Sun microsystems

THE NETWORK IS THE COMPUTER™

TPC Benchmark™ H
Full Disclosure Report

Sun Microsystems Sun Fire™ V490 Server
Using Sybase IQ 12.5

Submitted for Review
Report Date: feb 15, 2005

TPC Benchmark H Full Disclosure Report

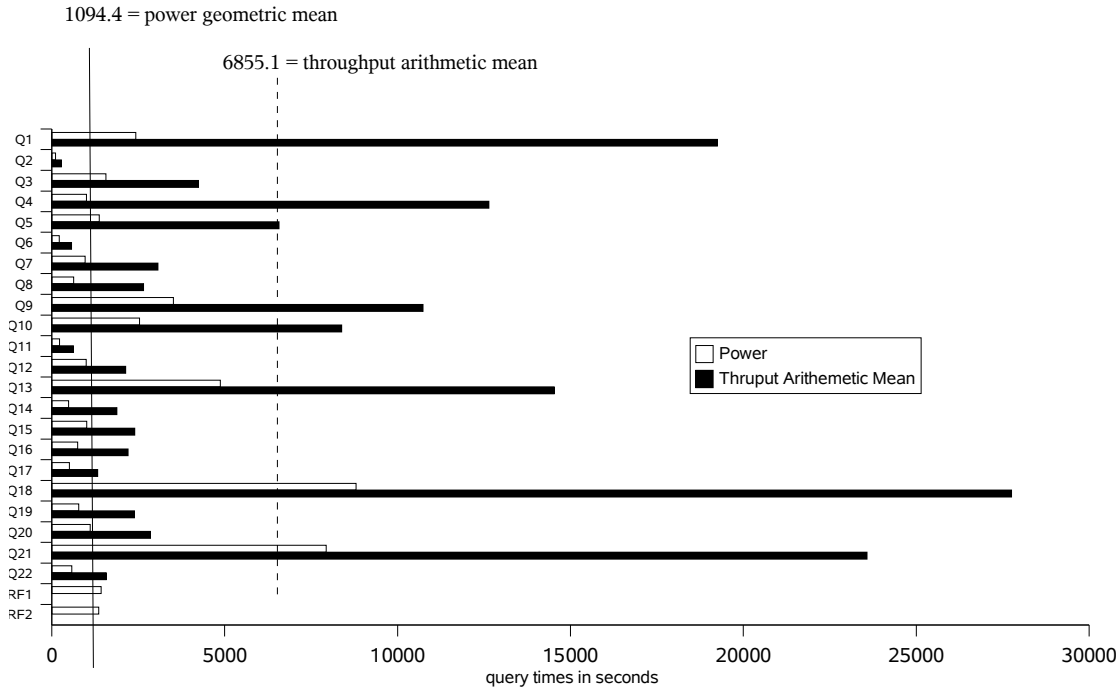
First Printing

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

Sun microsystems / SYBASE	Sun Fire™ V490 Server with Sybase IQ Multiplex 12.5	TPC-H Rev. 2.0
		Report Date: feb 15, 2005

Total System Cost	Composite Query per Hour Metric	Price/Performance
$142,918.15	**3446.2** QphH@1000GB	**$41** per QphH@1000GB

Database Size	Database Manager	Operating System	Other Software	Availability Date
1000GB	**Sybase IQ 12.5**	**Solaris 10**	**Solaris Volume Manager**	**mar 31, 2005**



Database Load Time = 19 hrs 36 mins Load Includes Backup: N Total Storage/Database Size=2.58

RAID (Base tables): RAID 1 RAID (Base tables and auxiliary data structures): RAID 1 RAID (All): N

System Configuration:

SunFire V490 Server with
- 4 UltraSPARC IV 1350 MHz processors (2 cores per processor)
- 32 GB memory
- 2 x 73GB (10K RPM) internal disks
- 3 x SE3310 SCSI JBOD disk array, each with 12 x 73GB (10K RPM) disks

Total Storage: 2583 GB

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

Sun Fire™ V490 Server with Sybase IQ Multiplex 12.5

TPC-H Rev. 2.0

Report Date: feb 15, 2005

Description	Part Number	Source	Unit Price	Qty	Ext. Price	3 Yr. maint.
Server Hardware						
SunFire V490 4x1.05GHz CPUs, 32G memory, 2x73G disk drives		1	75,995.00	1	75,995.00	
3 Year Gold Warranty Upgrade – V880 Server (includes HBAs and Internal disks)		1	10,332.00	1		10,332.00
Sun Discount		1			-12,919.15	-2,066.40
Server HardwareSubtotal					63,075.85	8,265.60
Storage						
SE3310 JBOD Disk Array(12X73GB 10,000 RPM disks)	XTA3310R01A0R876	2	11,995.00	3	35,985.00	
3 Year Gold Warranty Upgrade – SE3310 (includes all disks)	W9D-SE3310-3G	2	4,356.00	3		13,068.00
Continental Discount		2			-9,672.00	-915.00
SCSI cable	x1138A	1	95.00	6	570.00	
Ultra320 SCSI HBA	XG-XPCI2SCSILM320	1	510.00	3	1,530.00	
Sun Discount		1			-544.50	
Storage Subtotal					27,868.50	12,153.00
Server Software						
Sybase IQ-M Single App Svr, per cpu core	12841	3	2,595.00	8	20,760.00	
Sybase IQ 3 Years Extended Support 24 x 7	98480	3	1,557.00	8		12,456.00
Sybase Discount		3			-1,660.80	
Server Software Subtotal					19,099.20	12,456.00
			Total		110,043.55	32,874.60
			3 Yr. Cost		142,918.15	
			QphH@ 1000GB		3,446.20	
			$/QphH@ 100GB		$41	

Service for all Sun products is from Sun Microsystems, Inc.

Service for Sybase products is from Sybase Inc.

Notes (Source):

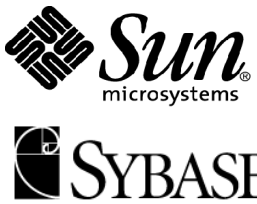
1. Sun Microsystems Inc.

2.Continential Resources Inc

3. Sybase Inc.

PriceQuotes provided in Appendix G

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



Sun Fire™ V490 Server with Sybase IQ Multiplex 12.5

TPC-H Rev. 2.0

Report Date: feb 15, 2005

Numerical Quantities

Measurement Results:

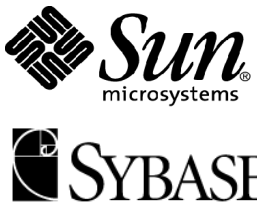
Database Scale Factor	= 1000GB
Total Data Storage / Database Size	=2.58
Start of database load time	= 2005-01-01 21:51:32
End of database load time	= 2005-01-02 17:27:20
Database Load Time	= 19 hrs 36 mins
Query Streams for Throughput Test	= 7
TPC-H Power	= 3289.4
TPC-H Throughput	= 3610.5
TPC-H Composite Query-per-Hour Rating (QphH@1000GB)	= 3446.2
Total System Price Over 3 Years	= $142,918.15
TPC-H Price/Performance Metric ($/QphH@1000GB)	= $41

Measurement Intervals:

Measurement Interval in Throughput Test (Ts)	= 153,551 seconds

Duration of Stream Execution:

Stream ID	Seed	Start Date	Start Time	End Date	End Time	Duration
Stream 00	102172720	Jan 2, 2005	17:32:24	Jan 3, 2005	6:05:32	12:33:08
Stream 01	102172721	Jan 3, 2005	6:05:33	Jan 4, 2005	23:34:03	41:28:30
Stream 02	102172722	Jan 3, 2005	6:05:33	Jan 4, 2005	23:45:32	41:39:59
Stream 03	102172723	Jan 3, 2005	6:05:33	Jan 4, 2005	23:39:47	41:34:14
Stream 04	102172724	Jan 3, 2005	6:05:33	Jan 5, 2005	00:04:51	41:59:18
Stream 05	102172725	Jan 3, 2005	6:05:33	Jan 4, 2005	23:53:55	41:48:22
Stream 06	102172726	Jan 3, 2005	6:05:33	Jan 5, 2005	00:44:44	42:39:11
Stream 07	102172727	Jan 3, 2005	6:05:33	Jan 5, 2005	00:11:07	42:05:34
Refresh		Jan 3, 2005	6:05:33	Jan 4, 2005	21:40:38	39:35:05



Sun Fire™ V490 Server with Sybase IQ Multiplex 12.5

TPC-H Rev. 2.0

Report Date: feb 15, 2005

TPC-H Timing Intervals (in seconds)

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Stream 00	2429.60	104.17	1568.78	1000.86	1372.29	214.14	965.27	632.39	3514.73	2539.43	222.34	992.18
Stream 01	20803.05	307.93	4159.60	13903.95	4466.48	437.90	2182.85	2028.69	10864.18	6496.75	449.81	2246.83
Stream 02	18174.70	287.13	6218.98	12156.15	6155.37	1077.55	2991.59	2731.89	6809.16	6856.41	1122.77	2245.16
Stream 03	19615.85	180.47	3469.60	12881.34	6806.39	431.82	1411.75	4719.27	9222.49	6832.63	598.53	1689.04
Stream 04	17135.72	170.26	3520.67	12070.05	9320.52	425.33	4798.05	1338.10	8486.34	7097.84	521.97	2449.13
Stream 05	16948.20	200.98	4843.46	17194.48	6363.97	543.92	3719.29	2230.36	10552.67	7459.38	425.83	2503.26
Stream 06	25973.15	309.80	4407.98	8560.91	7761.04	493.80	3480.21	2746.66	15249.30	14484.56	754.98	2763.67
Stream 07	16095.62	503.24	3099.80	11695.50	5135.14	613.02	2874.02	2767.20	13970.04	9460.32	504.52	1064.51
Minimum	16095.62	170.26	3099.80	8560.91	4466.48	425.33	1411.75	1338.10	6809.16	6496.75	425.83	1064.51
Average	19249.47	279.97	4245.73	12637.48	6572.70	574.76	3065.39	2651.74	10736.31	8383.98	625.49	2137.37
Maximum	25973.15	503.24	6218.98	17194.48	9320.52	1077.55	4798.05	4719.27	15249.30	14484.56	1122.77	2763.67

	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	RF1	Rf2
Stream 00	4876.63	489.29	1007.82	748.92	507.66	8797.69	782.09	1108.66	7938.21	580.66	1425.72	1355.02
Stream 01	12382.39	1108.93	2444.21	1719.78	1266.18	29164.76	1696	2145.36	27865.01	1166.92	4192.38	3443.28
Stream 02	12162.14	4264.56	2578.66	3224.73	2751.35	25874.7	2939.28	2139.87	25252.29	1980.67	2577.65	2842.64
Stream 03	17395.62	786.01	3464.07	2865.46	771.57	25053.58	2930.18	2192.42	23966.58	2365.12	2687.73	2278.79
Stream 04	17685.5	3068.74	2562.64	1680.16	1004.8	29319.95	3268.25	2071.81	21795.85	1360.89	2429.81	2291.51
Stream 05	13707.83	1137.35	2317.29	1833.73	889.26	25002.96	1451.56	2559.52	27656.3	958.18	2184.56	2602.83
Stream 06	10144.61	2188.07	1809.56	2140.17	916.02	25797.25	1747.56	2871.32	16745.11	2201.96	2074.6	2681.55
Stream 07	15942.07	1170.01	1290.13	1719.36	1244.84	33480.85	1561.72	4570.36	21601.09	1169.04	2621.99	2819.55
Minimum	10144.61	786.01	1290.13	1680.16	771.57	25002.96	1451.56	2071.81	16745.11	958.18	2074.60	2278.79
Average	14202.88	1960.52	2352.37	2169.06	1263.43	27670.58	2227.79	2650.09	23554.60	1600.40	2681.25	2708.59
Maximum	17685.50	4264.56	3464.07	3224.73	2751.35	33480.85	3268.25	4570.36	27865.01	2365.12	4192.38	3443.28

Table of Contents

INFO SIZING

Benchmark Sponsors:

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

Paul Kreneta
Chief Technologist, Sybase IQ
Sybase Inc.
One Sybase Drive
Dublin, CA 94568

February 7, 2005

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

Platform: **Sun Fire V490 Server**

Database Manager: **Sybase IQ 12.5**

Operating System: **Solaris 10**

The results were:

CPU (Speed)	Memory	Disks	**QphH@1000GB**
One (1) Sun Fire 490			
4 x UltraSPARC IV (1.35 Ghz) (2 cores/processor)	32 GB Main	38 x 73 GB	**3446.2**

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

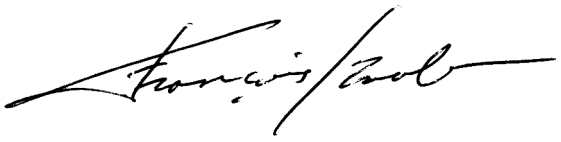
- The database records were defined with the proper layout and size
- The database population was generated using DBGEN
- The database was properly scaled to 1000GB and populated accordingly
- The compliance of the database auxiliary data structures was verified
- The database load time was correctly measured and reported
- The required ACID properties were verified and met

- The query input variables were generated by QGEN
- The query text was produced using minor modifications
- The execution of the queries against the SF1 database produced compliant answers
- A compliant implementation specific layer was used to drive the tests
- The throughput tests involved 7 query streams
- The ratio between the longest and the shortest query was such that no query timing was adjusted
- The execution times for queries and refresh functions were correctly measured and reported
- The repeatability of the measured results was verified
- The required amount of database log was configured
- The system pricing was verified for major components and maintenance
- The major pages from the FDR were verified for accuracy

Additional Audit Notes:

None.

Respectfully Yours,



François Raab, President

Bradley J. Askins, Auditor

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.

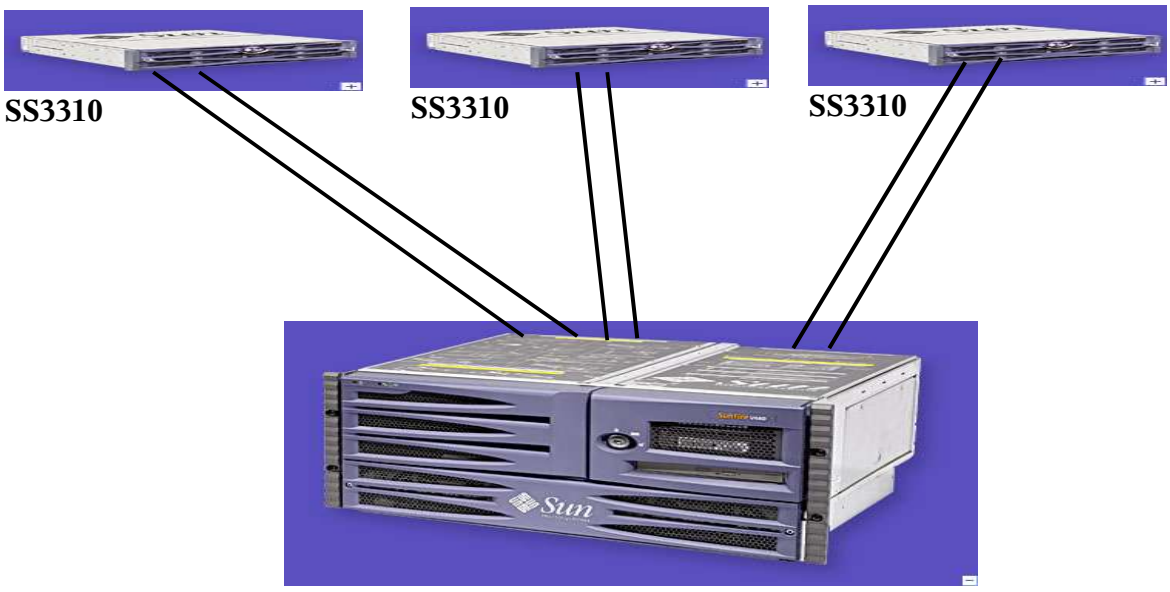
1.3 Configuration Diagram

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

The priced and measured configurations were identical:

3 X SS3310 SCSI Arrays

12 X 73 GB Disks



SUN Fire V490 Server

4 X 1350 MHz US IV processors

32 GB Memory

2 X 73 GB internal disks

2. Clause 1 Logical Database Design

2.1 Database Definition Statements

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

Appendix B contains the programs and scripts that create and analyze the tables 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. Column ordering was changed for some tables. Refer to the table create statements in Appendix B for further details.

2.3 Horizontal Partitioning

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

Horizontal partitioning was not used for any of the tables.

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.

All disks containing TPC-H tables, TPC-H indexes, the Sybase IQ utility db file and the Sybase IQ log file are housed on RAID1 volumes. A permanent irrecoverable failure of one of these disks was simulated by removing it from its array. After this was done processing continued without interruption, since the disk was a member of a RAID1 volume.. After an additional span of about 60 seconds, power to the server was cut off. The outcome is described in section 4.5.2.

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.

A system crash was produced by cutting off power to the V490 server. When power was restored, the system automatically 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).

A memory failure was simulated by cutting off power to the V490 server. The outcome is described in 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	*6000016166*
orders	*1,500,000,000*
Partsupp	*800,000,000*
Part	*200,000,000*
Customer	*150,000,000*
Supplier	*10,000,000*
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 stored on 16 RAID 1 volumes. Each volume was constructed from two raw partitions using the Solaris Volume Manager.
- The Temp database for Sybase IQ was configured using six raw partitions, two on each of two internal disks, and four on four of the external disks. The Temp databases were not mirrored.

The following table shows all the disk slices used for all the non-RAID1 devices.

Partition	Use	Size (in GB)
c1t9d0s0	root partition	3.91
c8t1d0s3	swap	3.9
c7t10d0s3	swap	5.9
c7t11d0s3	swap	5.9
c7t12d0s3	swap	5.9
c7t13d0s3	swap	5.9
c8t0d0s6	tempdb [via symbolic link /sybase2/T01]	62.49
c8t1d0s1	tempdb [via symbolic link /sybase2/T02]	62.44
c7t10d0s1	tempdb [via symbolic link /sybase2/T03]	62
c7t11d0s1	tempdb [via symbolic link /sybase2/T04]	62
c7t12d0s1	tempdb [via symbolic link /sybase2/T05]	62
c7t13d0s1	tempdb [via symbolic link /sybase2/T06]	62
c7t10d0s7	SVM state replica	.0015
c7t11d0s7	SVM state replica	.0015
c7t12d0s7	SVM state replica	.0015
c7t13d0s7	SVM state replica	.0015

The next table shows the disk slices used for the SVM RAID1 devices. The /sybase2 file system is used to store the IQ catalog file and IQ log file.

SVM Device Details:

Raw Partition Name	SVM Device Name	SVM Contained Device	Symbolic Link	Database Usage	Database Device Size GB	RAID
c8t0d0s1	d1	d2	none	/sybase2	2	RAID 1
c8t1d0s4	d1	d3	none	/sybase2)	2	RAID 1
c0t8d0s1	d15	d10	/sybase2/M01	IQ Main	68.35	RAID 1
c1t8d0s1	d15	d11	/sybase2/M01	IQ Main	68.35	RAID 1
c0t9d0s1	d25	d20	/sybase2/M02	IQ Main	68.35	RAID 1
c1t9d0s1	d25	d21	/sybase2/M02	IQ Main	68.35	RAID 1
c0t10d0s1	d35	d30	/sybase2/M03	IQ Main	68.35	RAID 1
c1t10d0s1	d35	d31	/sybase2/M03	IQ Main	68.35	RAID 1
c0t1d0s1	d45	d40	/sybase2/M04	IQ Main	68.35	RAID 1
c1t11d0s1	d45	d41	/sybase2/M04	IQ Main	68.35	RAID 1
c0t12d0s1	d55	d50	/sybase2/M05	IQ Main	68.35	RAID 1
c1t12d0s1	d55	d51	/sybase2/M05	IQ Main	68.35	RAID 1
c0t13d0s1	d65	d60	/sybase2/M06	IQ Main	68.35	RAID 1
c1t13d0s1	d65	d61	/sybase2/M06	IQ Main	68.35	RAID 1
c4t8d0s1	d75	d70	/sybase2/M07	IQ Main	68.35	RAID 1
c5t8d0s1	d75	d71	/sybase2/M07	IQ Main	68.35	RAID 1
c4t9d0s1	d85	d80	/sybase2/M08	IQ Main	68.35	RAID 1
c5t9d0s1	d85	d81	/sybase2/M08	IQ Main	68.35	RAID 1
c4t10d0s1	d95	d90	/sybase2/M09	IQ Main	68.35	RAID 1
c5t10d0s1	d95	d91	/sybase2/M09	IQ Main	68.35	RAID 1
c4t1d0s1	d105	d100	/sybase2/M10	IQ Main	68.35	RAID 1
c5t11d0s1	d105	d101	/sybase2/M10	IQ Main	68.35	RAID 1
c4t12d0s1	d115	d110	/sybase2/M11	IQ Main	68.35	RAID 1
c5t12d0s1	d115	d111	/sybase2/M11	IQ Main	68.35	RAID 1
c4t13d0s1	d125	d120	/sybase2/M12	IQ Main	68.35	RAID 1
c5t13d0s1	d125	d121	/sybase2/M12	IQ Main	68.35	RAID 1
c6t8d0s1	d135	d130	/sybase2/M13	IQ Main	68.35	RAID 1
c7t8d0s1	d135	d131	/sybase2/M13	IQ Main	68.35	RAID 1
c6t9d0s1	d145	d140	/sybase2/M14	IQ Main	68.35	RAID 1
c7t9d0s1	d145	d141	/sybase2/M14	IQ Main	68.35	RAID 1
c6t10d0s1	d155	d150	/sybase2/M15	IQ Main	68.35	RAID 1
c7t10d0s1	d155	d151	/sybase2/M15	IQ Main	68.35	RAID 1
c6t1d0s1	d165	d160	/sybase2/M16	IQ Main	68.35	RAID 1
c7t11d0s1	d165	d161	/sybase2/M16	IQ Main	68.35	RAID 1

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 utility db file and log file also resided on a RAID 1 devices.

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 =19 hrs 36 mins

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.

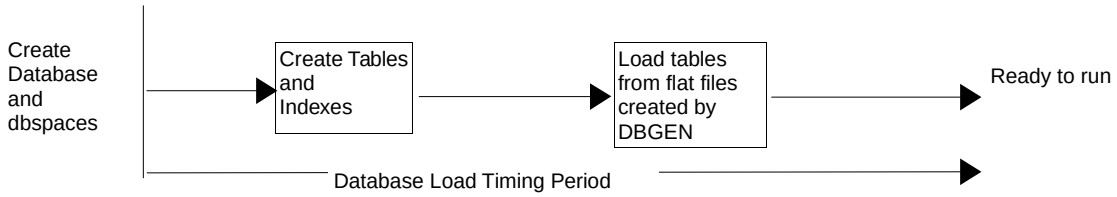
The data storage ratio is computed from the following information:

Disk Type	# Of Disks	Space Per Disk*	Sub-Total Disk Space**
internal	2	73 GB	135.97 GB
external	36	73 GB	2447.52 GB
		Total Space	**2583 GB**
		Data Storage Ratio	**2.58**

** 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 only the necessary adjustments for size differences.

6. Clause 5 Performance Metrics and Execution Rules

6.1 System Activity Between Load and Performance Tests

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

1. 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 reported on the page titled “Numerical Quantities”, contained in the beginning of this document and replicated in the Executive Summary document.

6.4 Number of Streams for the Throughput Test

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

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

6.6 Total Elapsed Time of the Measurement Interval

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

The total elapsed time of the throughput test is reported on the page titled “Numerical Quantities”, contained in the beginning of this document and replicated in the Executive Summary document.

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

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

The start and finish times for each refresh function:

Stream ID	Refresh Function	Start Date	Start Time	End Date	End Time
Stream 01	RF1	Jan 3, 2005	06:05:33	Jan 3, 2005	07:15:25
Stream 01	RF2	Jan 3, 2005	07:15:25	Jan 3, 2005	08:12:49
Stream 02	RF1	Jan 3, 2005	09:17:51	Jan 3, 2005	10:00:49
Stream 02	RF2	Jan 3, 2005	14:46:01	Jan 3, 2005	15:33:23
Stream 03	RF1	Jan 3, 2005	15:33:24	Jan 3, 2005	16:18:12
Stream 03	RF2	Jan 3, 2005	19:48:21	Jan 3, 2005	20:26:20
Stream 04	RF1	Jan 3, 2005	23:06:26	Jan 3, 2005	23:46:56
Stream 04	RF2	Jan 4, 2005	02:02:02	Jan 4, 2005	02:40:13
Stream 05	RF1	Jan 4, 2005	05:45:21	Jan 4, 2005	06:21:46
Stream 05	RF2	Jan 4, 2005	07:11:50	Jan 4, 2005	07:55:12
Stream 06	RF1	Jan 4, 2005	11:10:21	Jan 4, 2005	11:44:55
Stream 06	RF2	Jan 4, 2005	15:10:02	Jan 4, 2005	15:54:44
Stream 07	RF1	Jan 4, 2005	15:54:44	Jan 4, 2005	16:38:26
Stream 07	RF2	Jan 4, 2005	20:53:38	Jan 4, 2005	21:40:38

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

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

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

6.9 Performance Metrics

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

The performance metrics, and the numbers on which they are based, are reported on the page titled “Numerical Quantities”, contained in the beginning of this document and replicated in the Executive Summary document.

6.10 The Performance Metric and Numerical Quantities from Both Runs

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

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

QppH@1000GB	QthH@1000GB	QphH@1000GB
3289.4	**3610.5**	3,446.2
3310.5	3604.8	3454.5
0.64%	-0.16%	0.24%

6.11 System Activity Between Performance Tests

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

No activity occurred between Run 1 and Run 2. Moreover Sybase IQ was not restarted after the database load 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 newtest shellscript with the following arguments:

```
newtest all 1000 4m 64k 7 38 73 $142,918.15 1
```

Before proceeding with the actual phases of the TPC-H benchmark, newtest checks the validity of the arguments and displays all the important configuration options in effect. If any inconsistencies or violations of the TPC-H rules are detected, newtest aborts with an appropriate error message. It is designed to be largely idiot-proof, in the sense that it will not start a performance test unless (a) it is reasonably sure that the test configuration conforms to the TPC-H rules and (b) the user, after reviewing all the implict and explicit assumptions concerning the current configuration, gives to ok to proceed.

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

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

The load is performed when newtest executes the statements

```
dbisqlc -c "DSN=tpch" -q load_xxx.sql
```

where xxx is lineiten, orders, customer, parts, partsup, supplier, region and nation.

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

The Throughput Test is performed when newtest concurrently executes the 7 query stream scripts, stream [1-7].sql, together with the script update_throughput7.sql. The latter executes the refresh functions in parallel with the query streams according to the rules set out in the TPC-H specification document.

7.2 Implementation-Specific Layer

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

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

The performance tests are performed using dbisqlc. dbisqlc is a Sybase-provided utility that allows SQL statements to be executed against an Sybase IQ database. The dbisqlc utility is invoked from the command-line on the SUT. It reads input from files containing SQL statements and sends results to stdout. dbisqlc uses information in the .odbc.ini file to connect to the database. The performance test scripts utilizing dbisqlc can be found in Appendix E.

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

7.3 Profile-Directed Optimization

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

Profile-directed optimization was not used.

8. Clause 7 Pricing

8.1 Hardware and Software Used

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

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

8.2 Total Three Year Price

The total 3-year price of the entire configuration must be reported, including hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.

The total 3-year price of the configuration is $142,918.15. 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 Sun Microsystems Inc., Continential 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.

All hardware and software components used in the measured configuration are generally available as of mar 31, 2005.

9. Auditor's Information and Attestation Letter

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

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

Appendix A. Solaris 10 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

utility.cfg

```
#
# these are the default suggested startup parameters
for the asiq
#
# note that you will still need to provide the
# server with the server name on startup "-n", and
# possibly a different system port number
#
-n tpch490
-c  32m
-gd all
-gl all
-gm 15
-gp 4096
-ti 4400
-tl 300
-iqtc 128
-iqmc 128
```

tpch.cfg

```
# note the port number here must matche the one in
# .odbc.ini

-n tpch490
-x tcpip{port=3011}
-c  12m
-gd all
-gl all
-gm 12
-gp 4096
-ti 4400
-tl 300
-iqmc 19000
-iqtc 10000
-iqmt 1000
-iqgovern 12
-iqpartition 2
```

Sybase IQ Database Options

(altered from default)

options.sql

```
SET OPTION PUBLIC.Row_Counts='On';
SET OPTION PUBLIC.Main_Reserved_DBSpace_MB=300;

SET OPTION PUBLIC.Force_No_Scroll_Cursors='On';
SET OPTION PUBLIC.Max_IQ_Threads_Per_Connection=100;
SET OPTION PUBLIC.Query_Temp_Space_Limit=0;
SET OPTION PUBLIC.Hash_Thrashing_Percent=100;
SET OPTION
PUBLIC.SignificantDigitsForDoubleEquality=15;
SET OPTION PUBLIC.Flatten_Subqueries = 'On';
SET OPTION PUBLIC.Default_Like_Range_Selectivity = 1;
SET OPTION PUBLIC.Default_Having_Selectivity = 1;
SET OPTION PUBLIC.Max_Hash_Rows = 25000000;
SET OPTION PUBLIC.Sort_Phase1_Helpers=3;

SET OPTION PUBLIC.Sweeper_Threads_Percent=10;
SET OPTION PUBLIC.Prefetch_Threads_Percent=15;
SET OPTION PUBLIC.Wash_Area_Buffers_Percent = '20';
SET OPTION PUBLIC.Load_Memory_Mb=0;
SET OPTION PUBLIC.Append_Load='On';
SET OPTION PUBLIC.Garray_Fill_Factor_Percent=3;
SET OPTION PUBLIC.Minimize_Storage='On';
SET OPTION PUBLIC.Notify_Modulus=10000000;
SET OPTION PUBLIC.Allow_Nulls_By_Default='Off';
```

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=tpch490
CommLinks=tcpip{host=10.8.5.49;Port=3011}
DatabaseName=tpch
UserID=DBA
Password=SQL
DBG=yes
LOG=/tmp/tpch_odbc.log

[utility_db]
Driver=/export/home/sybase/asiq12/lib/dbodbc7_r.so.1
EngineName=tpch490
CommLinks=tcpip{host=10.8.5.49;Port=2638}
DatabaseName=utility_db
UserID=DBA
Password=SQL
DBG=yes
LOG=/tmp/utility_db_odbc.log
```

Solaris Parameters

(altered from default)

/etc/system

```
set tune_t_fsflushr=600
set autoup=36000000
```

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 (writer instance only)

```
================================================
create dbspace main2     as '/sybase2/M02' iq store;
create dbspace main3     as '/sybase2/M03' iq store;
create dbspace main4     as '/sybase2/M04' iq store;
create dbspace main5     as '/sybase2/M05' iq store;
create dbspace main6     as '/sybase2/M06' iq store;
create dbspace main7     as '/sybase2/M07' iq store;
create dbspace main8     as '/sybase2/M08' iq store;
create dbspace main9     as '/sybase2/M09' iq store;
create dbspace main10    as '/sybase2/M10' iq store;
create dbspace main11    as '/sybase2/M11' iq store;
create dbspace main12    as '/sybase2/M12' iq store;
create dbspace main13    as '/sybase2/M13' iq store;
create dbspace main14    as '/sybase2/M14' iq store;
create dbspace main15    as '/sybase2/M15' iq store;
create dbspace main16    as '/sybase2/M16' iq store;

create dbspace iqtemp2   as '/sybase2/T02' iq
temporary store;
create dbspace iqtemp3   as '/sybase2/T03' iq
temporary store;
create dbspace iqtemp4   as '/sybase2/T04' iq
temporary store;
create dbspace iqtemp5   as '/sybase2/T05' iq
temporary store;
create dbspace iqtemp6   as '/sybase2/T06' 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 bigint,
    o_custkey              unsigned int,
    o_orderstatus          char(1),
    o_totalprice           double precision,
    o_orderdate            date,
```

```
    o_orderpriority        char(15),
    o_clerk                char(15),
    o_shippriority         int,
    o_comment              varchar(79),
    PRIMARY KEY (o_orderkey)
  );
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 bigint,
    l_partkey              unsigned int,
    l_suppkey              unsigned int,
    l_linenumber           int,
    l_quantity             double precision,
    l_extendedprice        double precision,
    l_discount             double precision,
    l_tax                  double precision,
    l_returnflag           char(1),
    l_linestatus           char(1),
    l_shipdate             date,
    l_commitdate           date,
    l_receiptdate          date,
    l_shipinstruct         char(25),
    l_shipmode             char(10),
    l_comment              varchar(44)
  );

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+char(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'
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'
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'
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'
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'
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'
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_query1.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_throughput7.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;
tpch_wait;
call tpch_rf2 (c_path,'1');
commit;
tpch_wait;
call tpch_rf1 (c_path,'2');
commit;
tpch_wait;
call tpch_rf2 (c_path,'2');
commit;
tpch_wait;
call tpch_rf1 (c_path,'3');
commit;
tpch_wait;
call tpch_rf2 (c_path,'3');
commit;
tpch_wait;
call tpch_rf1 (c_path,'4');
commit;
tpch_wait;
call tpch_rf2 (c_path,'4');
commit;
tpch_wait;
call tpch_rf1 (c_path,'5');
commit;
tpch_wait;
call tpch_rf2 (c_path,'5');
commit;
tpch_wait;
call tpch_rf1 (c_path,'6');
commit;
tpch_wait;
call tpch_rf2 (c_path,'6');
commit;
tpch_wait;
call tpch_rf1 (c_path,'7');
commit;
tpch_wait;
call tpch_rf2 (c_path,'7');
commit;
set qstop = now(*);
select 'Stream updates STOP -- ', qstop ;
```

==========

gen_streams_new.ksh

==========

```
#!/bin/ksh

if (( $# < 3 ))
then
     echo "usage:  $0  seed  scale_factor
num_streams"
     exit
fi
```

```
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/pk
gs/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;

seed=$1
sf=$2
ns=$3

i=0

while ((i<=ns))
do

    qgen -c -p 0 -l qparm${i}.txt -i
$DSS_QUERY/init.sql -t $DSS_QUERY/complete.sql -r
$seed -s $sf \
    1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
21 22 > stream${i}.sql
    ((seed=seed+1))
    ((i=i+1))0
done

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 Setup

```
 Note: The instructions below pertain to the
controller
number and targets generated by the configuration used
in the benchmark. Even for identically configured
systems, the controller numbers and SCSI targets will
rarely match the numbers used here.


Perform the following steps to create the RAID
devices:

1) create the following 15 MB slices
     c7t10d0s7
     c7t11d0s7
     c7t12d0s7
     c7t13d0s7

using the Solaris format command

2) create the SVM state replicas via

     metadb -c 2 c7t10d0s7  c1t11d0s7 c7t12d0s7 \
                 c7t13d0s7

3) create 68.35 GB s0 slices on each of the following
   disks:
          c0t8d0
          c0t9d0
          c0t10d0
          c0t11d0
          c0t12d0
          c0t13d0
          c1t8d0
          c1t9d0
          c1t10d0
          c1t11d0
          c1t12d0
          c1t13d0
          c4t8d0
          c4t9d0
          c4t10d0
          c4t11d0
          c4t12d0
          c4t13d0
          c5t8d0
          c5t9d0
          c5t10d0
          c5t11d0
          c5t12d0
          c5t13d0
          c7t8d0
          c7t9d0
          c7t10d0
          c7t11d0
          c7t12d0
          c7t13d0
          c8t8d0
          c8t9d0
using the Solaris format command

4) create 16 mirrored devices via

#   c0 mirrored with c1
```

```
metainit d10 1 1  c0t8d0s0
metainit d11 1 1  c1t8d0s0
metainit  d15 -m d10
metattach d15 d11

metainit d20 1 1  c0t9d0s0
metainit d21 1 1  c1t9d0s0
metainit  d25 -m d20
metattach d25 d21

metainit d30 1 1  c0t10d0s0
metainit d31 1 1  c1t10d0s0
metainit  d35 -m d30
metattach d35 d31

metainit d40 1 1  c0t11d0s0
metainit d41 1 1  c1t11d0s0
metainit  d45 -m d40
metattach d45 d41

metainit d50 1 1  c0t12d0s0
metainit d51 1 1  c1t12d0s0
metainit  d55 -m d50
metattach d55 d51

metainit d60 1 1  c0t13d0s0
metainit d61 1 1  c1t13d0s0
metainit  d65 -m d60
metattach d65 d61

#   c4 mirrored with c5

metainit d70 1 1  c4t8d0s0
metainit d71 1 1  c5t8d0s0
metainit  d75 -m d70
metattach d75 d71

metainit d80 1 1  c4t9d0s0
metainit d81 1 1  c5t9d0s0
metainit  d85 -m d80
metattach d85 d81

metainit d90 1 1  c4t10d0s0
metainit d91 1 1  c5t10d0s0
metainit  d95 -m d90
metattach d95 d91

metainit d100 1 1  c4t11d0s0
metainit d101 1 1  c5t11d0s0
metainit  d105 -m d100
metattach d105 d101

metainit d110 1 1  c4t12d0s0
metainit d111 1 1  c5t12d0s0
metainit  d115 -m d110
metattach d115 d111

metainit d120 1 1  c4t13d0s0
metainit d121 1 1  c5t13d0s0
metainit  d125 -m d120
metattach d125 d121

#   c6 mirrored with c6

metainit d130 1 1  c6t8d0s0
metainit d131 1 1  c6t9d0s0
metainit  d135 -m d130
metattach d135 d131

metainit d140 1 1  c6t10d0s0
metainit d141 1 1  c6t11d0s0
metainit  d145 -m d140
metattach d145 d141

metainit d150 1 1  c6t12d0s0
metainit d151 1 1  c6t13d0s0
metainit  d155 -m d150
metattach d155 d151

# 1 mirror on c7

metainit d160 1 1  c7t8d0s0
metainit d161 1 1  c7t9d0s0
metainit  d165 -m d160
metattach d165 d161
```

```
5) create the followin 2 GB slices:
    c8t0d0s1
    c8t1d0s4
using the Solaris format command


6) create the mirrored volume d1 via the following:

metainit d2 1 1 c8t0d0s1
metainit d3 1 1 c8t1d0s4   # s1 gets paired with s4
metainit d1 -m d2
metattach d1 d3
```

File System Setup

```
7) create a filesystem on /dev/md/rdsk/d1


8) mount the filesystem on /sybase2
```

Database Device Links

```
# Create the symbolic links for the mirrored
# devices by running the following shellscript

cd /sybase2
ln -s  /dev/md/rdsk/d15   M01
ln -s  /dev/md/rdsk/d25   M02
ln -s  /dev/md/rdsk/d35   M03
ln -s  /dev/md/rdsk/d45   M04
ln -s  /dev/md/rdsk/d55   M05
ln -s  /dev/md/rdsk/d65   M06
ln -s  /dev/md/rdsk/d75   M07
ln -s  /dev/md/rdsk/d85   M08
ln -s  /dev/md/rdsk/d95   M09
ln -s  /dev/md/rdsk/d105  M10
ln -s  /dev/md/rdsk/d115  M11
ln -s  /dev/md/rdsk/d125  M12
ln -s  /dev/md/rdsk/d135  M13
ln -s  /dev/md/rdsk/d145  M14
ln -s  /dev/md/rdsk/d155  M15
ln -s  /dev/md/rdsk/d165  M16



# Create symbolic links for the temp devices
# by running the following shellscript

cd /sybase2
ln -s  /dev/rdsk/c8t0d0s6    T01
ln -s  /dev/rdsk/c8t1d0s1    T02
ln -s  /dev/rdsk/c7t10d0s1   T03
ln -s  /dev/rdsk/c7t11d0s1   T04
ln -s  /dev/rdsk/c7t12d0s1   T05
ln -s  /dev/rdsk/c7t13d0s1   T06
```

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',37734107,56586554400.7292032,53758257134.86945
63,55909065222.8284717,25.5220058532573342,38273.12973
46211374,.0499852958383577168,1478493
'N','F',991417,1487504710.38000107,1413082168.05409968
,1469649223.19436967,25.5164719205229819,38284.4677608
483374,.05009342667421348 09,38854
'N','O',74476040,111701729697.737336,106118230307.6073
83,110367043872.495174,25.5022267695849895,38249.11798
8907361,.049996586053555131,2920374
'R','F',37719753,56568041380.8983326,53741292684.60453
75,55889619119.8339581,25.5057936126907617,38250.85462
60985255,.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',721002.694799999952,2557.4
700000000003,'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',637029.566699999809,3790.8
9,'UNITED KINGDOM           ','01c9CILnNtfOQYmZj','33-
593-865-6378','accounts doze blithely! enticing, final
deposits sleep blithely special accounts. slyly
express accounts pla'
125341,'Customer#000125341',633508.086,4983.5100000000
006,'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',584878.113399999976,7321.10
999999999881,'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',576455.132,2030.1300000000
003,'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',566842.981499999881,7763.3
5,'VIETNAM                  ','7KzflgX MDOq7sOkI','31-
943-426-9837','dolphins sleep blithely among the slyly
final'
142549,'Customer#000142549',563537.236799999952,5085.9
899999999994,'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,'ROMAN
IA                  ','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 102172720
stream1 102172721
stream2 102172722
stream3 102172723
stream4 102172724
stream5 102172725
stream6 102172726
stream7 102172727
```

Query Parameters

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

stream0: 102172720

```
14      1996-06-01
2       1       NICKEL  EUROPE
9       medium
20      mint    1993-01-01      MOROCCO
6       1994-01-01      0.07    24
17      Brand#45        LG CASE
18      315
8       IRAQ    MIDDLE EAST     LARGE BURNISHED NICKEL
21      ARGENTINA
13      unusual accounts
3       FURNITURE       1995-03-24
22      15      21      29      11      34      20
10
16      Brand#14        PROMO PLATED    11      9       3
        30      10      42      21      29
4       1996-10-01
11      JORDAN  0.0000001000
15      1995-09-01
1       96
10      1993-11-01
19      Brand#23        Brand#14        Brand#34        5
        10      20
5       EUROPE  1994-01-01
7       MOROCCO IRAQ
12      MAIL    TRUCK   1996-01-01
```

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

stream1: 102172721

```
21      CHINA
3       MACHINERY       1995-03-10
18      313
5       MIDDLE EAST     1995-01-01
11      ARGENTINA       0.0000001000
7       GERMANY CANADA
6       1995-01-01      0.04    24
20      yellow  1996-01-01      ETHIOPIA
17      Brand#42        LG JAR
12      TRUCK   MAIL    1996-01-01
16      Brand#44        SMALL BRUSHED   47      16
37      21      6       32      22      41
15      1993-05-01
13      unusual accounts
10      1994-08-01
2       39      COPPER  AMERICA
8       CANADA  AMERICA MEDIUM BRUSHED NICKEL
14      1996-09-01
19      Brand#25        Brand#42        Brand#24
10      11      27
9       lemon
22      23      10      25      28      34      18
29
1       104
4       1994-07-01
==================================================
```

stream2: 102172722

```
==================================================
6       1995-01-01      0.09    25
17      Brand#44        LG CAN
14      1996-12-01
16      Brand#34        ECONOMY ANODIZED        37
21      2       40      16      8       26      12
19      Brand#23        Brand#34        Brand#23        6
        12      23
10      1993-05-01
9       indian
2       27      STEEL   MIDDLE EAST
15      1995-12-01
8       SAUDI ARABIA    MIDDLE EAST     MEDIUM PLATED
NICKEL
5       AFRICA  1995-01-01
22      29      17      21      14      15      32
13
12      RAIL    MAIL    1996-01-01
7       UNITED STATES   SAUDI ARABIA
13      unusual deposits
18      314
1       112
4       1997-02-01
20      indian  1995-01-01      ROMANIA
3       FURNITURE       1995-03-27
11      KENYA   0.0000001000
21      IRAN
==================================================
```

stream3: 102172723

```
==================================================
8       JAPAN   ASIA    MEDIUM ANODIZED BRASS
5       AMERICA 1995-01-01
4       1994-11-01
6       1995-01-01      0.07    25
17      Brand#41        MED CASE
7       MOZAMBIQUE      JAPAN
1       120
18      312
22      20      31      15      24      14      29
23
14      1997-03-01
9       ghost
10      1994-02-01
15      1993-09-01
11      BRAZIL  0.0000001000
20      sandy   1993-01-01      INDONESIA
2       15      BRASS   AMERICA
21      BRAZIL
19      Brand#35        Brand#12        Brand#22        1
        13      30
13      unusual deposits
16      Brand#14        STANDARD PLATED         24
14      8       37      28      39      42      18
12      REG AIR MAIL    1997-01-01
```

```
3       MACHINERY       1995-03-12

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

stream4: 102172724

```
================================================
5       ASIA    1995-01-01
21      ROMANIA
14      1997-06-01
19      Brand#32        Brand#45        Brand#11        6
        14      27
15      1996-04-01
17      Brand#43        MED JAR
12      SHIP    MAIL    1997-01-01
6       1995-01-01      0.04    24
4       1997-06-01
9       drab
8       EGYPT   MIDDLE EAST     SMALL POLISHED BRASS
16      Brand#44        MEDIUM POLISHED         27
12      40      11      9       31      17      21
11      MOROCCO 0.0000001000
2       2       NICKEL  MIDDLE EAST
10      1994-12-01
18      313
1       67
13      unusual deposits
7       INDIA   EGYPT
22      34      17      11      23      27      24
16
3       BUILDING        1995-03-29
20      deep    1996-01-01      UNITED STATES
================================================
```

stream5: 102172725

```
================================================
21      IRAQ
15      1993-12-01
4       1995-03-01
6       1996-01-01      0.02    25
7       ALGERIA VIETNAM
16      Brand#34        PROMO ANODIZED 5        33
24      15      7       10      18      32
19      Brand#34        Brand#33        Brand#11        1
        15      23
18      315
14      1997-10-01
22      11      12      13      28      16      17
26
11      CANADA  0.0000001000
13      unusual deposits
3       MACHINERY       1995-03-14
1       75
2       40      TIN     ASIA
5       EUROPE  1996-01-01
8       VIETNAM ASIA    SMALL BURNISHED BRASS
20      pale    1995-01-01      KENYA
12      FOB     SHIP    1993-01-01
17      Brand#45        MED CAN
10      1993-09-01
9       cream

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

stream6: 102172726

```
================================================
10      1994-06-01
3       BUILDING        1995-03-31
15      1996-07-01
13      express deposits
6       1996-01-01      0.07    25
8       JORDAN  MIDDLE EAST     STANDARD BRUSHED BRASS
9       chartreuse
7       PERU    JORDAN
4       1997-10-01
11      MOZAMBIQUE      0.0000001000
22      11      14      20      25      18      34
23
18      312
12      MAIL    FOB     1997-01-01
1       83
5       MIDDLE EAST     1996-01-01
16      Brand#14        SMALL BURNISHED         31
28      18      34      4       13      1       30
2       28      STEEL   MIDDLE EAST
14      1998-01-01
19      Brand#41        Brand#11        Brand#15        7
        16      30
20      blanched        1993-01-01      CANADA
17      Brand#42        JUMBO CASE
21      EGYPT
================================================
```

stream7: 102172727

```
================================================
18      314
8       ETHIOPIA        AFRICA  STANDARD PLATED BRASS
20      lime    1997-01-01      CHINA
21      VIETNAM
2       16      BRASS   ASIA
4       1995-07-01
22      28      21      15      29      17      14
24
17      Brand#44        JUMBO JAR
1       91
11      EGYPT   0.0000001000
9       blanched
19      Brand#43        Brand#54        Brand#54        2
        17      26
3       HOUSEHOLD       1995-03-16
13      express packages
5       AFRICA  1996-01-01
7       INDONESIA       ETHIOPIA
10      1993-03-01
16      Brand#44        LARGE POLISHED 35       25
43      48      19      36      8       46
6       1996-01-01      0.05    24
14      1993-04-01
15      1994-04-01
12      TRUCK   FOB     1993-01-01
```

Appendix E. Implementation-Specific Layer/Driver Code

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

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

newtest

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


cur_dir=$(pwd)

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


if (( $# < 1 ))
then
   echo
   echo "usage: $0    scope   (plus additional args
depending upon scope)                "
   echo
   echo "       scope values: "
   echo
   echo "           - restart     restart IQ with new
tpch.cfg, options.sql or MPSS values          "
   echo "           - load        create and load a
tpch database from scratch                    "
   echo "           - qi          do query i
(i=1,2,...,22)                                  "
   echo "           - stream0     do all 22 queries
without any refreshes                         "
   echo "           - refresh     do a single rerfesh
pair                                          "
   echo "           - power       do a single power
run                                           "
   echo "           - throughput  do a single
throughput run                                "
   echo "           - run1        do a single power-
throughput cycle (without a load)             "
   echo "           - all         do full audit run
(load + 2 power-throughput cycles + reports)  "
   echo
   exit
fi

scope=$1

case $scope in
       restart)  if (( $# < 3 || $# > 3 ))
                 then
                       echo
                       echo "usage: $0    restart
mpssheap   mpssstack   "
                       echo
                       exit
                 fi

                 mpssheap=$2
                 mpssstack=$3  ;;

          load)  if (( $# < 4  || $# > 4 ))
                 then
                       echo
                       echo "usage: $0    load
scale_factor  mpssheap   mpssstack   "
                       echo
                       exit
                 fi

                 let sf=$2
                 mpssheap=$3
                 mpssstack=$4  ;;

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

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

                 let sf=$2  ;;

                 # assumes asiq is running; no
shutdown and restart
                 # note that this arrangement allows
for a warmed up cache

       stream0)  if (( $# < 5 || $# > 5 ))
                 then
                       echo
                       echo "usage: $0    stream0
scale_factor  mpssheap   mpssstack   "
                       echo "
use_new_seed(0 or 1)          "
                       echo
                       exit
                 fi

                 let sf=$2
                 mpssheap=$3
                 mpssstack=$4
                 let use_new_seed=$5   ;;

       refresh)  if (( $# < 4 || $# > 4 ))
                 then
                       echo
                       echo "usage: $0    refresh
scale_factor  mpssheap   mpssstack   "
                       echo
                       exit
                 fi

                 let sf=$2
                 mpssheap=$3
                 mpssstack=$4  ;;

         power)  if (( $# < 5 || $# > 5 ))
                 then
                       echo
                       echo "usage: $0    power
scale_factor  mpssheap   mpssstack   "
                       echo "
use_new_seed(0 or 1)          "
                       echo
                       exit
                 fi

                 let sf=$2
                 mpssheap=$3
                 mpssstack=$4
                 let use_new_seed=$5  ;;

    throughput)  if (( $# < 6 || $# > 6 ))
                 then
                       echo
                       echo "usage: $0    throughput
scale_factor  mpssheap   mpssstack   "
                       echo "
num_refresh_streams           "
```

```
                        echo "
use_new_seed(0 or 1)          "
                        echo
                        exit
                fi

                let sf=$2
                mpssheap=$3
                mpssstack=$4
                nrfs=$5
                let use_new_seed=$6   ;;

          run1)  if (( $# < 9 || $# > 9 ))
                 then
                        echo
                        echo "usage: $0    run1
scale_factor  mpssheap  mpssstack          "
                        echo "
num_refresh_streams                         "
                        echo "
num_disks  disk_size(in GB)  system_cost    "
                        echo "
use_new_seed(0 or 1)                        "
                        echo
                        exit
                        echo
                fi

                let sf=$2
                mpssheap=$3
                mpssstack=$4
                nrfs=$5
                let nd=$6
                let ds=$7
                let sc=$8
                let use_new_seed=$9  ;;

           all)  if (( $# < 9 || $# > 9 ))
                 then
                        echo
                        echo "usage: $0    all
scale_factor  mpssheap  mpssstack           "
                        echo "
num_refresh_streams                         "
                        echo "
num_disks  disk_size(in GB)  system_cost    "
                        echo "
use_new_seed(0 or 1)                        "
                        echo
                        exit
                        echo
                fi

                let sf=$2
                mpssheap=$3
                mpssstack=$4
                nrfs=$5
                let nd=$6
                let ds=$7
                let sc=$8
                let use_new_seed=$9  ;;

             *)  echo
                 echo "ERROR: scope (=$scope) must be
one of:                                    "
                 echo
                 echo "
restart,load,q1,q2,...,q22,stream0,refresh,power,throu
ghput,run1,all  "
                 echo
                 echo
                 exit
esac


# make sure server is configured with enough
connections to run the
# specified number of streams

max_connections=$(grep '\-gm' tpch.cfg|cut -d' ' -f 2)

((min_req_connections=nrfs+1))

if ((max_connections<min_req_connections))
then
      echo
      echo "In order to run $nrfs query streams  -gm
in tpch.cfg must be at least $min_req_connections"
      echo "-gm is currently set to only
$max_connections"
      echo
      exit
fi


echo "      qm = max_connections = $(grep '\-gm'
tpch.cfg|cut -d' ' -f 2)"
# make sure MPSS values are legal for the scopes where
they have been supplied
# as commandline args (i.e. everything but the
standalone queries)

if [[ $scope != q* ]]
then
      if [[ $mpssheap != 8k && $mpssheap != 64k &&
$mpssheap != 512k && $mpssheap != 4m &&
            $mpssheap != 8K && $mpssheap != 64K &&
$mpssheap != 512K && $mpssheap != 4M    ]]
      then
            echo
            echo "ERROR: mpssheap (=$mpssheap) must be
one of [8k,64k,512k,4m]  "
            echo "       with k and m case-insensitive
"
            echo
            exit
      fi


      if [[ $mpssstack != 8k && $mpssstack != 64k &&
$mpssstack != 512k && $mpssstack != 4m &&
            $mpssstack != 8K && $mpssstack != 64K &&
$mpssstack != 512K && $mpssstack != 4M    ]]
      then
            echo
            echo "ERROR: mpssstack (=$mpssstack) must
be one of [8k,64k,512k,4m]  "
            echo "       with k and m case-insensitive
"
            echo
            exit
      fi
fi


# make sure use_new_seed value is legal for the scopes
where it is been supplied
# as a commandline arg

if [[ $scope = stream0 || $scope = power || $scope =
throughput || $scope = run1 || $scope = all ]]
then
      if ((use_new_seed != 0 && use_new_seed != 1))
      then
            echo
            echo "ERROR: use_new_seed must be 0 or 1"
            echo
            exit
      fi
fi



# make sure disk_size is reasonable for the scopes
where it is supplied
# as a commandline arg

if [[ $scope = run1 || $scope = all ]]
then
      if ((ds < 36))
      then
            echo
            echo "ERROR: disk_size (=$ds) appears to be
too small"    #  this may need to be changed
            echo
            exit
      fi

      if ((ds > 146))
      then
            echo
            echo "ERROR: disk_size (=$ds) appears to be
```

```
too large"  #  this may need to be changed
            echo "       if this is not he case, modify
the script $0"
            echo
            exit
      fi
fi


# check that the scale factor is legal where it has
been supplied as
# a command line arg (i.e. everything but restart)

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


# make sure that the number of refresh streams is at
last the minimum
# for the specified scale factor, when refreash
streams are required

if [[ $scope = throughput || $scope = run1 || $scope =
all ]]
then
      case $sf in
             1 )
                  mrfs=2
                  if ((nrfs<2))    # nrfs: number of
refresh streams
                  then
                        echo
                        echo "ERROR: number of refresh
stream (=$nrfs) must be at least 2 for scale $sf"
                        echo
                        exit
                  fi
                  echo " -- scale factor $sf not fully
implemented due to incomplete tpch_wait"
                  echo;;
           100 )
                  mrfs=5
                  if ((nrfs<5))    # nrfs: number of
refresh streams
                  then
                        echo
                        echo "ERROR: number of refresh
stream (=$nrfs) must be at least 5 for scale $sf"
                        echo
                        exit
                  fi ;;
           300 )
                  mrfs=6
                  if ((nrfs<6))    # nrfs: number of
refresh streams
                  then
                        echo
                        echo "ERROR: number of refresh
stream (=$nrfs) must be at least 6 for scale $sf"
                        echo
                        exit
                  fi
                  echo
                  echo " -- scale factor $sf not fully
implemented due to incomplete tpch_wait"
                  echo;;
          1000 )
                  mrfs=7
                  if ((nrfs<7))    # nrfs: number of
refresh streams
                  then
                        echo
                        echo "ERROR: number of refresh
stream (=$nrfs) must be at least 7 for scale $sf"
                        echo
                        exit
                  fi  ;;
          3000 )
                  mrfs=8
                  if ((nrfs<8))    # nrfs: number of
refresh streams
                  then
                        echo
                        echo "ERROR: number of refresh
stream (=$nrfs) must be at least 8 for scale $sf"
                        echo
                        exit
                  fi
                  echo
                  echo " -- scale factor $sf not fully
implemented due to incomplete tpch_wait"
                  echo;;
         10000 )
                  mrfs=9
                  if ((nrfs<9))    # nrfs: number of
refresh streams
                  then
                        echo
                        echo "ERROR: number of refresh
stream (=$nrfs) must be at least 9 for scale $sf"
                        echo
                        exit
                  fi
                  echo
                  echo " -- scale factor $sf not fully
implemented due to incomplete tpch_wait"
                  echo;;
             * )
                  echo
                  echo "ERROR:  scale factor (=$sf)
must be one of (1,100,300,1000,3000,10000)"
                  echo
                  exit
                  echo;;
      esac
fi


# make sure /sybase_stage contains the right amount of
data for the supplied scale factor
# whenever a load is done

if [[ $scope = load || $scope = all ]]
then
     cd /sybase_stage
     dbg=$(dbgen_size)
     if (( dbg < sf*1000000000 ))
     then
           echo
           echo "ERROR: dbgen files in /sybase_stage
contain only $(show_dbgen_size)"
           echo "        --  not enough for a scale
factor of $sf -- "
           echo
           exit
           echo
     fi

     if (( dbg > 2*sf*1000000000 ))
     then
           echo
           echo "ERROR: dbgen files in /sybase_stage
contain $(show_dbgen_size)"
           echo "       --  too much for a scale
factor of $sf --  "
           echo
           exit
     fi
     cd -
fi

# make sure the number of lineitem.tbl files in /
sybase_stage matches the number
# in load_lineitem.sql

if [[ $scope = load || $scope = all ]]
then
      liss=$(ls /sybase_stage/lineitem.tbl* | grep -v
u | wc -l)
      lill=$(grep lineitem load_lineitem.sql | wc -l)

      if ((liss>lill))
```

```
      then
            echo
            echo $liss lineitem.tbl files in /
sybase_stage but only $lill in load_lineitem.sql
            echo
            exit
      fi

      if ((lill>liss))
      then
            echo
            echo $liss lineitem.tbl files in /
sybase_stage but $lill in load_lineitem.sql
            echo
            exit
      fi

      oss=$(ls /sybase_stage/orders.tbl* | grep -v u |
wc -l)
      olo=$(grep orders load_orders.sql | wc -l)

      if ((oss>olo))
      then
            echo
            echo $oss orders.tbl files in /
sybase_stage but only $olo in load_orders.sql
            echo
            exit
      fi

      if ((olo>oss))
      then
            echo
            echo $oss orders.tbl files in /
sybase_stage but $olo in load_orders.sql
            echo
            exit
      fi
fi


# decide if orderkey should be bigint or int for
thoses scopes where it matters

if [[ $scope != q* && $scope != stream0 && $scope !=
restart ]]
then
      if ((sf >= 1000))
      then
         big=big     #  make l_orderkey and o_orderkey
bigint
      else
         big=        #  otherwise int
      fi
fi

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

echo " "
case $scope in
           q* )
                  echo "DOING QUERY $query_num  with: "
;;
          all )
                  echo "DOING FULL AUDIT TEST (load,
plus 2 full runs) with: " ;;
         load )
                  echo "ONLY DOING LOAD (and create
database) with: " ;;
        power )
                  echo "ONLY DOING ONE POWER TEST
with:" ;;
      refresh )
                  echo "ONLY DOING ONE REFRESH PAIR
with: " ;;
   throughput )
                  echo "ONLY DOING ONE THROUGHPUT TEST
with:" ;;
         run1 )
                  echo "ONLY DOING ONE POWER FOLLOWED
BY ONE THROUGHPUT TEST with:" ;;
      stream0 )
                  echo "ONLY DOING 22 SINGLE-USER
QUERIES (NO REFRESHES):" ;;
      restart )
                  echo "ONLY RESTARTING IQ with:" ;;
            * )
                  echo "ERROR:  scope (=$scope) must be
one of"
                  echo
"           (all,restart,load,power,refresh,stream0,th
roughput,run1)"
                  echo
                  exit ;;
esac

echo " "



if [[ $scope != restart ]]
then
      echo "      scale factor = $sf "
fi

if [[ $scope = all || $scope = load ]]
then
      cd /sybase_stage
      s=$(show_dbgen_size)
      cd -
      echo "      dbgen files: $s "
      echo "      $(grep -h "IQ PAGE SIZE"
create_database.sql)"
else
      echo "      IQ PAGE SIZE unchanged from existing
value"
fi


if [[ $scope != q* ]]
then
      echo "      MPSSHEAP = $mpssheap      "
      echo "      MPSSSTACK = $mpssstack    "
else
      echo "      MPSS values unchanged from existing
values"
fi

if [[ $scope = refresh || $scope = q* ]]
then
      echo "      iqmc and iqtc unchanged from
existing values"
else  # going to restart asiq
      echo "      iqmc = $(grep iqmc tpch.cfg|cut -d'
' -f 2) MB"
      echo "      iqtc = $(grep iqtc tpch.cfg|cut -d'
' -f 2) MB"
fi


if [[ $scope = all || $scope = run1 || $scope =
throughput ]]
then
      echo "      num refresh streams = $nrfs
(minimum for this scale factor: $mrfs"
      echo "      max connections = $(grep gm
tpch.cfg|cut -d' ' -f 2) "
fi


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

      echo

      echo "      $(grep "IQ PATH" create_database.sql
| tr -d "'"  | tr -s ' ' | cut -d' ' -f3)"

      for i in $(grep "iq store" create_dbspaces.sql |
tr -d "'" | tr -s ' ' | cut -d' ' -f5)
      do
         echo "      $i"
```

```
      done
      echo
      echo "      $(grep "TEMPORARY PATH"
create_database.sql | tr -d "'"  | tr -s ' '| cut -d'
' -f3)"

      for i in $(grep "temporary store"
create_dbspaces.sql | tr -d "'" | tr -s ' '| cut -d' '
-f5)
      do
         echo "      $i"
      done
      echo
fi


if [[ $scope != load && $scope != refresh && $scope !=
restart ]]
then
   if ((use_new_seed==1))
   then
         echo "      using newly gererated seed"
   else
         echo "      using existing seed"
   fi
   echo
fi


adb -k >/tmp/out <<END
tune_t_fsflushr/D
END
echo "      $(grep tune_t_fsflushr /tmp/out | tail
-1)"

adb -k >/tmp/out <<END
autoup/D
END
echo "      $(grep autoup /tmp/out | tail -1)"

adb -k >/tmp/out <<END
maxphys/D
END
echo "      $(grep maxphys /tmp/out | tail -1)"

if [[ $scope = refresh || $scope = q* ]]
then
      echo "      options.sql unchanged "
else  # going to restart asiq and rerun options.sql
      echo
      echo "      $(grep Max_Hash_Rows
options.sql)"
      echo "      $(grep Range_Selectivity
options.sql)"
      echo "      $(grep Having_Selectivity
options.sql)"
      echo "      $(grep Prefetch_Threads
options.sql)"
      echo "      $(grep Sweeper_Threads
options.sql)"
      echo "      $(grep Wash_Area
options.sql)"
fi

echo
echo

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


if [[ $ans = n || $ans = no ]]
then
     echo
     echo change options.sql, tpch.cfg,
create_database.sql or /etc/system and try again
     echo
     exit
fi

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

echo

if [[ $scope = q* ]]
then
     # assumes asiq is running; no shutdown and
restart
     do_one_query $query_num $sf
     echo
     exit
fi

echo
echo

# stop & restart IQ in all cases
# note: the qi case does not do a restart, but  qi
# exits a few line above
# when scope is load or all, the database is re-
created

# stop IQ if it's running
asiq_running=`ps -eaf | grep asiq | grep -v grep | wc
-l`
if ((asiq_running > 0))
then
     echo "Stopping IQ"
     dbstop -c "DSN=tpch" -y
fi

if [[ $scope = load || $scope = all ]]
then
     # remove the old stuff in /sybase2 and
     # then recreate the tpch database

     if [[ -f /sybase2/tpch.db ]]
     then
        rm -f /sybase2/tpch.db

        # there have been instances when this file
hasn't been removed; if
        # this happens exit immediately

        if [[ -f /sybase2/tpch.db ]]
        then
            echo
            echo unable to remove /sybase2/tpch.db
            echo
            exit
        fi

        echo "removed  /sybase2/tpch.db"
     fi

     if [[ -f /sybase2/tpch.log ]]
     then
        rm -f /sybase2/tpch.log

        # there have been instances when this file
hasn't been removed; if
        # this happens exit immediately

        if [[ -f /sybase2/tpch.log ]]
        then
            echo
            echo unable to remove /sybase2/tpch.log
            echo
            exit
        fi

        echo "removed  /sybase2/tpch.log"
     fi

     if [[ -f /sybase2/tpch.iqmsg ]]
     then
        ts=`date '+%m%d%H%M%S'`
        mv /sybase2/tpch.iqmsg  /
sybase2/tpch.iqmsg.$ts
        echo "renamed  /sybase2/tpch.iqmsg  to  /
sybase2/tpch.iqmsg.$ts"
     fi

     echo " "
     echo " "

     echo " "
     echo "Starting IQ with utility database: `date`
"

     #echo Sleeping  5 Seconds
```

```
     echo "  "
     sleep 5

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

     start_asiq @utility.cfg
     echo IQ started with utility database: `date`
     echo "  "
     echo "    Creating tpch database: `date`"
     echo "  "
     dbisqlc -c "DSN=utility_db" -q
create_database.sql  >cud.ooo
     if [[ -s cud.ooo ]]   # cud.ooo exists and non-
empty
     then
          echo "    tpch database created: `date`"
     else
          echo
          echo ERROR: create_database.sql  failed
          echo
          exit
     fi
     echo "  "
     echo "  "

     echo "    Shutting down IQ: `date`  "
     echo "  "
     sleep 5
     dbstop -c "DSN=utility_db" -y
     echo
     echo "IQ utility database shutdown complete:
`date`  "
     echo "  "
     echo "  "
fi

echo "sleeping 5 secs"
sleep 5


echo "Restarting IQ with tpch database: `date`   "
start_asiq @tpch.cfg /sybase2/tpch.db
echo "    IQ tpch database started: `date`   "
echo "  "

sleep 5

if [[ $scope = load || $scope = all ]]
then
     if [[ -e cdbs.ooo ]]
     then
          rm cdbs.ooo
     fi

     dbisqlc -c "DSN=tpch" -q create_dbspaces.sql
>cdbs.ooo
     if [[ -s cdbs.ooo ]]
     then
          echo "         dbspaces added"
     else
          echo
          echo ERROR: create_dbspaces.sql  failed
          echo
          exit
     fi

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

     dbisqlc -c "DSN=tpch" -q tpch_rf_${big}int.sql
>tpchrf.ooo
     if [[ -s tpchrf.ooo ]]
     then
          echo "         executed tpch_rf_${big}
int.sql"
     else
          echo
          echo ERROR: tpch_rf_${big}int.sql failed
          echo
          exit
     fi

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

     dbisqlc -c "DSN=tpch" -q tpch_wait.sql
>tpchw.ooo
     if [[ -s tpchw.ooo ]]
     then
          echo "         executed tpch_wait.sql"
     else
          echo
          echo ERROR: tpch_wait.sql failed
          echo
          exit
     fi

     echo "  "

fi

if [[ -e o.ooo ]]   # set options
then
     rm o.ooo
fi

dbisqlc -c "DSN=tpch" -q options.sql         >o.ooo
if [[ -s o.ooo ]]
then
      echo "         executed options.sql"
else
      echo
      echo ERROR: options.sql failed
      echo
      exit
fi

echo


sleep 5

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

sleep 5

echo "Restarting IQ with tpch database and setting
MPSS values: `date`   "
my_start_asiq $mpssheap  $mpssstack  @tpch.cfg  /
sybase2/tpch.db
echo "IQ restarted: `date`   "
echo "  "

if [[ $scope = restart ]]
then
      echo
      echo exiting $0
      echo
      exit
fi

if [[ $scope = load || $scope = all ]]
then
     sleep 5

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

     dbisqlc -c "DSN=tpch" -q create_tables_${big}
int.sql  >ct.ooo
     if [[ -s ct.ooo ]]
     then
          echo "    executed create_tables_${big}
int.sql"
     else
          echo
          echo ERROR: create_tables_${big}int.sql
failed
          echo
```

```
          exit
     fi

     sleep 5

     #
     # Load the database
     #
     echo "  "

     show_asiq_cpu  start_load  > start_load_cpu.out

     echo "   Load Started  `now_iq_format.bash`"  |
tee start_load.out
     dbisqlc -c "DSN=tpch" -q load_lineitem.sql >
load_lineitem.out &
     loadlpid=$!
     echo " "
     echo "      - lineitem load started: `date` "
     echo " "
     dbisqlc -c "DSN=tpch" -q load_region.sql
     echo "      - region load completed: `date` "
     dbisqlc -c "DSN=tpch" -q load_nation.sql
     echo "      - nation load completed: `date` "
     dbisqlc -c "DSN=tpch" -q load_customer.sql
     echo "      - customer load completed: `date` "
     dbisqlc -c "DSN=tpch" -q load_part.sql
     echo "      - part load completed: `date` "
     dbisqlc -c "DSN=tpch" -q load_supplier.sql
     echo "      - supplier load completed: `date` "
     dbisqlc -c "DSN=tpch" -q load_partsupp.sql
     echo "      - partsupp load completed: `date` "
     dbisqlc -c "DSN=tpch" -q load_orders.sql
     loadopid=$!
     wait $loadopid
     echo "      - orders load completed: `date` "
     wait $loadlpid
     echo "      - lineitem load completed: `date` "
     echo
     echo "   Load Finished  `now_iq_format.bash` " |
tee end_load.out
     echo
     show_asiq_cpu  end_load  > end_load_cpu.out

     slt=`tr -s ' ' <start_load.out | cut -d' ' -f 4,5
`
     elt=`tr -s ' ' <end_load.out | cut -d' ' -f 4,5 `
     lt=`calculate_load_time.bash "$elt" "$slt" `  #
calls secs_from_epoch.pl
     echo "Database Load Time:  $lt"
     echo " "

     if [[ $scope = load ]]
     then
          echo exiting $0
          echo
          exit
     fi
fi


# note: load has exited

if [[ $scope = stream0 || $scope = power || $scope =
throughput || $scope = run1 || $scope = all ]]
then
     if ((use_new_seed==1))
     then
           seed=`date '+%m%d%H%M%S'`

           echo "Generating $nrfs Query Streams "
           echo "Using Generated Seed = $seed "

           #gen_streams.ksh $seed $sf
           gen_streams_new.ksh    $seed  $sf  $nrfs
     else
           lu_seed=$(grep "as a seed" stream0.sql |
cut -d 'g' -f2 | cut -d 'a' -f1)
           seed=$lu_seed
           echo  "Using Existing Seed = $seed"
     fi
     echo
fi

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

     # After the load Completes run the Audit SQL

     dbisqlc -c "DSN=tpch" -q dbtables-syb.sql >
rdbtablest_start.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

     dbisqlc -q -c "DSN=tpch" check_options.sql >
check_options_start.out

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

# delete all streami.out files so that if the number
of streams in the current run is
# less than the number of streams from a prior run,
there won't be any old stream.out
# files left in the directory.
# This is important because some of the report
calculations use all the stream.out files
# in the current directory. If some stream.out files
belong to prior runs, then some of
# the report values will not be correct.

rm stream*.out

echo
echo "removed stream*.out"
echo
echo

if [[ $scope = refresh || $scope = stream0 || $scope =
power || $scope = run1 || $scope = all ]]
then
     if [[ $scope = power || $scope = run1 || $scope =
all ]]
     then
          echo
          echo "Start Run 1 Power Test   `date` "
          echo "----------------------          "
     elif [[ $scope = stream0 ]]
     then
          echo
          echo "Start Stream0   `date` "
          echo "-------------          "
     else
          echo
          echo "Start Refresh   `date` "
          echo "-------------          "
     fi


     if [[ $scope = refresh || $scope = power ||
$scope = run1 || $scope = all ]]
     then
          # 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

          show_asiq_cpu  start_power  >
start_power_cpu.out
          show_asiq_cpu  start_refresh1  >
start_refresh1_cpu.out
          echo
          echo "     Start RF1 in the background
`date`  "
          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 (in
update_power.sql) 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 10   # Sleep while Trigger file
exists
          done

          show_asiq_cpu  end_refresh1  >
end_refresh1_cpu.out

          # RF1 has completed and RF2 is waiting for
the Query Stream to Complete
     fi


     if [[ $scope = stream0 || $scope = power ||
$scope = run1 || $scope = all ]]
     then
           show_asiq_cpu  start_stream0
>start_stream0_cpu.out
           echo
           echo "     Start Query Stream   `date` "
           dbisqlc -c "DSN=tpch" -q stream0.sql >
stream0.out

           echo
           echo "     Finish Query Stream   `date` "
           show_asiq_cpu  end_stream0
>end_stream0_cpu.out
     fi


     if [[ $scope != stream0 ]]  # do refresh2
     then
           # 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
           #
           show_asiq_cpu  start_refresh2  >
start_refresh2_cpu.out
           echo
           echo "     Start RF 2   `date` "
           wait $rfspid
           echo "     End RF 2    `date`  "
           show_asiq_cpu  end_refresh2  >
end_refresh2_cpu.out
           show_asiq_cpu  end_power
>end_power_cpu.out
     else
           echo
           echo "End Stream0   `date`"
           echo "-----------          "
           echo make report by falling through to
report section
           echo
     fi


     if [[ $scope = refresh ]]
     then
           echo
           echo "     End RF 2    `date`    "
           echo "--------------------------"
           echo
           echo
           tpch_power_response_times.bash   refresh
           echo
           exit   # on refresh
     else
           echo
           echo "End Run 1 Power  Test    `date` "
           echo "---------------------          "
           echo
           echo
           ps -eaf | grep asiq| grep -v grep | tr -s '
' | cut -d ' ' -f8,9
           echo
     fi
fi

if [[ $scope = throughput || $scope = run1 || $scope =
all ]]
then
     show_asiq_cpu  start_throughput
>start_throughput_cpu.out
     echo
     echo "Start Run 1 Throughput `date`  "
     echo "----------------------         "
     echo "  "
     echo "     Start Query Streams  `date` "
     echo "  "

     ((i=1))
     while ((i<=nrfs))  # run all query streams
concurrently
     do
        dbisqlc -c "DSN=tpch" -q stream${i}.sql >
stream${i}.out &
        #  qs${i}pid=$!   not needed anymore since we
do a wait (for everything)
        ((i=i+1))
     done

     echo "     Start Refresh Streams  `date` "
     echo "  "

     dbisqlc -c "DSN=tpch" -q update_throughput$
{nrfs}.sql > update_throughput.out &
     rf0pid=$!

     wait $rf0pid
     echo "     All refresh streams have completed
`date`   "

     wait   # for everything
     echo "     All query and refresh streams have
completed  `date`   "
     echo
     echo "End Run 1 Throughput `date`  "
     echo "---------------------       "
     echo
     ps -eaf | grep asiq | grep -v grep | tr -s ' ' |
cut -d ' ' -f8,9
     echo
     show_asiq_cpu  end_throughput
>end_throughput_cpu.out
fi

if [[ $scope = stream0 || $scope = power || $scope =
throughput || $scope = run1 || $scope = all ]]
then
     dayHr=`date '+%m%d%H'`
     echo
     echo "Producing mrun1_${scope}_${sf}_report_$
{dayHr}.out "
     tpch_report.bash $scope $sf $mpssheap $mpssstack
run1_${scope}_${day} $seed $nrfs $nd $ds $sc \

> mrun1_${scope}_${sf}_report_${dayHr}.out
     echo
fi


if [[ $scope = all ]]
then
     echo
     ((i=0))
     while ((i<=nrfs))  # copy the streamX.out files
to audit naming standard
     do
        cp stream${i}.out m1s0${i}q.out
        ((i=i+1))
     done

     cp update_power.out m1s00rf.out
     cp update_throughput.out m1s01rf.out

     echo "Copied *.out files to audit naming
standard"
     echo
     echo "Copy all audit required files to audit dir"
```

```
     echo
     echo "FINISHED Run1  "
     echo

#
-------------------------------------------------------
-------------
     echo
     echo STARTING RUN 2
     rm *.lock

     echo "Start Run 2 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

     show_asiq_cpu  start_power  > start_power_cpu.out
     show_asiq_cpu  start_refresh1  >
start_refresh1_cpu.out
     echo
     echo "     Start RF1 in the background   `date` "
     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
        sleep 10
     done

     show_asiq_cpu  end_refresh1  >
end_refresh1_cpu.out

     # RF1 has completed and RF2 is waiting on the
Query Stream to Complete

     show_asiq_cpu  start_stream0
>start_stream0_cpu.out
     echo
     echo "     Start Query Stream   `date` "
     dbisqlc -c "DSN=tpch" -q stream0.sql >
stream0.out

     echo
     echo "     Finish Query Stream   `date` "
     show_asiq_cpu  end_stream0  >end_stream0_cpu.out

###  Finished Query Stream

     # 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

     show_asiq_cpu  start_refresh2  >
start_refresh2_cpu.out
     echo
     echo "     Start RF 2   `date` "
     wait $rfspid

     echo "     End RF 2    `date`  "
     show_asiq_cpu  end_refresh2  >
end_refresh2_cpu.out
     show_asiq_cpu  end_power    >end_power_cpu.out

     echo
     echo "End Run 2 Power  Test    `date` "
     echo "----------------------          "
     echo
     echo
     ps -eaf | grep asiq| grep -v grep | tr -s ' ' |
cut -d ' ' -f8,9
     echo
     show_asiq_cpu  start_throughput
>start_throughput_cpu.out
     echo
     echo "Start Run 1 Throughput `date`  "
     echo "----------------------         "
     echo "  "
     echo "     Start Query Streams  `date` "
     echo "  "

     ((i=1))
     while ((i<=nrfs))  # run all query streams
concurrently
     do
        dbisqlc -c "DSN=tpch" -q stream${i}.sql >
stream${i}.out &
        ((i=i+1))
     done

     echo Start the refresh streams  `date`
     echo

     dbisqlc -c "DSN=tpch" -q update_throughput$
{nrfs}.sql > update_throughput.out &
     rf0pid=$!

     echo
     wait $rf0pid
     echo "   All refresh streams have completed"
`date`

     wait  # for everything
     echo "   All query and refresh streams have
completed"  `date`

     echo
     echo "End Run 2 Throughput `date`  "
     echo "--------------------          "
     echo
     ps -eaf | grep asiq | grep -v grep | tr -s ' ' |
cut -d ' ' -f8,9
     echo
     show_asiq_cpu  end_throughput
>end_throughput_cpu.out
###

     dayHr=`date '+%m%d%H'`
     echo
     echo "Producing mrun2_${scope}_${sf}_report_$
{dayHr}.out "
     tpch_report.bash $scope $sf $mpssheap $mpssstack
run1_${scope}_${day} $seed $nrfs $nd $ds $sc \

> mrun2_${scope}_${sf}_report_${dayHr}.out
     echo

     ((i=0))
     while ((i<=nrfs))  # copy the streamX.out files
to audit naming standard
     do
        cp stream${i}.out m2s0${i}q.out
        ((i=i+1))
     done

     cp update_power.out m2s00rf.out
     cp update_throughput.out m2s01rf.out

     echo    "copied *.out files to audit naming
standard"
     echo
     echo

     dbisqlc -c "DSN=tpch" -q dbtables-syb.sql >
rdbtablest_end.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
```

```
     dbisqlc -q -c "DSN=tpch" check_options.sql >
check_options_end.out

     echo "FINISHED Run2 "
fi
```

==========

tpch_wait.sql

```
if exists (select 1
           from SYS.SYSPROCEDURE
           where proc_name = 'tpch_wait') then
    DROP procedure tpch_wait;
end if
;

-- Script to put a dealy between TPCH updates.
-- Normally we just want to sleep a bit to spread
updates out
-- through the entire throughput test. Sometimes we
run out of
-- space; if so, just wait some more...
create procedure tpch_wait()
begin

  declare local temporary table t_iq_spaceused(
    mainKB     unsigned bigint,
    mainKBUsed unsigned bigInt,
    tempKB     unsigned bigInt,
    tempKBUsed unsigned bigInt,
    )
    in SYSTEM on commit preserve rows;

  declare maintotal unsigned bigint;
  declare mainused unsigned bigint;
  declare temptotal unsigned bigint;
  declare tempused unsigned bigint;
  declare mainfree unsigned bigint;
  declare command varchar(255);

  select 'xp_cmdshell ''sleep 120''' into command;

  waitloop:
  LOOP
        truncate table t_iq_spaceused;
        execute immediate
             'iq utilities main into t_iq_spaceused
command statistics 30000' ;

        select          mainKB,
                        mainKBUsed,
                        tempKB,
                        tempKBUsed
          into maintotal, mainused, temptotal,
tempused
          from t_iq_spaceused;

        message 'TPCH main total: ',maintotal,'  main
used : ',mainused;
        message 'TPCH temp total: ',temptotal,'  temp
used : ',tempused;
        set mainfree = maintotal-mainused;
        message 'TPCH main free : ',mainfree;

        if ( mainfree > 130000000 )
                then leave waitloop;
                end if;

        select 'xp_cmdshell ''sleep 300''' into
command;
        execute immediate command;

  END LOOP waitloop;

  drop table t_iq_spaceused;
  commit;
end
;
```

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:

Andrew Neugebauer
1-925-236-6304
andrewn@sybase.com

For Sun pricing please contact:

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

Company Sun Microsystems
Contact Richard Gostanian
Phone 781-442-3063
Fax
Address 1 Network Drive, Burlington MA 01803

Quotation for Software and Support

SYBASE Sales lHollie Nash
Phone: 972-687-6412
Fax: 972-687-6409

CBSS#

	Catalogue Number	Product Description	License Type	Machine	P/S	List Price Per Unit	Quantity	Discount	Extended Price	Extended Support Fees
										3 YEARS
1	12841	Sybase IQ Single App Svr, per cpu core	CP	Sun	P	2,595	8		20,760.00	
3	98480	3 yr support Single App Svr, per cpu core				1,557	8			12,456.00
4		Discount (5% if total license + support dollar value > 25000)						-1660.80		
5										
6										
7										
8										
9										
10										
11										
12										

Quote Date: ***2/11/05***
Valid thru:

Total 31,555.20

Licence + 3 year support

Payment terms : Net 30 Days

5400 LBJ Freeway, Suite 1500, Dallas, TX 75240

Sales Quotation

Quote Number: T-US-659890-A

Quote Date: 2/8/05

Customer :

Tel / Fax : /

Sun : Sherri La Creta
Sun Microsystems, Inc.
1 Network Drive
Burlington MA 01803
Tel / Fax : 800-786-0404 opt. 6/781-442-8570

We are pleased to quote as follows:

Validity Period	Credit Terms	Shipping Terms
60 Days	Net 30 Days	Origin

Item	Product Number	Description	Qty	Unit List Price	Disc	Unit Net Price	Extended Net Price
1	A52-JN64C232GTB	SFV490, 4x1.35GHz, 32GB, 2x146GB HDD, DVD, S10, JES3	1	$75,995.00	17.00%	$63,075.85	$63,075.85
2	X320A	NO AMERICA/ASIA 220 PWR CRD KT	2	N/C	N/A	N/C	N/C
3	W9D-A52-3G	3 Years Gold Service for V490	1	$10,332.00	20.00%	$8,265.60	$8,265.60
4	SG-XPCI2SCSI-LM320	Sun StorEdge PCI dual channel Ultra320 differential SCSI host bus adapter	3	$510.00	30.00%	$357.00	$1,071.00
5	X1138A	Cable, SCSI, VHDCI/VHDCI, 2m	6	$95.00	15.00%	$80.75	$484.50

List Price Total:	$88,427.00

Total:	$72,896.95

YOU MUST READ THE FOLLOWING: THIS SUN QUOTATION AND ANY ORDER YOU SUBMIT FOR PRODUCTS OR SERVICES IS SUBJECT TO: (1) THE TERMS OF ANY EXISTING SALES AGREEMENT YOU HAVE WITH SUN GOVERNING THAT PRODUCT OR SERVICE, OR,IF NONE, BY SUN'S SALES TERMS FOUND AT http://www.sun.com/sales/salesterms, THE GENERAL TERMS OF WHICH ARE EITHER ATTACHED OR ON THE REVERSE SIDE HEREOF, AND (2) APPLICABLE SUN SERVICE LISTINGS AND STATEMENTS OF WORK FOUND AT http://www.sun.com/service/servicelist [(1) AND (2) COLLECTIVELY BEING CALLED "SUN SALES TERMS."]

ALL ORDERS MUST REFERENCE EITHER YOUR SALES AGREEMENT NUMBER OR THIS SALES QUOTATION AND BE IN CONFORMANCE WTIH SUN SALES TERMS. ORDERS ARE SUBJECT TO ACCEPTANCE BY SUN EITHER THROUGH ISSUANCE OF AN ORDER ACKNOWLEDGEMENT OR DELIVERY OF THE PRODUCTS OR SERVICES. THIS QUOTATION REMAINS FIRM FOR THE PERIOD LISTED ABOVE, EXCEPT THAT SUN MAY MODIFY THIS SALES QUOTATION IF THERE IS A TYPOGRAPHICAL ERROR OR THE AVAILABILITY OF PRODUCTS, SERVICES, OR CREDIT CHANGE.

CONTINENTAL RESOURCES, INC.

From: David Mancusi
Continental Resources, Inc.
175 Middlesex Turnpike
Bedford, MA 01730
Ph: (781) 533-0454
Fax: (781) 533-0395

To: Richard Gostanian
Sun Microsystems
Burlington, MA

QTY	Part#	Description	List $	Unit $	Extended $
		Sun Microsystems			
3	XTA3310R01A0R876	Sun StorEdge 3310 SCSI Array,RR,876GB(12x73GB 10K RPM disks),Ultra160 SCSI-JBOD, and 2 ACpower supplies	$11,995	$8,771	$26,313
3	X311L	North American Power Cord	$0	$0	$0
3	W9D-SE3310-3G	Service Uplift to 3 Year Gold Support	$4,356	$4,051	$12,153
				Total:	**$38,466**

Validity of 60 Days As of 2/11/05

TERMS: Net 30; and subject to credit review.
Pricing subject to change. Delivery is conveyed F.O.B. shipping point. Title passes upon payment in full.
Risk of loss is FOB shipping point. Payment of Freight Insurance modifies Risk of Loss to FOB destination. Taxes, be included in above quote. Return rights are restricted to vendor or manufacturers policy in existence at time of return.
Professional services requires a separate schedule of work or service agreement. Third party leases must be identified before shipment and Lessor
must be judged credit worthy by CRI. All payments are in United States Dollars.