp_cmdshell('sleep 3500'); -- sleep 3500 seconds
commit;
call tpch_rf1 (c_path,'4');
commit;
xp_cmdshell('sleep 3500'); -- sleep 3500 seconds
commit;
call tpch_rf2 (c_path,'4');
commit;
xp_cmdshell('sleep 3500'); -- sleep 3500 seconds
commit;
call tpch_rf1 (c_path,'5');
commit;
xp_cmdshell('sleep 3500'); -- sleep 3500 seconds
commit;
call tpch_rf2 (c_path,'5');
commit;
set qstop = now(*);
select 'Stream updates STOP -- ', qstop ;

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

gen_streams.ksh

```
================================================
#!/bin/ksh
PATH=/export/home/sybase/ASIQ-12_5/bin:
/export/home/sybase/OCS-12_5/bin:/usr/openwin/bin:
/bin:.:/usr/dist/pkgs/forte_dev/SUNWspro/bin:
/usr/ccs/bin:/usr/dt/bin:/usr/dist/pkgs/devpro,
v4.0/5.x-sparc/bin:/usr/dist/local/exe:/usr/dist/exe:
/usr/ucb:/usr/sbin:
/net/josie/export/home18/rgostan/bin:
/export/home/sybase/run/scripts:/etc:.:
/export/home/sybase/run/tpch/appendix/dbgen
export PATH
export DSS_PATH=/export/home/sybase/run/scripts;
export
DSS_CONFIG=/export/home/sybase/run/tpch/appendix/dbgen
;
export DSS_DIST=dists.dss;
export
DSS_QUERY=/export/home/sybase/run/tpch/appendix/templa
tes/queries;
#export
DSS_QUERY=/export/home/sybase/run/tpch/appendix/templa
tes/queries.debug;
let seed=$1;
qgen -c -p 0 -l qparm0.txt -i $DSS_QUERY/init.sql -t
$DSS_QUERY/complete.sql -r $seed -s 100 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 >
stream0.sql
let seed=seed+1;
qgen -c -p 1 -l qparm1.txt -i $DSS_QUERY/init.sql -t
$DSS_QUERY/complete.sql -r $seed -s 100 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 >
stream1.sql
let seed=seed+1;
qgen -c -p 2 -l qparm2.txt -i $DSS_QUERY/init.sql -t
$DSS_QUERY/complete.sql -r $seed -s 100 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 >
stream2.sql
let seed=seed+1;
qgen -c -p 3 -l qparm3.txt -i $DSS_QUERY/init.sql -t
$DSS_QUERY/complete.sql -r $seed -s 100 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 >
stream3.sql
let seed=seed+1;
qgen -c -p 4 -l qparm4.txt -i $DSS_QUERY/init.sql -t
$DSS_QUERY/complete.sql -r $seed -s 100 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 >
stream4.sql
let seed=seed+1;
qgen -c -p 5 -l qparm5.txt -i $DSS_QUERY/init.sql -t
$DSS_QUERY/complete.sql -r $seed -s 100 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 >
stream5.sql
let seed=seed+1;
qgen -c -p 6 -l qparm6.txt -i $DSS_QUERY/init.sql -t
$DSS_QUERY/complete.sql -r $seed -s 100 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 >
stream6.sql
let seed=seed+1;
qgen -c -p 7 -l qparm7.txt -i $DSS_QUERY/init.sql -t
$DSS_QUERY/complete.sql -r $seed -s 100 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 >
stream7.sql
let seed=seed+1;
qgen -c -p 8 -l qparm8.txt -i $DSS_QUERY/init.sql -t
$DSS_QUERY/complete.sql -r $seed -s 100 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 >
stream8.sql
let seed=seed+1;
qgen -c -p 9 -l qparm9.txt -i $DSS_QUERY/init.sql -t
$DSS_QUERY/complete.sql -r $seed -s 100 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 >
stream9.sql
echo $seed

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

ACID Test Execution Code

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

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

atomicity test

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

dbtest acid_atomic_main.tst > acid_atomic_main.out

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

consistency test

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

dbtest acid_consistency_main.tst >
acid_consistency_main.out

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

isolation tests

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

dbtest acid_isolation_main1.tst >
acid_isolation_main1.out
dbtest acid_isolation_main2.tst >
acid_isolation_main2.out
dbtest acid_isolation_main3.tst >
acid_isolation_main3.out
dbtest acid_isolation_main4.tst >
acid_isolation_main4.out
dbtest acid_isolation_main5.tst >
acid_isolation_main5.out
dbtest acid_isolation_main6.tst >
acid_isolation_main6.out

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

durability test

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

dbtest acid_durability_main.tst >
acid_durability_main.out

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

ACID Test Source Code

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

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

acid_atomic_main.tst

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

stringconnect "dsn=tpch;"

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

include 'acid_functions.tst'
commit

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

run test 'acid_atomic_setup.tst'

stringconnect "dsn=tpch;"
let counter=0

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

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

execute {call acid_transaction(^, ^,  ^, rprice,
quantity,
                                tax, disc, extprice,
ototal)
       } substitute ordr, line, delta
close cur2
let counter = counter+1

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

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

} ENDLOOP

commit

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

stringconnect "dsn=tpch;"

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

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

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

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

} ENDLOOP

close cur1
commit

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

End Test
```

==========

acid_atomic_setup.tst

==========

```
Description    "Creates aa_whattodo table"

stringconnect "dsn=tpch;"

% Drop Table if found

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

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

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

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

print 'Number of rows before load: ',ROWS

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

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

print 'Number of rows after load: ',ROWS

disconnect

End Test
```

==========

acid_consistency_main.tst

==========

```
stringconnect "dsn=tpch;"

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

include 'acid_functions.tst'

run test 'acid_consistency_setup.tst'

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

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

start test '-o' 'acid_consist_user1.ot' 'stream=1'
'acid_consistency_txn.tst'
sleep 1000
start test '-o' 'acid_consist_user2.ot' 'stream=2'
'acid_consistency_txn.tst'
sleep 1000
start test '-o' 'acid_consist_user3.ot' 'stream=3'
'acid_consistency_txn.tst'
sleep 1000
start test '-o' 'acid_consist_user4.ot' 'stream=4'
'acid_consistency_txn.tst'
sleep 1000
start test '-o' 'acid_consist_user5.ot' 'stream=5'
'acid_consistency_txn.tst'
sleep 1000
start test '-o' 'acid_consist_user6.ot' 'stream=6'
'acid_consistency_txn.tst'
sleep 1000
start test '-o' 'acid_consist_user7.ot' 'stream=7'
'acid_consistency_txn.tst'
sleep 1000
start test '-o' 'acid_consist_user8.ot' 'stream=8'
'acid_consistency_txn.tst'
sleep 1000
start test '-o' 'acid_consist_user9.ot' 'stream=9'
'acid_consistency_txn.tst'

synchronize 10
% let the log flush...
sleep 10000

stringconnect "dsn=tpch;"
run test '-o' 'acid_consistency_q2.ot'
'acid_consistency_query.tst'

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

```
print ' '

End Test
```

==========

acid_consistency_query.tst

==========

```
stringconnect "dsn=tpch;"

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

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

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

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

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

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

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

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

disconnect

END Test
```

==========

acid_consistency_setup.tst

==========

```
stringconnect "dsn=tpch;"

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

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

execute {checkpoint}

print 'acid_table CREATED!!'

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

LOOP ({let i = 1}; {i <= 9}; { let i = i + 1})
{
  LOOP ({let j = 1}; {j <= 100}; {let j = j + 1})
    {
       execute { call generate_acid_values()} into
ordr, line, delta
       execute { insert into acid_table values (^,^,
^,^,^) }
               substitute i,j,ordr,line,delta
    } endloop
  print (j-1)*i
} endloop

commit

fetch {select count(*) from acid_table } into ROWS
assert ROWS = 900
print 'Number of rows after load: ',ROWS

End Test
```

==========

acid_consistency_txn.tst

==========

```
stringconnect "dsn=tpch;"

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

LOOP ({let i = 1}; {i <= 100}; { let i = i + 1})
{
 fetch {select ordr, line, delta from acid_table
               where stream=^ and seqnum=^ }
        substitute stream, i
 if ROWSTATUS != FOUND then { print 'not enough rows'
               BREAK LOOP }
 endif

 print 'Acid Trenasaction ',i,
       ':  o_key-', ordr , '  l_key-', line , '
delta-' ,delta

 execute {call acid_transaction( ^, ^,  ^, rprice,
quantity,
                               tax, disc, extprice,
ototal)
 } substitute ordr, line, delta
 commit
 print 'committed'
 sleep 1000

}
ENDLOOP

synchronize 10

End Test
```

==========

acid_durability_main.tst

==========

```
stringconnect "dsn=tpch;"

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

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

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

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

start test '-o' 'acid_dura_user1.ot' 'stream=1'
'acid_durability_txn.tst'
sleep 1000
start test '-o' 'acid_dura_user2.ot' 'stream=2'
'acid_durability_txn.tst'
sleep 1000
start test '-o' 'acid_dura_user3.ot' 'stream=3'
'acid_durability_txn.tst'
sleep 1000
start test '-o' 'acid_dura_user4.ot' 'stream=4'
'acid_durability_txn.tst'
sleep 1000
start test '-o' 'acid_dura_user5.ot' 'stream=5'
'acid_durability_txn.tst'
sleep 1000
start test '-o' 'acid_dura_user6.ot' 'stream=6'
'acid_durability_txn.tst'
sleep 1000
start test '-o' 'acid_dura_user7.ot' 'stream=7'
'acid_durability_txn.tst'
sleep 1000
start test '-o' 'acid_dura_user8.ot' 'stream=8'
'acid_durability_txn.tst'
sleep 1000
start test '-o' 'acid_dura_user9.ot' 'stream=9'
'acid_durability_txn.tst'

synchronize 10

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


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

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

End Test
```

==========

acid_durability_query.tst

==========

```
stringconnect "dsn=tpch;"

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

let n=1

LOOP {
   fetch cur1 into str, seq, ord, lin, delta

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

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

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

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

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

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

 } ENDLOOP

disconnect

END Test
```

==========

acid_durability_setup.tst

==========

```
stringconnect "dsn=tpch;"

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

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

execute {checkpoint}

print 'acid_table CREATED!!'

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

LOOP ({let i = 1}; {i <= 9}; { let i = i + 1})
{
  LOOP ({let j = 1}; {j <= 200}; { let j = j + 1})
    {
       execute { call generate_acid_values()} into
ordr, line, delta
       execute { insert into acid_table values (^,^,
^,^,^) }
```

```
              substitute i,j,ordr,line,delta
    } endloop
  print (j-1)*i
} endloop

commit
execute {checkpoint}

fetch {select count(*) from acid_table } into ROWS

print 'Number of rows after load: ',ROWS

End Test
```

==========

acid_durability_txn.tst

==========

```
stringconnect "dsn=tpch;"

execute {select now(*)} into times
print 'Durability test start = ', times
print ' '
print 'stream  trans. o_key   l_key   p_key   s_key
delta   date_t '

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

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

 execute {select l_partkey, l_suppkey from lineitem
               where l_orderkey=^ and l_linenumber=^}
        substitute ordr, line
        into p_key, s_key

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

 assert SQLCODE=0 then { DIE } endassert
 commit

 execute {select now(*)} into times
 print  stream,'      ',
        'txn ',i, '    ',
        ordr,  '       ',
        line,  '       ',
        p_key, '       ',
        s_key, '       ',
        delta, '       ',
        times, '       '

 sleep 1000
}
ENDLOOP

synchronize 10

End Test
```

==========

acid_functions.tst

==========

```
print 'creating the sleep procedure'

allow error -265
execute { DROP PROCEDURE dbo.sleep}
allow no error

execute{ create procedure dbo.sleep(in sleep_time
integer default null)
begin
  declare command varchar(255);
  select 'xp_cmdshell ''sleep '+str(sleep_time)+''''
into command;
  execute immediate command
end;
}

print 'creating the Acid Transaction'

allow error -265
execute { DROP PROCEDURE acid_transaction }
allow no error

execute{ CREATE PROCEDURE acid_transaction(
                IN  o_key      INT,
                IN  l_key      INT,
                IN  delta      INT,
                OUT rprice     Numeric(18,8),
                OUT quantity   INT,
                OUT tax        Numeric(18,8),
                OUT disc       Numeric(18,8),
                OUT extprice   Numeric(18,8),
                OUT ototal     Numeric(18,8)
                )
ON EXCEPTION RESUME
BEGIN
 DECLARE pkey           INT ;
 DECLARE skey           INT ;
 DECLARE cost           NUMERIC(18,8) ;
 DECLARE new_extprice   NUMERIC(18,8) ;
 DECLARE new_ototal     NUMERIC(18,8) ;
 DECLARE new_quantity   INT ;
 DECLARE c_sqlstate     char(5);
 LOOP1: LOOP
    COMMIT;
    SELECT o_totalprice
      INTO ototal
      FROM orders
     WHERE o_orderkey = o_key ;
    SELECT l_quantity,
           l_extendedprice,
           l_partkey,
           l_suppkey,
           l_tax,
           l_discount
      INTO quantity,
           extprice,
           pkey,
           skey,
           tax,
           disc
      FROM lineitem
     WHERE l_orderkey   = o_key
       AND l_linenumber = l_key;
    -- CLEAN UP IMPRECICE NUMBERS
    SET ototal = ototal -
"TRUNCATE"("truncate"(extprice*(1-disc),2)*(1+tax),2);
    SET rprice = "TRUNCATE"((extprice  / quantity),2);
    SET cost = "TRUNCATE"((rprice * delta),2);
    SET new_extprice = extprice + cost;
    SET new_ototal = "TRUNCATE"(new_extprice * (1.0 -
disc),2);
    SET new_ototal = "TRUNCATE"(new_ototal * (1.0 +
```

```
tax),2);
    SET new_ototal = ototal + new_ototal ;
    SET new_quantity = quantity + delta ;
    --
    -- Update LineItem
    --
    UPDATE lineitem
       SET l_quantity      = new_quantity,
           l_extendedprice = new_extprice
     WHERE l_orderkey=o_key
       AND l_linenumber=l_key;
    SELECT SQLSTATE INTO c_sqlstate;
    IF c_sqlstate = '00000' THEN
        --
        -- Update Orders
        --
        UPDATE orders
           SET o_totalprice = new_ototal
         WHERE o_orderkey = o_key;
        SELECT SQLSTATE INTO c_sqlstate;
        IF c_sqlstate = '00000'  THEN
            INSERT INTO history VALUES ( pkey, skey,
o_key, l_key, delta, now(*)) ;
            SELECT SQLSTATE INTO c_sqlstate;
            IF c_sqlstate = '00000'  THEN
                LEAVE LOOP1;
            END IF;
        END IF;
    END IF;
 END LOOP LOOP1;
 RETURN(0);
END;
}

print 'Acid transaction created'
print ' '

print 'Creating Acid query'

allow error -265
execute { DROP PROCEDURE acid_single_query }
allow no error

execute{
CREATE PROCEDURE acid_single_query( IN o_key INT, OUT
o_total NUMERIC(26,16) )
BEGIN
    SELECT o_total =
        sum ("truncate" ("truncate"(
                         round(cast(l_extendedprice as
numeric(26,16)),2) *
                         (1 - round(cast(l_discount as
numeric(26,16)),2)),2)
                    * (1 + round(cast(l_tax as
numeric(26,16)),2)) , 2))
    FROM lineitem WHERE l_orderkey = o_key;
 END
}

print 'Acid query created'
print ' '

print 'Creating Generate_acid_values function'

allow error -265
execute { DROP PROCEDURE generate_acid_values }
allow no error

execute{
create procedure generate_acid_values(
out orderkey   int,
out linenumber int,
out delta      int)

BEGIN

 declare seed         bigint;
 declare rand_dbl     double precision;
 declare rand_int     int;
 declare out_key      int;

 declare times cursor for select datediff(millisecond,
convert(char(10),getdate(), 116),now(*));
 declare random1 cursor for select rand(seed);
 declare random cursor for select rand();
 declare get_order cursor for
        select o_orderkey from orders where o_orderkey
= rand_int;
 declare get_linenumber cursor for
        select max(l_linenumber) from lineitem
        where l_orderkey = orderkey;

 open times;
 fetch next times into seed;
 open random1;
 fetch next random1 into rand_dbl;

 set out_key = 0;
 loop1:
 while out_key = 0 LOOP
        open random;
        open get_order;

        fetch next random into rand_dbl;
        set rand_int = rand_dbl * 6001215 +1;
        fetch next get_order into out_key;

        close random;
        close get_order;
 end loop loop1;

 set orderkey = out_key;

 open get_linenumber;
 fetch next get_linenumber into linenumber;
 close get_linenumber;

 open random;
 fetch next random into rand_dbl;
 set delta = rand_dbl * 100 + 1;
 close random;

END
}
commit
execute {checkpoint}
print 'Generate_acid_values function created'
print ' '

print 'Creating Generate_Ps_Values function'

allow error -265
execute { DROP PROCEDURE generate_ps_values }
allow no error

execute{
create procedure generate_ps_values(
out partkey    int,
out suppkey    int)

BEGIN

 declare seed         bigint;
 declare rand_dbl     double precision;
```

```
 declare rand_int     int;
 declare out_key      int;
 declare counter      int;

 declare times cursor for select datediff(millisecond,
convert(char(10),getdate(), 116),now(*));
 declare random1 cursor for select rand(seed);
 declare random cursor for select rand();
 declare get_supp cursor for
        select ps_suppkey from partsupp
        where ps_suppkey = rand_int;
 declare get_part cursor for
        select ps_partkey from partsupp
        where ps_suppkey = suppkey;

 open times;
 fetch next times into seed;
 open random1;
 fetch next random1 into rand_dbl;
 close random1;

 set out_key = 0;
 while out_key = 0 LOOP
        open random;
        open get_supp ;

        fetch next random into rand_dbl;
        set rand_int = rand_dbl * 10000 +1;
        fetch next get_supp into out_key;

        close random;
        close get_supp ;
 end loop;
 set suppkey = out_key;

 set out_key = 0;
 set counter  = 0;
 open random;
 open get_part;
 fetch next random into rand_dbl;
 set rand_int = rand_dbl * 10 +1;

 loop1:
 while counter < rand_int LOOP
        set counter = counter+1;
        fetch next get_part into out_key;
 end loop loop1;

 set partkey = out_key;
 close random;
 close get_part;

END
}
commit
execute {checkpoint}
print 'Generate_Ps_Values function created'
print ' '

print 'Creating history table'

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

execute {
create table history (
   h_p_key    unsigned INT NOT NULL ,
   h_s_key    unsigned INT NOT NULL ,
   h_o_key    unsigned INT NOT NULL ,
   h_l_key    INT NOT NULL,
   h_delta    INT NOT NULL,
   h_date_t   TIMESTAMP NOT NULL)
}

commit
execute {checkpoint}
print 'history table created'
print ' '
```

==========

acid_isolation_main1.tst

==========

```
stringconnect "dsn=tpch;"

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

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

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

End Test
```

==========

acid_isolation_main2.tst

==========

```
stringconnect "dsn=tpch;"

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


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

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

End Test
```

==========

acid_isolation_main3.tst

==========

```
stringconnect "dsn=tpch;"

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

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

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

```
End Test
```

==========

acid_isolation_main4.tst

==========

```
stringconnect "dsn=tpch;"

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

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

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

End Test
```

==========

acid_isolation_main5.tst

==========

```
stringconnect "dsn=tpch;"

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

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

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

End Test
```

==========

acid_isolation_main6.tst

==========

```
stringconnect "dsn=tpch;"

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

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

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

End Test
```

==========

acid_isolation_setup.tst

==========

```
stringconnect "dsn=tpch;"

% Drop Table if found

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

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

execute {checkpoint}

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

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

print 'Number of rows before load: ',ROWS

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

commit

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

print 'Number of rows after load: ',ROWS

disconnect

End Test
```

==========

acid_isolation_test1.tst

==========

```
stringconnect "dsn=tpch;"

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

 print 'The following are the data input values for
the ACID Transaction.'
 print '(user 1) o_key-',ordr, '  l_key-', line, '
delta-',delta

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

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

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

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

End Test
```

==========

acid_isolation_test1_query.tst

==========

```
stringconnect "dsn=tpch;"

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

 execute {select ordr from acid_isolation_table}
        into ordr

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

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

disconnect

END Test
```

==========

acid_isolation_test2.tst

==========

```
stringconnect "dsn=tpch;"

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

 print 'The following are the data input values for
the ACID Transaction.'
 print '(user 1) o_key-',ordr, '  l_key-', line, '
delta-',delta

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

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

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

End Test
```

==========

acid_isolation_test2_query.tst

==========

```
stringconnect "dsn=tpch;"

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

 execute {select ordr from acid_isolation_table}
        into ordr

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

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

disconnect

END Test
```

==========

acid_isolation_test3_transaction1.tst

==========

```
stringconnect "dsn=tpch;"

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

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

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

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

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

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

```
 print '30 second timer started'
SYNCHRONIZE 2
 sleep 30000

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

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

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

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

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

 disconnect
 End Test
```

==========

acid_isolation_test3_transaction2.tst

==========

```
stringconnect "dsn=tpch;"

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

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

SYNCHRONIZE 2
sleep 5000

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

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

sleep 5000
print ' '
execute {select now(*)} into times
print 'User 2 -- About to commit: ', times
commit

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

print ' '

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

End Test
```

==========

acid_isolation_test4_transaction1.tst

==========

```
stringconnect "dsn=tpch;"

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

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

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

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

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

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

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

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

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

```
 print 'USER 1 -- original quantity = ', quantity
 print ' '

 disconnect
 End Test
```

acid_isolation_test4_transaction2.tst

```
stringconnect "dsn=tpch;"

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

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

SYNCHRONIZE 2
sleep 5000

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

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

sleep 5000
print ' '
execute {select now(*)} into times
print 'User 2 -- About to commit: ', times
commit

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

End Test
```

acid_isolation_test5_query.tst

```
stringconnect "dsn=tpch;"

synchronize 2

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

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

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

print ' '

End Test
```

acid_isolation_test5_transaction1.tst

```
stringconnect "dsn=tpch;"

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

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

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

execute {call acid_transaction( ^, ^,  ^, rprice,
```

```
quantity, tax, disc,
       extprice, ototal) } substitute ordr, line,
delta

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

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

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

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

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

End Test
```

==========

acid_isolation_test6_query.tst

==========

```
stringconnect "dsn=tpch;"

print 'User1 Query: '
print ' '
print 'User1 starts its query (Q1) here.'
execute {select now(*)} into qstart
print 'Start time for User1 Q1 =', qstart
print ' '
compare fetchall {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 <= dateadd(day, -1, '1998-12-01')
group by l_returnflag,l_linestatus
order by l_returnflag,l_linestatus
 } in 'queryresult'

execute {select now(*)} into qstop
print 'Stop time for User1 Q1 =', qstop
print ' '

End Test
```

==========

acid_isolation_test6_transaction1.tst

==========

```
stringconnect "dsn=tpch;"

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

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

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

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

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

```
       into l_price, l_quant
print 'User 2 o_totalprice = ', o_tprice
print 'User 2 l_extendedprice = ', l_price,'
l_quantity = ', l_quant
print 'User 2 l_partkey       = ', l_ptky,'
l_suppkey = ', l_spky
print ' '

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

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

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

End Test
```

==========

Disk Configuration Details

==========

Solaris Volume Manager Configuration

```
# Create the state database:
metadb -a -f c2t8d0s1
metadb -a -f c2t9d0s1

# Initialize 3310 Devices:

metainit d5 1 1 c2t8d0s4
metainit d6 1 1 c2t9d0s4
metainit d7 1 1 c2t10d0s4
metainit d8 1 1 c2t11d0s4
metainit d9 1 1 c2t12d0s4

metainit d10 1 1 c3t8d0s4
metainit d11 1 1 c3t9d0s4
metainit d12 1 1 c3t10d0s4
metainit d13 1 1 c3t11d0s4
metainit d14 1 1 c3t12d0s4

# Create the mirrors:

metainit d15 -m d5
metattach d15 d10
metainit d16 -m d6
metattach d16 d11
metainit d17 -m d7
metattach d17 d12
metainit d18 -m d8
metattach d18 d13
metainit d19 -m d9
metattach d19 d14

# Create a RAID 1 volume for the /sybase2 file system

metainit d2 1 1 c2t10d0s1
metainit d3 1 1 c2t11d0s1
metainit d4 -m d2
metattach d4 d3
```

Database Device Links

```
# Create the symbolic links:

ln -s /dev/md/rdsk/d15 M01
ln -s /dev/md/rdsk/d16 M02
ln -s /dev/md/rdsk/d17 M03
ln -s /dev/md/rdsk/d18 M04
ln -s /dev/md/rdsk/d19 M05
ln -s /dev/rdsk/c2t8d0s5 T01
ln -s /dev/rdsk/c2t9d0s5 T02
ln -s /dev/rdsk/c2t10d0s5 T03
ln -s /dev/rdsk/c2t11d0s5 T04
ln -s /dev/rdsk/c2t12d0s5 T05
ln -s /dev/rdsk/c3t8d0s5 T06
ln -s /dev/rdsk/c3t9d0s5 T07
ln -s /dev/rdsk/c3t10d0s5 T08
ln -s /dev/rdsk/c3t11d0s5 T09
ln -s /dev/rdsk/c3t12d0s5 T10
```

==========

entries from /etc/vfstab

==========

```
# Root File System

/dev/dsk/c0t0d0s0       /dev/rdsk/c0t0d0s0      /
ufs     1       no      -

# File Systems for Load Files Generated by dbgen
/dev/md/dsk/d1          /dev/md/rdsk/d1
/sybase_stage   ufs     1       yes     largefiles

# RAID 1 File System for .db .log and db device links

/dev/md/dsk/d4          /dev/md/rdsk/d4
/sybase2        ufs     1       yes     -

# Swap Space Configuration

swap    -       /tmp    tmpfs   -       yes     -
/dev/dsk/c0t0d0s1       -       -       swap    -
no      -
```

Appendix C. Query Text and Query Output

```
==================================================
qualification query 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 <= dateadd(day,-90,'1998-12-01')
% group by
% l_returnflag,
% l_linestatus
% order by
% l_returnflag,
% l_linestatus;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.32000 seconds - current
time 16:40:13
'A','F',
37734107,56586554400.7292032,53758257134.8694563,55909
065222.8284717,25.5220058532573342,38273.1297346211374
,.0499852958383577168,1478493
'N','F',
991417,1487504710.38000107,1413082168.05409968,1469649
223.19436967,25.5164719205229819,38284.4677608483374,
.0500934266742134809,38854
'N','O',
74476040,111701729697.737336,106118230307.607383,11036
7043872.495174,25.5022267695849895,38249.117988907361,
.049996586053555131,2920374
'R','F',
37719753,56568041380.8983326,53741292684.6045375,55889
619119.8339581,25.5057936126907617,38250.8546260985255
,.0500094058300870121,1478870
% total of 4 rows written

==================================================
qualification query 2
==================================================

% select top 100
% s_acctbal,
% s_name,
% n_name,
% p_partkey,
% p_mfgr,
% s_address,
% s_phone,
% s_comment
% from
% part,
% supplier,
% partsupp,
% nation,
% region
% where
% p_partkey = ps_partkey
% and s_suppkey = ps_suppkey
% and p_size = 15
% and p_type like 'BRASS'
% and s_nationkey = n_nationkey
% and n_regionkey = r_regionkey
% and r_name = 'EUROPE'
% and ps_supplycost = (
% select
% min(ps_supplycost)
% from
% partsupp,
% supplier,
% nation,
% region
% where
% p_partkey = ps_partkey
% and s_suppkey = ps_suppkey
% and s_nationkey = n_nationkey
% and n_regionkey = r_regionkey
% and r_name = 'EUROPE'
% )
% order by
% s_acctbal desc,
% n_name,
% s_name,
% p_partkey;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.76000 seconds - current
time 16:40:25
9938.53,'Supplier#000005359       ','UNITED KINGDOM
',185358,'Manufacturer#4           ','QKuHYh,
vZGiwu2FWEJoLDx04','33-429-790-6131','blithely silent
pinto beans are furiously. slyly final deposits acros'
9937.84,'Supplier#000005969       ','ROMANIA
',108438,'Manufacturer#1           ','ANDENSOSmk,
miq23Xfb5RWt6dvUcvt6Qa','29-520-692-3537','carefully
slow deposits use furiously. slyly ironic platelets
above the ironic'
9936.22,'Supplier#000005250       ','UNITED KINGDOM
',249,'Manufacturer#4           ','B3rqp0xbSEim4Mpy2RH
J','33-320-228-2957','blithely special packages are.
stealthily express deposits across the closely final
instructi'
9923.77000000000119,'Supplier#000002324       ',
'GERMANY                  ',29821,'Manufacturer#4
','y3OD9UywSTOk','17-779-299-1839','quickly express
packages breach quiet pinto beans. requ'
9871.22,'Supplier#000006373       ','GERMANY
',43868,'Manufacturer#5           ','J8fcXWsTqM',
'17-813-485-8637','never silent deposits integrate
furiously blit'
9870.78,'Supplier#000001286       ','GERMANY
',81285,'Manufacturer#2           ','YKA,
E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH','17-516-924-4574',
'final theodolites cajole slyly special,'
9870.78,'Supplier#000001286       ','GERMANY
',181285,'Manufacturer#4           ','YKA,
E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH','17-516-924-4574',
'final theodolites cajole slyly special,'
9852.52000000000119,'Supplier#000008973       ',
'RUSSIA                   ',18972,'Manufacturer#2
','t5L67YdBYYH6o,Vz24jpDyQ9','32-188-594-7038',
'quickly regular instructions wake-- carefully unusual
```

```
braids into the expres'
9847.83,'Supplier#000008097       ','RUSSIA
',130557,'Manufacturer#2           ',
'xMe97bpE69NzdwLoX','32-375-640-3593','slyly regular
dependencies sleep slyly furiously express dep'
9847.57,'Supplier#000006345       ','FRANCE
',86344,'Manufacturer#1           ',
'VSt3rzk3qG698u6ld8HhOByvrTcSTSvQlDQDag',
'16-886-766-7945','silent pinto beans should have to
snooze carefully along the final reques'
% total of 100 rows written
```

qualification query 3

```
% select top 10
% 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 < '1995-03-15'
% and l_shipdate > '1995-03-15'
% group by
% l_orderkey,
% o_orderdate,
% o_shippriority
% order by
% revenue desc,
% o_orderdate;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.36000 seconds - current
time 16:40:27
2456423,406181.011100000024,'1995-03-05',0
3459808,405838.698899999917,'1995-03-04',0
492164,390324.061,'1995-02-19',0
1188320,384537.935899999976,'1995-03-09',0
2435712,378673.055799999952,'1995-02-26',0
4878020,378376.795200000048,'1995-03-12',0
5521732,375153.9215,'1995-03-13',0
2628192,373133.309399999976,'1995-02-22',0
993600,371407.45949999994,'1995-03-05',0
2300070,367371.145200000107,'1995-03-13',0
% total of 10 rows written
```

qualification query 4

```
% select
% o_orderpriority,
% count(*) as order_count
% from
% orders
% where
% o_orderdate >= '1993-07-01'
% and o_orderdate < dateadd(month,3,'1993-07-01')
% and exists (
% select
% *
% from
% lineitem
% where
% l_orderkey = o_orderkey
% and l_commitdate < l_receiptdate
% )
% group by
% o_orderpriority
% order by
% o_orderpriority;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.25000 seconds - current
time 16:40:31
'1-URGENT       ',10594
'2-HIGH         ',10476
'3-MEDIUM       ',10410
'4-NOT SPECIFIED',10556
'5-LOW          ',10487
% total of 5 rows written
```

qualification query 5

```
% 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 >= '1994-01-01'
% and o_orderdate < dateadd(year,1,'1994-01-01')
% group by
% n_name
% order by
% revenue desc;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.65000 seconds - current
time 16:40:36
'INDONESIA                ',55502041.1696999431
'VIETNAM                  ',55295086.9966999531
'CHINA                    ',53724494.2565999746
'INDIA                    ',52035512.000200057
'JAPAN                    ',45410175.6954000235
% total of 5 rows written
```

qualification query 6

```
% select
% sum(l_extendedprice * l_discount) as revenue
% from
% lineitem
% where
% l_shipdate >= '1994-01-01'
% and l_shipdate < dateadd(year,1,'1994-01-01')
% and l_discount between .06 - 0.01 and .06 + 0.01
% and l_quantity < 24;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.15000 seconds - current
time 16:40:41
123141078.228299007
% total of 1 rows written
```

==========

qualification query 7

==========

```
% 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 '1995-01-01' and '1996-12-31'
% ) as shipping
% group by
% supp_nation,
% cust_nation,
% l_year
% order by
% supp_nation,
% cust_nation,
% l_year;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.67000 seconds - current
time 16:40:43
'FRANCE                    ','GERMANY
',1995,54639732.7335999489
'FRANCE                    ','GERMANY
',1996,54633083.3075999737
'GERMANY                   ','FRANCE
',1995,52531746.6696999669
'GERMANY                   ','FRANCE
',1996,52520549.0223998487
% total of 4 rows written
```

==========

qualification query 8

==========

```
% 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 '1995-01-01' and
'1996-12-31'
% and p_type = 'ECONOMY ANODIZED STEEL'
% ) as all_nations
% group by
% o_year
% order by
% o_year;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.86000 seconds - current
time 16:40:47
1995,.0344358904066548347
1996,.041485521293530345
% total of 2 rows written
```

==========

qualification query 9

==========

```
% 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;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.69000 seconds - current
time 16:40:48
'ALGERIA                  ',1998,31342867.2345000029
'ALGERIA                  ',1997,57138193.0233001232
'ALGERIA                  ',1996,56140140.1330001235
'ALGERIA                  ',1995,53051469.6533999741
'ALGERIA                  ',1994,53867582.128600049
'ALGERIA                  ',1993,54942718.132400012
'ALGERIA                  ',1992,54628034.7126999021
'ARGENTINA                ',1998,30211185.708099997
'ARGENTINA                ',1997,50805741.75230003
'ARGENTINA                ',1996,51923746.5754999459
% total of 175 rows written
```

==========

qualification query 10

==========

```
% select top 20
% 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 >= '1993-10-01'
% and o_orderdate < dateadd(month,3,'1993-10-01')
% 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;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.50000 seconds - current
time 16:40:55
57040,'Customer#000057040',734235.2455,632.87,'JAPAN
','Eioyzjf4pp','22-895-641-3466','requests sleep
blithely about the furiously i'
143347,'Customer#000143347',
721002.694799999952,2557.4700000000003,'EGYPT
','1aReFYv,Kw4','14-742-935-3718','fluffily bold
excuses haggle finally after the u'
60838,'Customer#000060838',
679127.307700000048,2454.77,'BRAZIL
','64EaJ5vMAHWJlBOxJklpNc2RJiWE','12-913-494-9813',
'furiously even pinto beans integrate under the
ruthless foxes; ironic, even dolphins across the slyl'
101998,'Customer#000101998',
637029.566699999809,3790.89,'UNITED KINGDOM
','01c9CILnNtfOQYmZj','33-593-865-6378','accounts doze
blithely! enticing, final deposits sleep blithely
special accounts. slyly express accounts pla'
125341,'Customer#000125341',
633508.086,4983.51000000000006,'GERMANY
','S29ODD6bceU8QSuuEJznkNaK','17-582-695-5962',
'quickly express requests wake quickly blithely'
25501,'Customer#000025501',
620269.784899999976,7725.04,'ETHIOPIA
','  W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ',
'15-874-808-6793','quickly special requests sleep
evenly among the special deposits. special deposi'
115831,'Customer#000115831',
596423.867200000167,5098.1,'FRANCE
','rFeBbEEyk dl ne7zV5fDrmiq1oK09wV7pxqCgIc',
'16-715-386-3788','carefully bold excuses sleep
alongside of the thinly idle'
84223,'Customer#000084223',594998.023899999976,528.65,
'UNITED KINGDOM           ','nAVZCs6BaWap rrM27N
2qBnzc5WBauxbA','33-442-824-8191','pending, final
ideas haggle final requests. unusual, regular
asymptotes affix according to the even foxes.'
54289,'Customer#000054289',
585603.391799999952,5583.02,'IRAN
','vXCxoCsU0Bad5JQI ,oobkZ','20-834-292-4707','express
requests sublate blithely regular requests. regular,
even ideas solve.'
39922,'Customer#000039922',
584878.113399999976,7321.10999999999881,'GERMANY
','Zgy4s50l2GKN4pLDPBU8m342gIw6R','17-147-757-8036',
'even pinto beans haggle. slyly bold accounts inte'
6226,'Customer#000006226',576783.760599999905,2230.09,
'UNITED KINGDOM           ','8gPu8,
NPGkfyQQ0hcIYUGPIBWc,ybP5g,','33-657-701-3391',
'quickly final requests against the regular
instructions wake blithely final instructions. pa'
922,'Customer#000000922',576767.533299999833,3869.25,
'GERMANY                  ',
'Az9RFaut7NkPnc5zSD2PwHgVwr4jRzq','17-945-916-9648',
'boldly final requests cajole blith'
```

```
147946,'Customer#000147946',
576455.132,2030.1300000000003,'ALGERIA
','iANyZHjqhyy7Ajah0pTrYyhJ','10-886-956-3143',
'furiously even accounts are blithely above the
furiousl'
115640,'Customer#000115640',
569341.193299999952,6436.1,'ARGENTINA
','Vtgfia9qI 7EpHgecU1X','11-411-543-4901','final
instructions are slyly according to the'
73606,'Customer#000073606',
568656.857799999952,1785.67,'JAPAN
','xuR0Tro5yChDfOCrjkd2ol','22-437-653-6966',
'furiously bold orbits about the furiously busy
requests wake across the furiously quiet theodolites.
d'
110246,'Customer#000110246',
566842.981499999881,7763.35,'VIETNAM
','7KzflgX MDOq7sOkI','31-943-426-9837','dolphins
sleep blithely among the slyly final'
142549,'Customer#000142549',
563537.236799999952,5085.9899999999994,'INDONESIA
','ChqEoK43OysjdHbtKCp6dKqjNyvvi9','19-955-562-2398',
'regular, unusual dependencies boost slyly; ironic
attainments nag fluffily into the unusual packages?'
146149,'Customer#000146149',557254.9865,1791.55,
'ROMANIA                ','s87fvzFQpU',
'29-744-164-6487','silent, unusual requests detect
quickly slyly regul'
52528,'Customer#000052528',556397.350899999976,551.79,
'ARGENTINA              ','NFztyTOR10UOJ',
'11-208-192-3205','unusual requests detect. slyly
dogged theodolites use slyly. deposit'
23431,'Customer#000023431',
554269.536000000119,3381.86,'ROMANIA
','HgiV0phqhaIa9aydNoIlb','29-915-458-2654',
'instructions nag quickly. furiously bold accounts
cajol'
% total of 20 rows written
```

==========

qualification query 11

==========

```
% 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;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.49000 seconds - current
time 16:41:01
129760,17538456.8599999994
166726,16503353.9199999988
191287,16474801.9699999988
161758,16101755.5399999976
34452,15983844.7200000018
139035,15907078.3400000006
9403,15451755.6199999988
154358,15212937.8799999982
38823,15064802.8599999994
85606,15053957.150000003
% total of 1048 rows written
```

==========

qualification query 12

==========

```
% 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 >= '1994-01-01'
% and l_receiptdate < dateadd(year,1,'1994-01-01')
% group by
% l_shipmode
% order by
% l_shipmode;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.26000 seconds - current
time 16:41:03
'MAIL      ',6202,9324
'SHIP      ',6200,9262
% total of 2 rows written
```

==========

qualification query 13

==========

```
% 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 'specialrequests'
% group by
% c_custkey
% ) as c_orders (c_custkey, c_count)
% group by
% c_count
% order by
% custdist desc,
% c_count desc;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.17000 seconds - current
time 16:41:06
0,50004
9,6641
10,6566
11,6058
8,5949
12,5553
13,4989
19,4748
7,4707
18,4625
% total of 42 rows written
```

==========

qualification query 14

==========

```
% select
% 100.00 * sum(case
% when p_type like 'PROMO'
% then l_extendedprice * (1 - l_discount)
% else 0
% end) / sum(l_extendedprice * (1 - l_discount)) as
promo_revenue
% from
% lineitem,
% part
% where
% l_partkey = p_partkey
% and l_shipdate >= '1995-09-01'
% and l_shipdate < dateadd(month,1,'1995-09-01');
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.24000 seconds - current
time 16:41:19
16.3807786263955563
% total of 1 rows written
```

==========

qualification query 15

==========

```
 Executing command:
% create view revenue0 (supplier_no, total_revenue) as
% select
% l_suppkey,
% sum(l_extendedprice * (1 - l_discount))
% from
% lineitem
% where
% l_shipdate >= '1996-01-01'
% and l_shipdate < dateadd(month,3,'1996-01-01')
% group by
% l_suppkey;
% execution time 0.81000 seconds - current time 16:41:
21

 Executing command:
%
% select
% s_suppkey,
% s_name,
% s_address,
% s_phone,
% total_revenue
% from
% supplier,
% revenue0
% where
% s_suppkey = supplier_no
% and total_revenue = (
% select
% max(total_revenue)
% from
% revenue0
% )
% order by
% s_suppkey;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.27000 seconds - current
time 16:41:21
8449,'Supplier#000008449       ','Wp34zim9qYFbVctdW',
'20-469-856-8873',1772627.20870000005
% total of 1 rows written
```

==========

qualification query 16

==========

```
% select
% p_brand,
% p_type,
% p_size,
% count(distinct ps_suppkey) as supplier_cnt
% from
% partsupp,
% part
% where
% p_partkey = ps_partkey
% and p_brand <> 'Brand#45'
% and p_type not like 'MEDIUM POLISHED'
% and p_size in (49, 14, 23, 45, 19, 3, 36, 9)
% and ps_suppkey not in (
% select
% s_suppkey
% from
% supplier
% where
% s_comment like 'CustomerComplaints'
% )
% group by
% p_brand,
% p_type,
% p_size
% order by
```

```
% supplier_cnt desc,
% p_brand,
% p_type,
% p_size;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.27000 seconds - current
time 16:41:22
'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

% total of 18314 rows written
```

==========

qualification query 17

==========

```
% 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
% );
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.22000 seconds - current
time 16:41:28
348406.054285713732
% total of 1 rows written
```

==========

qualification query 18

==========

```
% select top 100
% 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;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.34000 seconds - current
time 16:41:29
'Customer#000128120',128120,4722021,'1994-04-07',
544089.089999999881,323
'Customer#000144617',144617,3043270,'1997-02-12',
530604.43999999994,317
'Customer#000013940',13940,2232932,'1997-04-13',
522720.61,304
'Customer#000066790',66790,2199712,'1996-09-30',
515531.82,327
'Customer#000046435',46435,4745607,'1997-07-03',
508047.99,309
'Customer#000015272',15272,3883783,'1993-07-28',
500241.33,302
'Customer#000146608',146608,3342468,'1994-06-12',
499794.58,303
'Customer#000096103',96103,5984582,'1992-03-16',
494398.78999999994,312
'Customer#000024341',24341,1474818,'1992-11-15',
491348.26,302
'Customer#000137446',137446,5489475,'1997-05-23',
487763.25,311
% total of 57 rows written
```

==========

qualification query 19

==========

```
% 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'
% );
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.35000 seconds - current
time 16:41:46
3083843.05780000031
% total of 1 rows written
```

==========

qualification query 20

==========

```
% 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 >= '1994-01-01'
% and l_shipdate < dateadd(year,1,'1994-01-01')
% )
% )
% and s_nationkey = n_nationkey
% and n_name = 'CANADA'
% order by
% s_name;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.37000 seconds - current
time 16:41:51
'Supplier#000000020       ','iybAE,RmTymrZVYaFZva2SH,
j'
'Supplier#000000091       ',
'YV45D7TkfdQanOOZ7q9QxkyGUapU1oOWU6q3'
'Supplier#000000197       ',
'YC2Acon6kjY3zj3Fbxs2k4Vdf7X0cd2F'
'Supplier#000000226       ','83qOdU2EYRdPQAQhEtn
GRZEd'
'Supplier#000000285       ',
'Br7e1nnt1yxrw6ImgpJ7YdhFDjuBf'
'Supplier#000000378       ','FfbhyCxWvcPrO8ltp9'
'Supplier#000000402       ','i9Sw4DoyMhzhKXCH9By,
AYSgmD'
'Supplier#000000530       ','0qwCMwobKY
OcmLyfRXlagA8ukENJv,'
'Supplier#000000688       ','D
fw5ocppmZpYBBIPI718hCihLDZ5KhKX'
'Supplier#000000710       ','f19YPvOyb QoYwjKC,
oPycpGfieBAcwKJo'
% total of 204 rows written
```

==========

qualification query 21

==========

```
% select top 100
% 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;
```

```
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.54000 seconds - current
time 16:41:53
'Supplier#000002829       ',20
'Supplier#000005808       ',18
'Supplier#000000262       ',17
'Supplier#000000496       ',17
'Supplier#000002160       ',17
'Supplier#000002301       ',17
'Supplier#000002540       ',17
'Supplier#000003063       ',17
'Supplier#000005178       ',17
'Supplier#000008331       ',17
% total of 100 rows written
```

==========

qualification query 22

==========

```
% select
% cntrycode,
% count(*) as numcust,
% sum(c_acctbal) as totacctbal
% from
% (
% select
% substring(c_phone,1,2) as cntrycode,
% c_acctbal
% from
% customer
% where
% substring(c_phone,1,2) 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,1,2) in
% ('13', '31', '23', '29', '30', '18', '17')
% )
% and not exists (
% select
% *
% from
% orders
% where
% o_custkey = c_custkey
% )
% ) as custsale
% group by
% cntrycode
% order by
% cntrycode;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.18000 seconds - current
time 16:42:07
'13',888,6737713.98999999881
'17',861,6460573.72
'18',964,7236687.40000001431
'23',892,6701457.95000000954
'29',948,7158866.62999999642
'30',909,6808436.13000000119
'31',922,6806670.17999998569
% total of 7 rows written
```

Appendix D. Seed and Query Substitution Parameters

This Appendix contains Seed values and substitution parameters for each stream

Seed Values

```
stream0 531171959
stream1 531171960
stream2 531171961
stream3 531171962
stream4 531171963
stream5 531171964
```

Query Parameters

```
====================================================
stream0: 531171959
====================================================

1       97
2       12      TIN     AFRICA
3       AUTOMOBILE      1995-03-06
4       1993-12-01
5       MIDDLE EAST     1997-01-01
6       1997-01-01      0.06    25
7       JORDAN  MOROCCO
8       MOROCCO AFRICA  LARGE POLISHED TIN
9       navajo
10      1994-08-01
11      RUSSIA  0.0000010000
12      FOB     MAIL    1995-01-01
13      unusual requests
14      1995-03-01
15      1994-11-01
16      Brand#14        SMALL BRUSHED   3       46
7       10      32      44      134
17      Brand#42        LG BOX
18      313
19      Brand#25        Brand#11        Brand#45
1       12      22
20      steel   1993-01-01      EGYPT
21      ALGERIA
22      14      13      31      25      10      15
       18

====================================================
stream1: 531171960
====================================================

1       105
2       49      COPPER  EUROPE
3       FURNITURE       1995-03-23
4       1996-07-01
5       AFRICA  1993-01-01
6       1993-01-01      0.03    24
7       ETHIOPIA        GERMANY
8       GERMANY EUROPE  LARGE BURNISHED TIN
9       medium
10      1993-05-01
11      IRAN    0.0000010000
12      MAIL    TRUCK   1997-01-01
13      unusual requests
14      1995-06-01
15      1997-06-01
16      Brand#44        ECONOMY BURNISHED       16
17      8       22      29      3615      14
17      Brand#44        LG PACK
18      314
19      Brand#22        Brand#44        Brand#34
6       13      29
20      frosted 1996-01-01      ROMANIA
21      PERU
22      29      14      27      11      16      34
       28

====================================================
stream2: 531171961
====================================================

1       113
2       37      STEEL   AMERICA
3       MACHINERY       1995-03-08
4       1994-04-01
5       AMERICA 1993-01-01
6       1993-01-01      0.09    25
7       RUSSIA  UNITED STATES
8       UNITED STATES   AMERICA MEDIUM BRUSHED TIN
9       lemon
10      1994-03-01
11      UNITED KINGDOM  0.0000010000
12      RAIL    FOB     1995-01-01
13      unusual accounts
14      1995-09-01
15      1995-03-01
16      Brand#34        STANDARD PLATED 20      39
1       19      30      29      2531
17      Brand#41        LG DRUM
18      312
19      Brand#24        Brand#22        Brand#33
1       14      25
20      purple  1994-01-01      INDONESIA
21      INDONESIA
22      19      22      11      15      31      21
       24

====================================================
stream3: 531171962
====================================================

1       60
2       25      NICKEL  EUROPE
3       FURNITURE       1995-03-25
4       1996-11-01
5       ASIA    1993-01-01
6       1993-01-01      0.06    25
7       KENYA   MOZAMBIQUE
8       MOZAMBIQUE      AFRICA  MEDIUM PLATED TIN
9       indian
10      1994-12-01
11      IRAQ    0.0000010000
12      AIR     FOB     1995-01-01
13      express accounts
14      1995-12-01
15      1997-10-01
16      Brand#14        MEDIUM BRUSHED  23      26
6       39      22      12      321
17      Brand#43        MED BOX
18      313
19      Brand#31        Brand#15        Brand#33
7       15      22
20      chiffon 1993-01-01      UNITED KINGDOM
21      ARGENTINA
```

```
22      33      14      11      25      19      21
       24

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

stream4: 531171963

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

1       68
2       13      TIN     AMERICA
3       MACHINERY       1995-03-10
4       1994-08-01
5       MIDDLE EAST     1993-01-01
6       1993-01-01      0.04    24
7       FRANCE  INDIA
8       INDIA   ASIA    MEDIUM ANODIZED TIN
9       ghost
10      1993-09-01
11      UNITED STATES   0.0000010000
12      REG AIR FOB     1996-01-01
13      express accounts
14      1996-03-01
15      1995-06-01
16      Brand#54        PROMO ANODIZED  24      29
38      28      18      10      324
17      Brand#45        MED PACK
18      315
19      Brand#33        Brand#43        Brand#22
2       16      29
20      misty   1996-01-01      JORDAN
21      CHINA
22      23      18      11      19      10      14
       29

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

stream5: 531171964

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

1       76
2       50      COPPER  MIDDLE EAST
3       BUILDING        1995-03-27
4       1997-03-01
5       AFRICA  1993-01-01
6       1993-01-01      0.09    24
7       UNITED KINGDOM  ALGERIA
8       ALGERIA AFRICA  SMALL POLISHED NICKEL
9       drab
10      1994-06-01
11      JAPAN   0.0000010000
12      SHIP    FOB     1996-01-01
13      express accounts
14      1996-07-01
15      1993-03-01
16      Brand#34        SMALL PLATED    30      40
29      2       15      14      3433
17      Brand#42        MED DRUM
18      312
19      Brand#35        Brand#31        Brand#21
7       17      25
20      yellow  1995-01-01      CANADA
21      IRAQ
22      13      28      18      26      11      32
       29
```

Appendix E. Implementation-Specific Layer/Driver Code

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

do_test

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

#!/bin/bash
echo Startup IQ: `date`
start_asiq @utility.cfg
echo IQ started: `date`
echo "  "
echo  Creating Database: `date`
echo "  "
dbisqlc -c "DSN=utility_db" -q create_database.sql
echo  Database Created: `date`
echo "  "

echo Shutting down IQ: `date`
dbstop -c "DSN=utility_db" -y
echo IQ shutdown: `date`
echo "  "

echo Sleeping  5 Seconds
echo "  "
sleep 5


echo Restart IQ with TPCH database: `date`
start_asiq @tpch.cfg /sybase2/tpch.db
echo IQ restarted: `date`
echo "  "
echo Adding dbspaces
dbisqlc -c "DSN=tpch" -q create_dbspaces.sql
echo Set Database Options `date`

dbisqlc -c "DSN=tpch" -q options.sql
dbisqlc -c "DSN=tpch" -q tpch_rf.sql

echo "  "

echo Shutting down IQ: `date`
dbstop -c "DSN=tpch" -y
echo Restart IQ with TPCH database: `date`
start_asiq @tpch.cfg /sybase2/tpch.db
echo IQ restarted: `date`
echo "  "
echo Create Tables `date`

dbisqlc -c "DSN=tpch" -q create_tables_noRI.sql

echo "  "
echo Dump the IQ Configuration `date`

dbisqlc -c "DSN=tpch" -q check_options.sql
#
# Load the database
#
echo "  "
echo Starting Load  `now_iq_format.bash` | tee
start_load.out
dbisqlc -c "DSN=tpch" -q load_lineitem.sql >
load_lineitem.out &
loadlpid=$!
echo "   Lineitem is load started `date` "
sleep 16800
dbisqlc -c "DSN=tpch" -q load_region.sql
echo "   Region loaded `date` "
dbisqlc -c "DSN=tpch" -q load_nation.sql
echo "   Nation loaded `date` "
dbisqlc -c "DSN=tpch" -q load_customer.sql
echo "   Customer loaded `date` "
dbisqlc -c "DSN=tpch" -q load_part.sql
echo "   Part loaded `date` "
dbisqlc -c "DSN=tpch" -q load_supplier.sql
echo "   Supplier loaded `date` "
dbisqlc -c "DSN=tpch" -q load_partsupp.sql
echo "   Partsupp loaded `date` "
dbisqlc -c "DSN=tpch" -q load_orders.sql
loadopid=$!
echo "   Orders loaded `date` "
wait $loadopid
wait $loadlpid
seed=`date '+%m%d%H%M%S'`;
#
# Generate the Query Streams using the seed based on
database load completion
#
echo $seed;
./gen_streams.ksh $seed
echo "   Lineitem loaded `date` "
echo Load Finished  `now_iq_format.bash` | tee
end_load.out
echo "  "
echo "Run the Audit Script   `date` "
echo "  "
#
# After the load Completes run the Audit SQL
#
dbisqlc -c "DSN=tpch" -q dbtables-syb.sql >
rdbtablest.out
dbisqlc -c "DSN=tpch" -q dew_cat1.sql >
dew_cat1_start.out
dbisqlc -c "DSN=tpch" -q dew_cat2.sql >
dew_cat2_start.out
dbisqlc -c "DSN=tpch" -q dew_cat3.sql >
dew_cat3_start.out
echo "  "
echo "Start the Power Test   `date` "
echo "  "
#
# Touch the lock files so that the RF will pause after
RF1 and wait for the query stream to complete
#
touch /export/home/sybase/run/scripts/rf1.lock
touch /export/home/sybase/run/scripts/rf2.lock
#
# Start the RF Stream in the Background
#
dbisqlc -c "DSN=tpch" -q update_power.sql >
update_power.out &
rfspid=$!
#
# Wait for RF1 to Complete before Starting the Query
Stream
#
# the rf1.lock file will be removed after RF1
completes
# RF will then wait for rf2.lock to be removed before
continuing
#
while [ -f /export/home/sybase/run/scripts/rf1.lock ]
do
 # Sleep while Trigger file exists
 # echo "Waiting until trigger file is removed"
 sleep 10
done
#
# Continue RF1 has completed and RF2 is waiting on the
Query Stream to Complete
#
```

```
echo "  "
echo "Start Query Stream 0 `date` "
echo "  "
#
# Start the Query Stream
#
dbisqlc -c "DSN=tpch" -q stream0.sql > stream0.out
#
# Remove the lock file so that the RF will continue
with RF2
#
rm -f /export/home/sybase/run/scripts/rf2.lock
#
# Now wait for the RF stream 0 to complete
#
wait $rfspid

echo End Power `date`
echo "  "
echo Start Throughput `date`
echo "  "
echo Run Query Streams
dbisqlc -c "DSN=tpch" -q stream1.sql > stream1.out &
qs1pid=$!
sleep 4
dbisqlc -c "DSN=tpch" -q stream2.sql > stream2.out &
qs2pid=$!
sleep 4
dbisqlc -c "DSN=tpch" -q stream3.sql > stream3.out &
qs3pid=$!
sleep 4
dbisqlc -c "DSN=tpch" -q stream4.sql > stream4.out &
qs4pid=$!
sleep 4
dbisqlc -c "DSN=tpch" -q stream5.sql > stream5.out &
qs5pid=$!
sleep 4
echo Run refresh
echo "  "
dbisqlc -c "DSN=tpch" -q update_throughput.sql >
update_throughput.out &
rf0pid=$!
echo "  "
wait $rf0pid
wait $qs1pid
wait $qs2pid
wait $qs3pid
wait $qs4pid
wait $qs5pid

echo "Start the Power Test   `date` "
echo "  "
mv stream0.out m1s00q.out
mv update_power.out m1s00rf.out
mv stream1.out m1s01q.out
mv stream2.out m1s02q.out
mv stream3.out m1s03q.out
mv stream4.out m1s04q.out
mv stream5.out m1s05q.out
mv update_throughput.out m1s01rf.out
# exit
#
# Touch the lock files so that the RF will pause after
RF1 and wait for the query stream to complete
#
touch /export/home/sybase/run/scripts/rf1.lock
touch /export/home/sybase/run/scripts/rf2.lock
#
# Start the RF Stream in the Background
#
dbisqlc -c "DSN=tpch" -q update_power.sql >
update_power.out &
rfspid=$!
#
# Wait for RF1 to Complete before Starting the Query
Stream
#
# the rf1.lock file will be removed after RF1
completes
# RF will then wait for rf2.lock to be removed before
continuing
#
while [ -f /export/home/sybase/run/scripts/rf1.lock ]
do
 # Sleep while Trigger file exists
 # echo "Waiting until trigger file is removed"
 sleep 10
done
#
# Continue RF1 has completed and RF2 is waiting on the
Query Stream to Complete
#
echo "  "
echo "Start Query Stream 0 `date` "
echo "  "
#
# Start the Query Stream
#
dbisqlc -c "DSN=tpch" -q stream0.sql > stream0.out
#
# Remove the lock file so that the RF will continue
with RF2
#
rm -f /export/home/sybase/run/scripts/rf2.lock
#
# Now wait for the RF stream 0 to complete
#
wait $rfspid

echo End Power `date`
echo "  "
echo Start Throughput `date`
echo "  "
echo Run Query Streams
dbisqlc -c "DSN=tpch" -q stream1.sql > stream1.out &
qs1pid=$!
sleep 4
dbisqlc -c "DSN=tpch" -q stream2.sql > stream2.out &
qs2pid=$!
sleep 4
dbisqlc -c "DSN=tpch" -q stream3.sql > stream3.out &
qs3pid=$!
sleep 4
dbisqlc -c "DSN=tpch" -q stream4.sql > stream4.out &
qs4pid=$!
sleep 4
dbisqlc -c "DSN=tpch" -q stream5.sql > stream5.out &
qs5pid=$!
sleep 4
echo Run refresh
echo "  "
dbisqlc -c "DSN=tpch" -q update_throughput.sql >
update_throughput.out &
rf0pid=$!
echo "  "
wait $rf0pid
wait $qs1pid
wait $qs2pid
wait $qs3pid
wait $qs4pid
wait $qs5pid

mv stream0.out m2s00q.out
mv update_power.out m2s00rf.out
```

```
mv stream1.out m2s01q.out
mv stream2.out m2s02q.out
mv stream3.out m2s03q.out
mv stream4.out m2s04q.out
mv stream5.out m2s05q.out
mv update_throughput.out m2s01rf.out
dbisqlc -c "DSN=tpch" -q dew_cat1.sql >
dew_cat1_end.out
dbisqlc -c "DSN=tpch" -q dew_cat2.sql >
dew_cat2_end.out
dbisqlc -c "DSN=tpch" -q dew_cat3.sql >
dew_cat3_end.out
```

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-syb.sql

==========

```
-- ================================================
--  FILENAME
--       DBTABLES.SQL
--  DESCRIPTION
--       CHECK ROW COUNT AND ROW STRUCTURE/CONTENT FOR
EACH TABLE
--       IN THE TPC-H DATABASE.
--
-- ================================================
--
-- GET TIMESTAMP
SELECT 'START TIME', CONVERT(CHAR(30), GETDATE(),
120);
go
-- ================================================
--           TABLE: LINEITEM
-- ================================================
SELECT COUNT(*) FROM LINEITEM;
go
SELECT * FROM LINEITEM
WHERE L_ORDERKEY IN
      ( 4, 26598,  148577,  387431, 56704, 517442,
600000)
      AND L_LINENUMBER = 1
ORDER BY L_ORDERKEY;
go
-- ================================================
--           TABLE: ORDERS
-- ================================================
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM ORDERS;
go
SELECT * FROM ORDERS
WHERE O_ORDERKEY IN ( 7, 44065, 287590, 411111,
483876, 599942 )
ORDER BY O_ORDERKEY;
go
-- ================================================
--           TABLE: PART
-- ================================================
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM PART;
go
SELECT * FROM PART
WHERE P_PARTKEY IN (1,984,8743,9028,13876,17899,20000)
ORDER BY P_PARTKEY;
go
-- ================================================
--           TABLE: PARTSUPP
-- ================================================
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM PARTSUPP;
go
SELECT* FROM PARTSUPP
   WHERE PS_PARTKEY = 3398
   AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
      FROM PARTSUPP WHERE PS_PARTKEY = 3398);
go
SELECT* FROM PARTSUPP
   WHERE PS_PARTKEY =15873
   AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
      FROM PARTSUPP WHERE PS_PARTKEY = 15873);
go
SELECT* FROM PARTSUPP
   WHERE PS_PARTKEY = 11394
   AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
      FROM PARTSUPP WHERE PS_PARTKEY = 11394);
go
SELECT* FROM PARTSUPP
   WHERE PS_PARTKEY = 6743
   AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
      FROM PARTSUPP WHERE PS_PARTKEY = 6743);
go
SELECT* FROM PARTSUPP
   WHERE PS_PARTKEY = 19763
   AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
      FROM PARTSUPP WHERE PS_PARTKEY =19763);
go
-- ================================================
--           TABLE: SUPPLIER
-- ================================================
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM SUPPLIER;
go
SELECT * FROM SUPPLIER
WHERE  S_SUPPKEY IN (83,265,492,784,901,1000)
ORDER BY S_SUPPKEY;
go
-- ================================================
--           TABLE: CUSTOMER
-- ================================================
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM CUSTOMER;
go
SELECT * FROM CUSTOMER
WHERE  C_CUSTKEY IN (832,2653,4924,7845,92016,108070)
ORDER BY C_CUSTKEY;
go
-- ================================================
--           TABLE: NATION & REGION
-- ================================================
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT * FROM REGION;
go
SELECT COUNT(*) FROM NATION;
go
SELECT * FROM NATION
WHERE N_NATIONKEY IN (3,10,14,20)
ORDER BY N_NATIONKEY;
go
-- ================================================
--           CHECK KEY VALUES
-- ================================================
-- GET TIMESTAMP
```

```
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
if exists (select name from sysobjects where
name='MINMAX')
  drop table MINMAX
go
CREATE TABLE MINMAX
(TNAME CHAR(15),
KEYMIN INTEGER,
KEYMAX INTEGER);
go
INSERT INTO MINMAX
SELECT 'LINEITEM_ORD',MIN(L_ORDERKEY),MAX(L_ORDERKEY)
FROM LINEITEM;
go
INSERT INTO MINMAX
SELECT 'LINEITEM_NBR',MIN(L_LINENUMBER),
MAX(L_LINENUMBER)
FROM LINEITEM;
go
INSERT INTO MINMAX
SELECT 'ORDERS',MIN(O_ORDERKEY),MAX(O_ORDERKEY)
FROM ORDERS;
go
INSERT INTO MINMAX
SELECT 'CUSTOMER',MIN(C_CUSTKEY),MAX(C_CUSTKEY)
FROM CUSTOMER;
go
INSERT INTO MINMAX
SELECT 'PART',MIN(P_PARTKEY),MAX(P_PARTKEY)
FROM PART;
go
INSERT INTO MINMAX
SELECT 'SUPPLIER',MIN(S_SUPPKEY),MAX(S_SUPPKEY)
FROM SUPPLIER;
go
INSERT INTO MINMAX
SELECT 'PARTSUPP_PART',MIN(PS_PARTKEY),MAX(PS_PARTKEY)
FROM PARTSUPP;
go
INSERT INTO MINMAX
SELECT 'PARTSUPP_SUPP',MIN(PS_SUPPKEY),MAX(PS_SUPPKEY)
FROM PARTSUPP;
go
INSERT INTO MINMAX
SELECT 'NATION',MIN(N_NATIONKEY),MAX(N_NATIONKEY)
FROM NATION;
go
INSERT INTO MINMAX
SELECT 'REGION',MIN(R_REGIONKEY),MAX(R_REGIONKEY)
FROM REGION;
go
SELECT * FROM MINMAX;
go
if exists (select name from sysobjects where
name='MINMAX')
  drop table MINMAX
go
SELECT 'END TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
```

==========

dew_cat1.sql

==========

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

==========

dew_cat2.sql

==========

```
 select T.table_name    ,
        T.table_type    ,
        C.column_name   ,
        C.column_id
 From   SYS.SYSTABLE T,
        SYS.SYSCOLUMN C,
        SYS.SYSDOMAIN D,
        SYS.SYSUSERPERMS SU
 where T.creator = SU.user_id
   and T.table_id = C.table_id
   and C.domain_id = D.domain_id
order by 1,2;
```

==========

dew_cat3.sql

==========

```
SELECT  index_name,T.table_name  ,
        column_name   ,
        index_type
 from   SYS.SYSTABLE T,
        SYS.SYSCOLUMN C,
        SYS.SYSINDEX I,
        SYS.SYSUSERPERMS UP,
        SYS.SYSFILE F,
        SYS.SYSIXCOL IC
 where T.table_id = C.table_id
   and C.table_id = I.table_id
   and T.file_id = F.file_id
   and I.table_id = IC.table_id
   AND I.index_id = IC.index_id
   AND IC.column_id = C.column_id
   and T.creator = UP.user_id;
```

Appendix G. Pricing information

For Sybase pricing please contact:

Deborah Harrington
978-287-1535
dharring@sybase.com

For Sun pricing please contact:

Guido Ficco
781-442-0069
guido.ficco@sun.com

Company
Contact
Phone
Fax
Address

CBSS#

Quotation for Software and Support
2 cpu Server

SYBASE Sales Rep: Hollie Nash
Phone: 972-687-6412
Fax: 972-687-6409

	Catalogue Number	Product Description	License Type	Machine	P/S	List Price Per Unit	Quantity	Discount % ** Standard	Net Unit Price	Net Extended Price	Net Extended Support Fees
1	12841	Sybase IQ-M Single App Svr - 2 cpu's	SR	Sun	P	$ 15,000	1	20%	$12,000	$12,000	$7,920
2	98480	Extended support/24 x 7					1				
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											
						Sub-total				$ 12,000.00	$ 7,920.00
						Total			$ 19,920.00	License + 3 Year maintenance	

** - Excludes royalty based products

Quote Date:
Valid thru:

Payment terms : Net 30 Days

5400 LBJ Freeway, Suite 1500, Dallas, TX 75240

Date: June 23, 2003

To: Jay Halloran
Sun Microsystems

Ph: 781-442-2635

From:
Nick Defino/Dave Mancusi
Continental Resources, Inc.
175 Middlesex Turnpike
Bedford, MA 01730
Tel: 781-533-0307/0454

Qty.	Part Number	Description	List $	Your Unit $	Extended $
		SUN Microsystems			
1	XTA3310R01A0R436	SE3310-436GB-12x36-JBOD	$11,995.00	$8,565.00	$8,565.00
1	X6758A	dual-ultra-3-SCSI-HBA	$800.00	$571.00	$571.00
1	X1138A	2M-SCSI-cable	$95.00	$82.00	$82.00
				Total:	$9,218.00

Orders subject to credit approval.
Payment terms net 30 days.
Prices valid for 60 days, availability subject to change.

INTEGRITY
NETWORKING SYSTEMS, INC.

1001 Medical Arts Ave. NE
Albuquerque, NM 87102
(505) 294-7747 OFFICE
(505) 275-1125 FAX

Quotation valid for 60 days

DATE: June 23, 2003
QUOTE #: **1304** Please place quote number on any resulting orders.
TO:
ATTN:
PHONE:
FAX:

In accordance with your request we are pleased to submit the following quotation

ITEM NO.	QTY	PART NUMBER	DESCRIPTION	UNIT PRICE	EXT. UNIT PRICE
1	1	N32-XUB2-9S-204AV2	Sun Fire V240, 2 x 1GHz UltraSPARC IIIi, 4 x 512 MB DIMMS, 2 x 36GB Drives, 4 x 10/100/1000 Gigibit Ethernet, ALOM Remote Manager, Three PCI Slots, n+1 Redundent Power.	$5,781.00	$ 5,781.00
2	2	X311L	Localized Power Cord Kit	$0.00	$ -
3	1	W9D-N32-3G	Sunfire V240 Upgrade to 3 years of Gold Support	$5,012.00	$ 5,012.00
4	4	X7404A	2GB Memory Expansion Kit (2X1GB low-profile DDR PC2100 registered ECC DIMMs) For Use in Sunfire V210 and V240	$1,243.00	$ 4,972.00
5	1	ALW-20-WS-2XMEM	Sun Upgrade Allowance code	$0.00	$ -
6	1	UG-RMA	Sun Upgrade RMA Kit	$0.00	$ -
7	1	SOLZS-090C9AYS	Solaris 9 (latest release) Slim kit. Contains Multilingual CD & DVD Media with minimal documentation, no bonus software, SPARC Platform Edition.	$36.00	$ 36.00
				SUBTOTAL	$ 15,801.00
				FREIGHT	
				TOTAL	$ 15,801.00

All orders are subject to Integrity Networking Systems, Inc. standard terms and conditions. Open product will not be returned.

Delivery: 30 Days ARO
FOB: Destination
Terms: Net 30 Days
Tax $ -

Kevin Wichers
Integrity Networking Systems, Inc. Authorization