TPC Benchmark™ H
Full Disclosure Report

Sun Microsystems Sun Fire™ X4270
Using Sybase IQ 15.1 ESD#1 Single Application Server

Submitted for Review
Report Date: Dec 4, 2009

TPC Benchmark H Full Disclosure Report

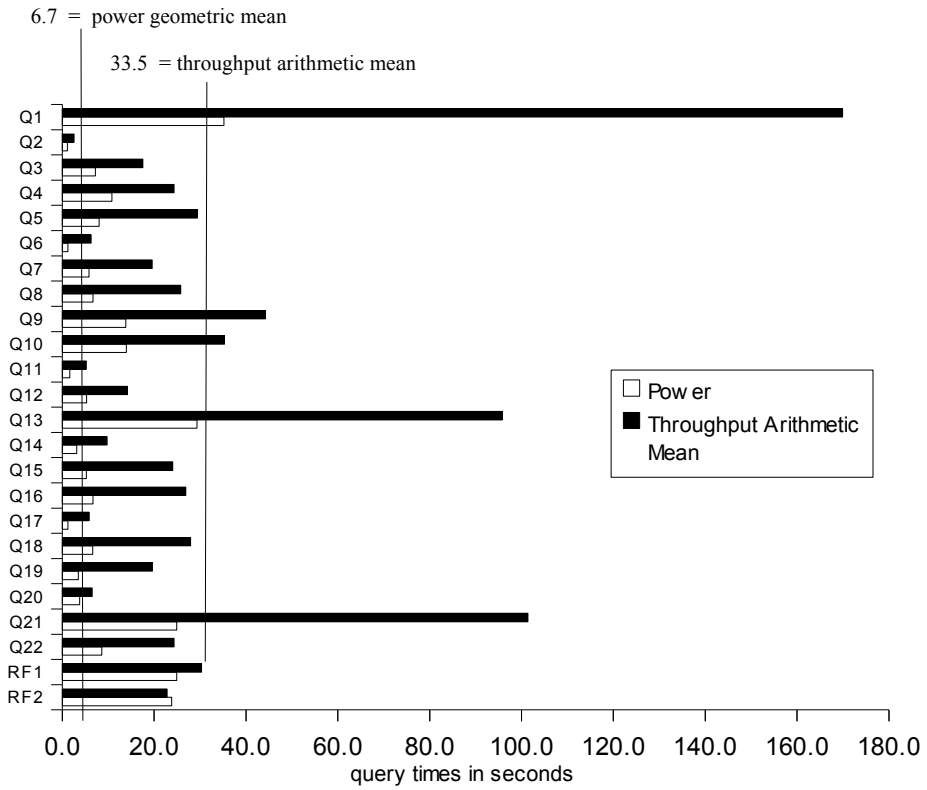
First Printing

1 Sybase Drive , Dublin, CA 94568 U.S.A

Sun microsystems / SYBASE	**Sun Fire™ X4270 with Sybase IQ 15.1 ESD#1 Single Application Server**	**TPC-H Rev. 2.8.0** **Report Date: Dec 4, 2009**

Total System Cost	Composite Query per Hour Metric	Price/Performance
$61,217.99	**53,501.6** QphH@100GB	**$1.14** per QphH@100GB

Database Size	Database Manager	Operating System	Other Software	Availability Date
100GB	**Sybase IQ 15.1 ESD#1 Single Application Server**	**Solaris 10**	**Solaris Volume Manager**	**Dec 4, 2009**



Database Load Time = 25 min 2 secs	Load Includes Backup: N	Total Storage/Database Size= 3.74

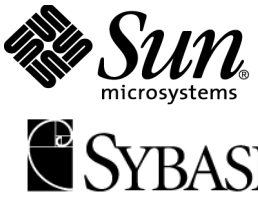
RAID (Base tables): RAID 1	RAID (Base tables and auxiliary data structures): RAID 1	RAID (All): N

System Configuration:

SunFire X4270 with
- 2 Intel Xeon 5570 2.93 GHz Quad Core processors
- 120 GB memory
- 1x 146GB SAS internal disk (10K RPM)
- 8 x 32GB internal SSDs

Total Storage: 374.4 GB

1 GB = 1024*1024*1024 bytes



Sun Fire™ X4270 with Sybase IQ 15.1 ESD#1 Single Application Server

TPC-H Rev. 2.8.0

Report Date: Dec 4, 2009

Description	Part Num	Source	Unit Price	Qty	Ext. Price	3 Yr. maint.
Server Hardware						
SunFire X4270, 2x2.93GHz xeon 5570 CPUs, 120 GB mem, 8 x 32 GB SSDs, 1 x 146 GB SAS disk	7960173	1	33,038.00	1	33,038.00	
3 Year Gold Warranty Upgrade – X4270Server	X4270-S1-AA	1	844.2	1		844.20
Sun Server Discount					-5,946.84	
TechRack 6U rack	TR-06U-30	3	349.00	1	349.00	
ASUS VW193TR 19" LCD monitor	N82E16824236069	4	119.99	1	119.99	
1year support for monitor	N82E16824236069	4	16.99	3		50.97
LITE-ON SK-1688U USB keyboard(1 + 2 spares)	N82E16823107128	4	6.99	3	20.97	
Server Hardware Subtotal					27,581.12	895.17
External Storage						
None						
Server Software						
Sybase IQ 15.1 Single App Svr – per cpu core	12193	2	2,595.00	8	20,760.00	
Sybase IQ 3 Years Extended Support 24 x 7	98477	2	1,713	8		13,704.00
Sybase discount					1,038.00	685.00
Server Software Subtotal					19,722.00	13,018.80
			Total		47,303.12	13,913.97
			3 Yr. Cost		61,217.09	
			QphH@ 100G		53,501.6	
			$/QphH@ 100(		1.14	

Service for Sun products is from Sun Microsystems, Inc.

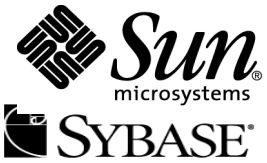
Service for Sybase products is from Sybase Inc.

Notes (Source):

1. Sun Microsystems Inc.
2. Sybase Inc.
3. http://www.techrack.com/catalog/Desktop_Enclosure.html
4. http://www.newegg.com

PriceQuotes provided in Appendix G

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



Sun Fire™ X4270 with Sybase IQ 15.1 ESD#1 Single Application Server

TPC-H Rev. 2.8.0

Report Date: Dec 4, 2009

Numerical Quantities

Measurement Results:

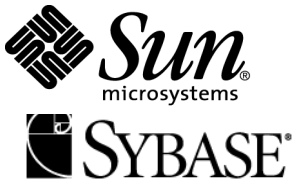
Database Scale Factor	= 100GB
Total Data Storage / Database Size	= 3.74
Start of database load time	= 2009-11-04 15:43:40
End of database load time	= 2009-11-04 16:08:42
Database Load Time	= 25mins:2secs
Query Streams for Throughput Test	= 5
TPC-H Power	= 53,923.4
TPC-H Throughput	= 53,083.1
TPC-H Composite Query-per-Hour Rating (QphH@100GB)	= 53,501.6
Total System Price Over 3 Years ($US)	= 61,217.99
TPC-H Price/Performance Metric ($/QphH@100GB)	= 1.14

Measurement Intervals:

Measurement Interval in Throughput Test (Ts)	= 746 seconds

Duration of Stream Execution:

Stream ID	Seed	Start Date	Start Time	End Date	End Time	Duration
Stream 00	1104160842	4-Nov-09	16:08:45	4-Nov-09	16:12:52	0:04:07
Stream 01	1104160843	4-Nov-09	16:12:52	4-Nov-09	16:25:05	0:12:12
Stream 02	1104160844	4-Nov-09	16:12:52	4-Nov-09	16:24:59	0:12:07
Stream 03	1104160845	4-Nov-09	16:12:52	4-Nov-09	16:25:05	0:12:12
Stream 04	1104160846	4-Nov-09	16:12:52	4-Nov-09	16:25:17	0:12:25
Stream 05	1104160847	4-Nov-09	16:12:52	4-Nov-09	16:25:18	0:12:26



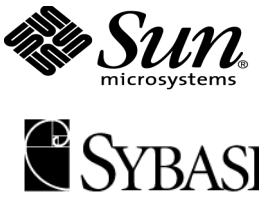
Sun Fire™ X4270 with Sybase IQ 15.1 ESD#1 Single Application Server

TPC-H Rev. 2.8.0

Report Date: Dec 4, 2009

TPC-H Timing Intervals (in seconds)

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Stream0	35.2	1.1	7.2	10.8	8.0	1.2	5.8	6.7	13.8	13.9	1.6	5.3
Stream1	155.5	3.0	26.0	22.3	22.2	6.8	20.1	17.3	37.1	32.6	5.9	13.1
Stream2	165.1	1.9	14.6	19.4	26.8	14.9	18.2	24.3	60.2	37.3	3.5	14.9
Stream3	167.6	2.8	15.7	26.6	33.4	2.2	23.2	48.4	41.9	34.9	5.2	14.2
Stream4	186.0	3.4	10.9	25.6	49.0	3.6	19.2	24.9	47.0	40.0	4.8	12.6
Stream5	175.4	1.5	20.3	27.7	15.7	3.7	17.1	13.9	35.0	31.8	6.7	15.9
min	155.5	1.5	10.9	19.4	15.7	2.2	17.1	13.9	35.0	31.8	3.5	12.6
avg	169.9	2.5	17.5	24.3	29.4	6.2	19.6	25.8	44.2	35.3	5.2	14.1
max	186.0	3.4	26.0	27.7	49.0	14.9	23.2	48.4	60.2	40.0	6.7	15.9



Sun Fire™ X4270 with Sybase IQ 15.1 ESD#1 Single Application Server

TPC-H Rev. 2.8.0

Report Date: Dec 4, 2009

	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream0	23.3	3.1	5.2	6.7	1.2	6.6	3.5	3.8	24.9	8.6	24.9	23.8
Stream1	74.6	8.1	29.7	24.0	3.4	37.5	26.4	10.8	129.3	26.6	31.1	22.9
Stream2	111.8	7.5	19.1	37.7	17.5	23.5	13.9	3.7	72.6	18.1	30.1	23.0
Stream3	106.4	10.1	16.6	21.3	2.3	37.2	16.8	5.3	80.7	19.5	31.2	23.0
Stream4	110.0	8.7	21.5	24.8	2.2	16.7	16.2	2.8	98.3	16.4	33.5	22.8
Stream5	76.3	14.4	33.3	26.6	3.8	24.7	24.9	9.8	126.2	40.9	25.5	22.4
min	74.6	7.5	16.6	21.3	2.2	16.7	13.9	2.8	72.6	16.4	25.5	22.4
avg	95.8	9.8	24.0	26.9	5.8	27.9	19.6	6.5	101.4	24.3	30.3	22.8
max	111.8	14.4	33.3	37.7	17.5	37.5	26.4	10.8	129.3	40.9	33.5	23.0

Table of Contents

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

Benchmark Sponsors: Peter Thawley
Senior Director
Sybase, Inc.
1 Sybase Drive
Dublin, CA 94568

Dec 3, 2009

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

Platform:	**Sun Fire X4275 Server**
Database Manager:	**Sybase IQ 15. ESD#1 Single Application Server**
Operating System:	**Solaris 10**

The results were:

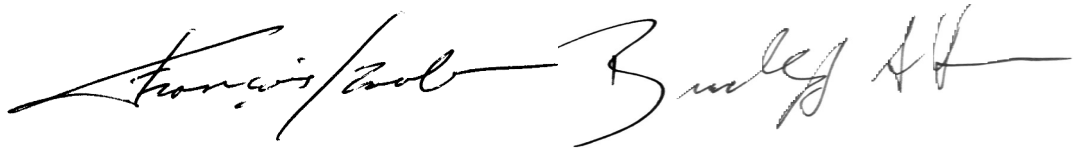
CPU (Speed)	Memory	Disks	**QphH@100GB**
1 (one) Sun Fire X4275 with:			
2 x Xeon 5570 (2.93 GHz)	120 GB Main	8 x 32 GB (int.) 1 x 146 GB (int)	**53501.6**

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

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

- The database load time was correctly measured and reported
- The required ACID properties were verified and met
- The query input variables were generated by QGEN
- The query text was produced using minor modifications
- The execution of the queries against the SF1 database produced compliant answers
- A compliant implementation specific layer was used to drive the tests
- The throughput tests involved 5 query streams
- The ratio between the longest and the shortest query was such that no query timing was adjusted
- The execution times for queries and refresh functions were correctly measured and reported
- The repeatability of the measured results was verified
- The required amount of database log was configured
- The system pricing was verified for major components and maintenance
- The major pages from the FDR were verified for accuracy

Respectfully Yours,



Francois Raab, President

Bradley J. Askins, Auditor

1. General Items

1.1 Benchmark Sponsor

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

Sybase Inc. is the sponsor 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.

Configuration (Priced and Measured are identical):

SUN Fire X4270 Server with

2 X 2.93GHz Intel Xeon 5570 processors

120 GB Memory

8 x 32 GB internal SSDs

1 x 146 GB 10K SAS internal disk

No External Storage

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 2.8 of DBGEN and QGEN were used for this TPC-H benchmark.

3.3 Generating Values for Substitution Parameters

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

The supplied QGEN version 2.8 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 2 (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 seven 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 run for the same O_KEY used in step 1 and returned the values associated with O_KEY just before the ACID transaction began..
3. The ACID Transaction was resumed and COMMITTED.

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 run on 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.

4.4.3 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 COMMIT..
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, indexes and the catalog file are on RAID1 volumes. When one of these disks was removed from the server during the durability test, Sybase IQ continued to run as if nothing happened until the (non-mirrored) IQ temp segment on that disk was referenced. At that time IQ crashed, as expected. The IQ was then restarted and the durability success file and the HISTORY table were successfully compared.

4.5.2 System Crash

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

The system crash and memory failure tests were combined by cutting off power to the X4270 server during the durability test. When power was restored, the system rebooted and the database was restarted. The durability success file and the HISTORY table were compared successfully.

4.5.3 Memory Failure

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

See section 4.5.2.

5. Clause 4 Scaling and Database Population

5.1 Ending Cardinality of Tables

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

Table	*Rows*
Lineitem	600037902
orders	*150000000*
partsupp	*80000000*
Part	*20000000*
Customer	*15000000*
Supplier	*1000000*
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 4 RAID 1 volumes. Each volume was constructed from raw partitions on each of two internal SSDs using the Solaris Volume Manager.
- The Temp database for Sybase IQ was configured using 8 raw partitions, one on each of the internal SSDs disks. The Temp database was not mirrored.

All the details for configuring the storage used for the tables, logs, temp database, swap space, etc. are provided 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 tpch log file also resided on a RAID 1 device.

5.5 Modifications to the DBGEN

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

The supplied DBGEN version 2.8 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 =25 min 2 secs

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.

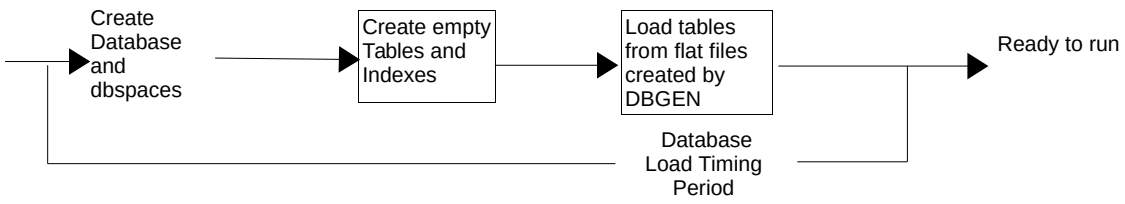
The data storage ratio is computed from the following information:

Disk Type	# Of Disks	Space Per Disk*	Sub-Total Disk Space**
internal	1	146 GB	135.97 GB
internal	8	32 GB	238.24 GB
		Total Space	**374.21 GB**
		Data Storage Ratio	**3.74**

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

5.8 Database Load Mechanism Details and Illustration

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



The test database was loaded using flat files. All load scripts are included in Appendix B.

5.9 Qualification Database Configuration

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

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

5 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:

Refresh Function	Start Date	Start Time	End Date	End Time
RF1	4-Nov-09	16:12:53	4-Nov-09	16:13:24
RF2	4-Nov-09	16:13:24	4-Nov-09	16:13:47
RF1	4-Nov-09	16:13:47	4-Nov-09	16:14:17
RF2	4-Nov-09	16:14:47	4-Nov-09	16:15:10
RF1	4-Nov-09	16:15:10	4-Nov-09	16:15:42
RF2	4-Nov-09	16:15:42	4-Nov-09	16:16:05
RF1	4-Nov-09	16:16:35	4-Nov-09	16:17:08
RF2	4-Nov-09	16:17:38	4-Nov-09	16:18:01
RF1	4-Nov-09	16:18:01	4-Nov-09	16:18:27
RF2	4-Nov-09	16:18:57	4-Nov-09	16:19:19

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:

RunID	QppH@100GB	QthH@100GB	QphH@100GB
Run1	53,923.4	53,083.1	53,501.6
Run2	56,009.2	51,968.5	53,951.0
%Difference	3.72%	-2.14%	-0.40%

6.11 System Activity Between Performance Tests

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

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

7. Clause 6 SUT and Driver Implementation

7.1 Driver

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

The entire test is run by executing the ntest15 shellscript as follows

ntest15 all 100 5 8 0 0 1

The text of ntest15 is provided in Appendix E and the texts of all the scripts invoked by ntest15 are provided in Appendix B.

The query streams within the power and throughput tests are generated by a script called gen_streams_new.ksh which uses qgen to generate the query stream files.

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 a 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. ACID test scripts utilizing dbtest can be found 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.

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 $61,217.99. 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. 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 available as of Dec 4, 2009.

9. Auditor's Information and Attestation Letter

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

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

Appendix A. Solaris 10 and Sybase IQ 15.1 ESD#1 Parameters

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

==========

Sybase IQ Server Configuration Parameters

==========

utility.cfg

```
-n bur93-212
-x tcpip{port=2638}
-c 32m
-gd all
# -gl all
-gm 15
-gp 4096
# -ti 4400
-tl 300
-iqtc 128
-iqmc 128
```

tpch.cfg

```
-n bur93-212
-x tcpip{port=3002}
-c 12m
-gd all
-gl all
-gm 8
-gp 4096
-ti 4350
-tl 300
-iqtss 250
#-iqtss 500 default 220 too small for 100gb best 87000 + 31400 with 20 mb left over
-iqmc 86950
#-iqmc 20200
-iqtc 31450
-iqmt 500
-iqgovern 8
#-iqgovern 12 orig
-iqpartition 8
#-iqpartition 2 orig
#-iqnumbercpus 14
#-iqwmem 70000 should be sum of iqmc and iqtc
```

==========

Sybase IQ Database Options

(altered from default)

==========

options.sql

==========

```
-------- configuration options ----------

SET OPTION PUBLIC.default_dbspace = 'user1';

SET OPTION PUBLIC.Allow_Nulls_By_Default='Off';
SET OPTION PUBLIC.Append_Load='On';
SET OPTION PUBLIC.Force_No_Scroll_Cursors='On';
SET OPTION PUBLIC.Load_Memory_Mb=0;
SET OPTION PUBLIC.Max_IQ_Threads_Per_Connection=100;
SET OPTION PUBLIC.Minimize_Storage='On';
SET OPTION PUBLIC.Notify_Modulus=10000000;
SET OPTION PUBLIC.Query_Temp_Space_Limit=0;
SET OPTION PUBLIC.Row_Counts='On';
SET OPTION PUBLIC.Hash_Thrashing_Percent=100;
SET OPTION PUBLIC.SignificantDigitsForDoubleEquality=15;
SET OPTION PUBLIC.Garray_Fill_Factor_Percent=1;

------- performance options ----------

SET OPTION PUBLIC.Max_Hash_Rows = 9200000; -- for 100GB seems 91 is better than 10
SET OPTION PUBLIC.Default_Having_Selectivity_PPM = 10;
SET OPTION PUBLIC.index_preference=8;
SET OPTION PUBLIC.subquery_flattening_preference = 1; -- q22
SET OPTION PUBLIC.fp_prefetch_size = 100;
SET OPTION PUBLIC.row_prefetch_size = 40 ; -- for several queries default = 40 -- go even higher
SET OPTION PUBLIC.Prefetch_Threads_Percent=; -- this is the default
SET OPTION PUBLIC.STRING_RTRUNCATION='Off';
------- statistics options --------`
SET OPTION PUBLIC.Query_Plan='on';
SET OPTION PUBLIC.Query_Plan_As_Html='on';
SET OPTION PUBLIC.Query_Detail='on';
SET OPTION PUBLIC.Query_Timing='on';
SET OPTION PUBLIC.Query_Plan_After_Run='on';
```

==========

==========

Sybase IQ Environment Variables

==========

```
SYBASE_JRE6_64=/export/home/sybase/shared/JRE-6_0_7
OS_CFG=SunAMD64
HZ=100
SHELL=/usr/bin/bash
TERM=ansi
OLDPWD=/export/home/sybase
LD_LIBRARY_PATH=/export/home/sybase/IQ-15_1/lib64:
SYBROOT=/export/home/sybase
SYBASE=/export/home/sybase
MAIL=/var/mail/sybase
PATH=/export/home/sybase/IQ-
15_1/bin64:/usr/bin:.:/bin:/usr/bin:/usr/sbin:/export
/home/sybase/run/scripts:/usr/sfw/bin
PWD=/export/home/sybase/run/scripts
TZ=US/Eastern
IQDIR15=/export/home/sybase/IQ-15_1
SHLVL=1
HOME=/export/home/sybase
LOGNAME=sybase
_=/usr/bin/env
```

==========

.odbc.ini

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

[tpch]
Driver=/export/home/sybase/IQ-15_1/lib64/dbodbc11_r.so
EngineName=bur93-212
CommLinks=tcpip{host=129.148.93.212;Port=3002}
DatabaseName=tpch
UserID=DBA
Password=sql
DBG=yes
LOG=/tmp/tpch_odbc.log

[utility_db]
EngineName=bur93-212
```

```
CommLinks=tcpip{host=129.148.93.212;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
set lotsfree = 4096
set bufhwm = 10000
```

Appendix B. Programs and Scripts

==========

create_database.sql

```
create database '/sybase2/tpch.db'
collation 'ISO_BINENG'
case respect
page size 4096
blank padding on
java off
iq path '/sybase2/firstmain' iq size 1000
temporary path '/sybase2/T01'
iq page size 524288;
```

==========

add_main_files.sql

==========

```
create dbspace user1 using
FILE main01 '/sybase2/M01',
FILE main02 '/sybase2/M02',
FILE main03 '/sybase2/M03',
FILE main04 '/sybase2/M04';
```

==========

add_temp_files.sql

```
alter dbspace IQ_SYSTEM_TEMP ADD
FILE temp02 '/sybase2/T02',
FILE temp03 '/sybase2/T03',
FILE temp04 '/sybase2/T04',
FILE temp05 '/sybase2/T05',
FILE temp06 '/sybase2/T06',
FILE temp07 '/sybase2/T07',
FILE temp08 '/sybase2/T08';
```

==========

create_tables_int.sql

==========

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


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


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


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


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


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


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


CREATE TABLE lineitem
  (
    l_orderkey           unsigned int,    /*  */
    l_partkey            unsigned int,
    l_suppkey            unsigned int,
    l_linenumber         int,
    l_quantity           double precision,
    l_extendedprice      double precision,
    l_discount           double precision,
    l_tax                double precision,
    l_returnflag         char(1),
    l_linestatus         char(1),
    l_shipdate           date,
```

```
  l_commitdate          date,
  l_receiptdate         date,
  l_shipinstruct        char(25),
  l_shipmode            char(10),
  l_comment             varchar(44)
);

CREATE HG INDEX l_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_int.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 s_cmd varchar(128);
  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(16,5);
  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(16,5))/1000;
  SET s_cmd='Stream updates Update update_'+c_stream+'_RF1 LENGTH --
'+cast(n_seconds AS varchar(20))+ ' seconds' ;
  SELECT s_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 s_cmd varchar(128);
  DECLARE outfilename varchar(128); -- Debug
  DECLARE c_lf varchar(2);
  DECLARE t_qstart timestamp;
  DECLARE t_qstop timestamp;
  DECLARE n_seconds numeric(16,5);
  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
     WHERE l_orderkey in (select d_orderkey from  #delete_table);
  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
     WHERE o_orderkey in (select d_orderkey from  #delete_table);
  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(16,5))/1000;
  SET s_cmd='Stream updates Update update_'+c_stream+'_RF2 LENGTH --
'+cast(n_seconds as varchar(20))+ ' seconds' ;
  SELECT s_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
  'dbgen_files/orders.tbl.1',
  'dbgen_files/orders.tbl.2',
  'dbgen_files/orders.tbl.3',
  'dbgen_files/orders.tbl.4',
  'dbgen_files/orders.tbl.5',
  'dbgen_files/orders.tbl.6',
  'dbgen_files/orders.tbl.7',
  'dbgen_files/orders.tbl.8'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
```

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

load_lineitem.sql

```
================================================
LOAD TABLE ORDERS (
O_ORDERKEY                              '|',
O_CUSTKEY                               '|',
O_ORDERSTATUS                           '|',
O_TOTALPRICE                            '|',
O_ORDERDATE                             '|',
O_ORDERPRIORITY                         '|',
O_CLERK                         '|',
O_SHIPPRIORITY                          '|',
O_COMMENT                               '|'
)
FROM
  'dbgen_files/orders.tbl.1',
  'dbgen_files/orders.tbl.2',
  'dbgen_files/orders.tbl.3',
  'dbgen_files/orders.tbl.4',
  'dbgen_files/orders.tbl.5',
  'dbgen_files/orders.tbl.6',
  'dbgen_files/orders.tbl.7',
  'dbgen_files/orders.tbl.8'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;

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

update_power.sql

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

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

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

update_throughput5.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='dbgen_files/';
select 'Stream updates START -- ', qstart ;
select @@servername, db_name();

xp_cmdshell 'sleep 0';
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;

set qstop = now(*);
select 'Stream updates STOP -- ', qstop ;
================================================
```

gen_streams_new.ksh

```
================================================
#!/bin/ksh

if (( $# < 2 ))
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/cc
s/bin:/usr/dt/bin:/usr/dist/pkgs/devpro,v4.0/5.x-
sparc/bin:/usr/dist/local/exe:/usr/dist/exe:/usr/ucb:/usr/sbin:/net/josie/export/hom
e18/rgostan/bin:/export/home/sybase/run/scripts:/etc:.:/export/home/sybase/run/t
pch/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/templates/queries;
#export
DSS_QUERY=/export/home/sybase/run/tpch/appendix/templates/queries.debug;

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

i=0

while ((i<=ns))
do
  qgen -c -p $i -l qparm${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))
done

((last_seed=seed-1))
```

==========

ACID Test Execution Code

==========

==========

atomicity test

==========

```
#!/bin/ksh

cd $ACID_ROOT/atomicity
dbtest $ACID_ROOT/atomicity/acid_atomic_main.tst >
$ACID_RESULTS/acid_atomic_main.out

roll_back=1

rm -f $ACID_RESULTS/atomc $ACID_RESULTS/atomr

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

==========

consistency test

==========

```
#!/bin/ksh
cd $ACID_ROOT/consistency/
initial_size=`cat /sybase2/tpch.iqmsg|wc -l`

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

final_size=`cat /sybase2/tpch.iqmsg|wc -l`

lines_during_test=`expr $final_size - $initial_size`

tail -$lines_during_test /sybase2/tpch.iqmsg |grep -i chk>
$ACID_RESULTS/consckpt.`date '+%y%m%d_%H%M%S'`
```

==========

isolation tests

==========

```
#!/bin/ksh


export ACID_ROOT=$HOME//run/scripts/acid15/acid_scripts/acid_scripts
export
ACID_RESULTS=$HOME//run/scripts/acid15/acid_scripts/acid_scripts/isolatio
n


ps
echo
echo starting isolation_1
cd $ACID_ROOT/isolation/isolation_1
dbtest acid_isolation_main1.tst > $ACID_RESULTS/iso1.`date '+%y%m%d_
%H%M%S'`
echo finished isolation_1
sleep 20

ps
echo
echo starting isolation_2
cd $ACID_ROOT/isolation/isolation_2
dbtest acid_isolation_main2.tst > $ACID_RESULTS/iso2.`date '+%y%m%d_
%H%M%S'`
echo finished isolation_2
sleep 20

ps
echo
echo starting isolation_3
cd $ACID_ROOT/isolation/isolation_3
dbtest acid_isolation_main3.tst > $ACID_RESULTS/iso3.`date '+%y%m%d_
%H%M%S'`
echo finished isolation_3
sleep 40

ps
echo
echo starting isolation_4
cd $ACID_ROOT/isolation/isolation_4
dbtest acid_isolation_main4.tst > $ACID_RESULTS/iso4.`date '+%y%m%d_
%H%M%S'`
echo finished isolation_4
sleep 40

ps
echo
echo starting isolation_5
cd $ACID_ROOT/isolation/isolation_5
dbtest $ACID_ROOT/isolation/isolation_5/acid_isolation_main5.tst >
$ACID_RESULTS/iso5.`date '+%y%m%d_%H%M%S'`
echo finished isolation_5
sleep 20
```

```
ps
echo
echo starting isolation_6
cd $ACID_ROOT/isolation/isolation_6
dbtest $ACID_ROOT/isolation/isolation_6/acid_isolation_main6.tst >
$ACID_RESULTS/iso6.`date '+%y%m%d_%H%M%S'`
echo finished isolation_6
```

===

durability test

===

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

STREAM_COUNT=7

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

===

acid_atomic_main.tst

===

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

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

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

stringconnect "dsn=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 ' '

%include 'acid_atomic_setup.tst'
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


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

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

execute {select count(*)
     from history
     where h_o_key =^ and h_l_key =^}
     substitute ordr, line
     into total_history_count
execute {select o_totalprice, l_quantity, l_extendedprice
     from orders, lineitem
     where o_orderkey = l_orderkey and o_orderkey =^ and l_linenumber = ^}
     substitute ordr, line
     into o_total, l_quan, l_price
print       'Before Rolling back:'
print   'o_totalprice = ',o_total,'    l_quantity = ',l_quan,
     '      l_extendedprice = ',l_price

print   'Before Rollback History table count=',total_history_count


let counter = counter+1

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

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

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

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

print   'After Rollback History table count=',total_history_count

} ENDLOOP

commit

%
% Atomicity test with commit
%
stringconnect "dsn=tpch;"
```

```
print ' '
print 'Starting atomicity test with commit '
print ' '
%include 'acid_atomic_setup.tst'
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       'Initial values:'
print       'o_totalprice = ',o_total,'          l_quantity = ',l_quan,
            '           l_extendedprice = ',l_price
print       ''
print       ''
print   'Before Commit:'
print       '           l_extendedprice = ',l_price

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


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

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


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

print   'Before Commit History table count=',total_history_count

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

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

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

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

print ' '

} ENDLOOP

close cur1
commit

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

End Test
```

==========

acid_atomic_setup.tst

==========

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

stringconnect "dsn=tpch;"

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

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

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

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

print 'Number of rows before load: ',ROWS

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

commit

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

print 'Number of rows after load: ',ROWS

disconnect

End Test
```

==========

acid_consistency_main.tst

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

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

stringconnect "dsn=tpch;"

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

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

%run test 'acid_consistency_setup.tst'

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

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

let i = 1
LOOP {
    if i > 7 then { BREAK LOOP } endif
        let ot_file = "acid_consist_user", i, ".ot"
        let my_str = "stream=", i
        print ot_file, my_str
        start test '-o' ot_file my_str 'acid_consistency_txn.tst'
        sleep 7000
        let i = i + 1
} ENDLOOP


% synchronize 7

% let the log flush...   7*100*1000 = 700000
sleep 1000000
stringconnect "dsn=tpch;"
%include 'acid_consistency_query.tst'
run test '-o' 'acid_consistency_q2.ot' 'acid_consistency_query.tst'

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

End Test
```

===

acid_consistency_query.tst

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

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

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

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

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

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

disconnect

END Test
```

===

acid_consistency_setup.tst

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

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

stringconnect "dsn=tpch;"

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

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

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

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

print 'acid_table created'
execute {create table latest(stream int ,last int null) on SYSTEM }
LOOP ({let j = 1}; {j <= 7}; {let j = j + 1})
   {
          execute { insert into latest(stream,last) values (^,0) }
                substitute j
   } endloop
commit

print 'latest created'
```

```
LOOP ({let i = 1}; {i <= 7}; { 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 = 700
print 'Number of rows after load: ',ROWS

End Test
```

===

acid_consistency_txn.tst

===

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

stringconnect "dsn=tpch;"

execute {set temporary option chained='on'}

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

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

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

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

print   'After Commit:'
print   'o_totalprice = ',o_total,'     l_quantity = ',l_quan,
       '      l_extendedprice = ',l_price
print   'After Commit History table count=',total_history_count
print ' '
 execute { update latest set last=^ where stream=^ } substitute i,stream
 commit
 execute { set temporary option isolation_level=1 }
 execute { select max(last) from latest } into biggest
 execute { select min(last) from latest } into smallest
 commit
 execute { set temporary option isolation_level=3 }

 let num=1200*(i-smallest)
 if i+4>=biggest
 then {let num=num+7000}
 endif
--print 'user',stream,' = ',num
 sleep num
}
ENDLOOP
disconnect

End Test
```

===

acid_durability_main.tst

===

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

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

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

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

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

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

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

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

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

End Test
```

==========

acid_durability_query.tst

==========

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

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

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

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

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

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

 } ENDLOOP

disconnect

END Test
```

==========

acid_durability_setup.tst

==========

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

stringconnect "dsn=tpch;"

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

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

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

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

print 'acid_table created'

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

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

print 'latest created'

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

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

End Test
```

==========

acid_durability_txn.tst

==========

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

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 '
allow no error

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

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

 if i=101 then {
     let smallest=0
```

```
    allow error -210
            commit
    execute { set temporary option isolation_level=1 }
    execute { select min(last) from latest } into smallest
    execute { set temporary option isolation_level=3 }
    commit
    LOOP{
        if smallest >= 100 then {
                                print 'Stream ', stream,
                                        ' Entering the Second phase with
delays'
                                break loop}
                        endif
        let sleep_time =10
        sleep sleep_time
            commit
            execute { set temporary option isolation_level=1 }
        execute { select min(last) from latest } into smallest
            execute { set temporary option isolation_level=3 }
            commit
    }
    ENDLOOP
    allow no error
    let commit_delay=5}
 endif


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

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

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

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

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

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

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

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


End Test
```

==========

acid_functions.tst

==========

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Created By: David Walrath
% Create Date: 7/15/1999
%
% This script creates various functions used by the Acid tests.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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)
--in SYSTEM
}

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


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);
 DECLARE num            INT ;

 LOOP1: LOOP

 COMMIT;

 acid1:
 BEGIN ATOMIC

  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 message 'Completed ',o_key,' .....';
                END IF;
            END IF;
  END IF;

 END acid1;

-- if c_sqlstate = '00000'
-- then commit;
-- else rollback;
-- end if;

 if c_sqlstate = '00000'
 then LEAVE LOOP1;
 end if;

 select cast( rand()*4.5 as int) into num;
 message 'rollback sleep=', num,' sqlstate=',c_sqlstate;
 call dbo.sleep(num);

END LOOP LOOP1;

-- commit ;
 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
            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)) into
o_total
   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
```

```
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
```

```
print 'Generate_Ps_Values function created'
print ' '



print 'Creating Generate_acid_values2 function'

allow error -265
execute { DROP PROCEDURE generate_acid_values2 }
allow no error

execute{
create procedure generate_acid_values2(
in  streams         int,
in  txns      int
)

BEGIN

 declare seed                    int;
 declare rand_dbl       double precision;
 declare rand_int       int;
 declare i int;
 declare j 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();

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

 open random1;
 fetch next random1 into rand_dbl;

 set i=1;
 set j=1;
 loop1:
 while i < streams LOOP
   loop2:
    while j < txns LOOP

         insert into acid_table (stream,seqnum) values (i,j);

   end loop loop2;
 end loop loop1;
 commit;

 update acid_table
         set line=cast(rand(rowid(acid_table)+seed)*1500000+1 as int);
 commit;
 update acid_table
         set okey=o_orderkey
         from orders where o_orderkey=line;
 update acid_table
         set delta=cast(rand(line)*100+1 as int);
 update acid_table
         set line=max(l_linenumber)
         from lineitem where l_orderkey=ordr;
 commit;


END
}
commit

print 'Generate_acid_values function2 created'
print ' '
```

===

acid_isolation_main1.tst

===

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Created by: Masood Dirin
%  Created Date: 5/24/1999
%  Script name: tpcd_acid_isolation_main1.tst
%  ------------------------------------------------------------------------
%
%  Purpose of this test:
%  This test will run the first isolation test, which demonstrate
%  isolation for the read-write conflict of a read-write transaction
%  and a read-only transaction when the read-write transaction is commited.
%  Run the test as follow:
%
%  dbtest tpcd_acid_isolation_main1.tst > tpcd_acid_isolation_main1.ot
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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

stringconnect "dsn=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

===

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

%  Created Date: 5/24/1999

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

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

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

stringconnect "dsn=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

==

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

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

stringconnect "dsn=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

==

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

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

stringconnect "dsn=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

==

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


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

stringconnect "dsn=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

```
=====================================================
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Created by: Masood Dirin
%  Created Date: 5/27/1999
%  Script name: tpcd_acid_isolation_main6.tst
%  ------------------------------------------------------------------------
%  Run the test as follow:
%
%  dbtest -u tpcd_acid_isolation_main6.tst > tpcd_acid_isolation_main6.ot
%
%  Note:  -u switch will be used to archive the User1 query result
%  in a file named queryresult.cfr. This switch needs to be used each time
%  the test being run, since the query results will be different as the
%  results of the updates on the lineitem tables.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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

stringconnect "dsn=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

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

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

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

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

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

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

stringconnect "dsn=tpch;"
```

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

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

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

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

```
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

==

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

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

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

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

disconnect

END Test
```

==

acid_isolation_test2.tst

==

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Created by: Masood Dirin
%  Created Date: 5/24/1999
%  Script name: tpcd_acid_isolation_test1.tst
%
%  Part of tpcd_acid_isolation_main1.tst
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

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

```
stringconnect "dsn=tpch;"

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

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

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

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

End Test
```

==

acid_isolation_test2_query.tst

==

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

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

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

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

```
print 'User 2 completed query = ', times

disconnect

END Test
```

==

acid_isolation_test3_transaction1.tst

==

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

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

stringconnect "dsn=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

===

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


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

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

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

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

```
commit

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

print ' '

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

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

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

End Test
```

==

acid_isolation_test4_transaction1.tst

==

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

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

stringconnect "dsn=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 {select l_extendedprice from lineitem where l_linenumber=^ and l_orderkey=^}
```

```
                substitute line, ordr into extprice3

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

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

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

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

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

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

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

 disconnect
 End Test
```

===

acid_isolation_test4_transaction2.tst

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


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

stringconnect "dsn=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
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 ' '

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

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

End Test
```

===

acid_isolation_test5_query.tst

===

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

```
% it as part of tpcd_acid_isolation_main5.tst file.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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

stringconnect "dsn=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

===

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

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

stringconnect "dsn=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

==

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%  Created by: Masood Dirin
%  Created Date: 5/27/1999
%  Script name: tpcd_acid_isolation_query_test6.tst
%  ---------------------------------------------------------------------
%
%  This test could be run by itself, but it is recommended to run it as
%  part of tpcd_acid_isolation_main6.tst file. Run this file by itself as
%  follow:
%  dbtest -u acid_isolation_query_test6.tst > acid_isolation_query_test6.ot
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

Test                "tpcd_acid_isolation_query_test6.tst"
Description         "Run Acid isolation query for test 6"

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

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

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

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

stringconnect "dsn=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. Solaris chooses the controller numbers and target numbers at boot time depending upon the cabling configuration and the slot location of the HBAs. Thus another equivalently configured system may not have the same the controller numbers and SCSI targets as shown below.

Using the **format** command, partition the disks as follows

c0t0d0

0	root	wm	67 -	810	5.70GB
1	swap	wu	1 -	66	517.72MB
2	backup	wm	0 -	8920	68.34GB
7	home	wm	811 -	8920	62.13GB
8	boot	wu	0 -	0	7.84MB

c0t1d0

0	unassigned	wm	1 -	3134	24.01GB
1	unassigned	wm	3135 -	3526	3.00GB
2	backup	wu	0 -	3886	29.78GB
3	unassigned	wm	3527 -	3788	2.01GB
4	unassigned	wm	3789 -	3827	305.93MB
5	unassigned	wm	3828 -	3840	101.98MB
7	unassigned	wm	3841 -	3853	101.98MB
8	boot	wu	0 -	0	7.84MB

c0t2d0

0	unassigned	wm	1 -	3134	24.01GB
1	unassigned	wm	3135 -	3526	3.00GB
2	backup	wu	0 -	3886	29.78GB
3	unassigned	wm	3527 -	3788	2.01GB
4	unassigned	wm	3789 -	3827	305.93MB
5	unassigned	wm	3828 -	3840	101.98MB
8	boot	wu	0 -	0	7.84MB

c0t6d0

0	unassigned	wm	1 -	3134	24.01GB
1	unassigned	wm	3135 -	3526	3.00GB

```
  2     backup    wu       0 - 3886       29.78GB
  3 unassigned    wm    3527 - 3788        2.01GB
  4 unassigned    wm    3789 - 3827      305.93MB
  5 unassigned    wm    3828 - 3840      101.98MB
  8       boot    wu       0 -    0        7.84MB

c0t7d0
  0 unassigned    wm       1 - 3134       24.01GB
  1 unassigned    wm    3135 - 3526        3.00GB
  2     backup    wu       0 - 3886       29.78GB
  3 unassigned    wm    3527 - 3788        2.01GB
  4 unassigned    wm    3789 - 3827      305.93MB
  5 unassigned    wm    3828 - 3840      101.98MB
  8       boot    wu       0 -    0        7.84MB
c0t11d0
  0 unassigned    wm       1 - 3134       24.01GB
  1 unassigned    wm    3135 - 3526        3.00GB
  2     backup    wu       0 - 3886       29.78GB
  3 unassigned    wm    3527 - 3788        2.01GB
  4 unassigned    wm    3789 - 3827      305.93MB
  5 unassigned    wm    3828 - 3840      101.98MB
  8       boot    wu       0 -    0        7.84MB
c0t12d0
  0 unassigned    wm       1 - 3134       24.01GB
  1 unassigned    wm    3135 - 3526        3.00GB
  2     backup    wu       0 - 3886       29.78GB
  3 unassigned    wm    3527 - 3788        2.01GB
  4 unassigned    wm    3789 - 3827      305.93MB
  5 unassigned    wm    3828 - 3840      101.98MB
  8       boot    wu       0 -    0        7.84MB

c0t14d0
  0 unassigned    wm       1 - 3134       24.01GB
  1 unassigned    wm    3135 - 3526        3.00GB
  2     backup    wu       0 - 3886       29.78GB
  3 unassigned    wm    3527 - 3788        2.01GB
  4 unassigned    wm    3789 - 3827      305.93MB
  5 unassigned    wm    3828 - 3840      101.98MB
  8       boot    wu       0 -    0        7.84MB

c0t15d0
  0 unassigned    wm       1 - 3134       24.01GB
  1 unassigned    wm    3135 - 3526        3.00GB
  2     backup    wu       0 - 3886       29.78GB
  3 unassigned    wm    3527 - 3788        2.01GB
  4 unassigned    wm    3789 - 3827      305.93MB
  5 unassigned    wm    3828 - 3840      101.98MB
  8       boot    wu       0 -    0        7.84MB
```

Then use **svm** as follows to create the mirrors used for the IQ log and the database devices

```
metainit d1 1 1 c0t1d0s4
metainit d2 1 1 c0t2d0s4
metainit d3 -m d111 d112


metainit d11 1 1  c0t0d0s0
metainit d12 1 1  c0t1d0s0
metainit d10 -m d11 d12

metainit d21 1 1  c0t6d0s0
metainit d22 1 1  c0t7d0s0
metainit d20 -m d21 d22

metainit d31 1 1 c0t11d0s0
metainit d32 1 1 c0t12d0s0
metainit d30 -m d31 d32

metainit d41 1 1  c0t14d0s0
```

```
metainit d42 1 1  c0t15d0s0
metainit d40 -m d41 d42
```

File System Setup

```
run the following:

create a filesystem on /dev/md/rdsk/d1 and mount
/dev/md/dsk/d1 on /sybase2

newfs  /dev/md/rdsk/d1

mount  /dev/md/dsk/d1  /sybase2

echo "/dev/md/dsk/d1  /dev/md/rdsk/d1  /sybase2   ufs     1  yes   -" >>/etc/vfstab
```

Database Device Links

```
Finally create the following links in /sybase2 to be used as temp devices:


/sybase2/T01 -> /dev/rdsk/c0t1d0s3
/sybase2/T02 -> /dev/rdsk/c0t2d0s3
/sybase2/T03 -> /dev/rdsk/c0t6d0s3
/sybase2/T04 -> /dev/rdsk/c0t7d0s3
/sybase2/T05 -> /dev/rdsk/c0t11d0s3
/sybase2/T06 -> /dev/rdsk/c0t12d0s3
/sybase2/T07 -> /dev/rdsk/c0t14d0s3
/sybase2/T08 -> /dev/rdsk/c0t15d0s3

and create the following links in /sybase2 to be used as  database devices:

/sybase2/M01 -> /dev/md/rdsk/d10
/sybase2/M02 -> /dev/md/rdsk/d20
/sybase2/M03 -> /dev/md/rdsk/d30
/sybase2/M04 -> /dev/md/rdsk/d40
```

Appendix C. Query Text and Query Output

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

qualification query 1

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

% select
% l_returnflag,
% l_linestatus,
% sum(l_quantity) as sum_qty,
% sum(l_extendedprice) as sum_base_price,
% sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
% sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
% avg(l_quantity) as avg_qty,
% avg(l_extendedprice) as avg_price,
% avg(l_discount) as avg_disc,
% count(*) as count_order
% from
% lineitem
% where
% l_shipdate <= dateadd(day,-90,'1998-12-01')
% group by
% l_returnflag,
% l_linestatus
% order by
% l_returnflag,
% l_linestatus;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.32000 seconds - current time 16:40:13
'A','F',37734107,56586554400.7292032,53758257134.8694563,55909065222.8284717,25.5220058532573342,
38273.1297346211374,.0499852958383577168,1478493
'N','F',991417,1487504710.38000107,1413082168.05409968,1469649223.19436967,25.5164719205229819,38
284.4677608483374,.0500934266742134809,38854
'N','O',74476040,111701729697.737336,106118230307.607383,110367043872.495174,25.5022267695849895,
38249.117988907361,.049996586053555131,2920374
'R','F',37719753,56568041380.8983326,53741292684.6045375,55889619119.8339581,25.5057936126907617,
38250.8546260985255,.0500094058300870121,1478870
% total of 4 rows written

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

qualification query 2

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

% select top 100
% s_acctbal,
% s_name,
% n_name,
% p_partkey,
% p_mfgr,
% s_address,
% s_phone,
% s_comment
% from
% part,
% supplier,
% partsupp,
% nation,
% region
% where
```

```
% p_partkey = ps_partkey
% and s_suppkey = ps_suppkey
% and p_size = 15
% and p_type like 'BRASS'
% and s_nationkey = n_nationkey
% and n_regionkey = r_regionkey
% and r_name = 'EUROPE'
% and ps_supplycost = (
% select
% min(ps_supplycost)
% from
% partsupp,
% supplier,
% nation,
% region
% where
% p_partkey = ps_partkey
% and s_suppkey = ps_suppkey
% and s_nationkey = n_nationkey
% and n_regionkey = r_regionkey
% and r_name = 'EUROPE'
% )
% order by
% s_acctbal desc,
% n_name,
% s_name,
% p_partkey;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.76000 seconds - current time 16:40:25
9938.53,'Supplier#000005359       ','UNITED KINGDOM           ',185358,'Manufacturer#4
','QKuHYh,vZGiwu2FWEJoLDx04','33-429-790-6131','blithely silent pinto beans are furiously. slyly
final deposits acros'
9937.84,'Supplier#000005969       ','ROMANIA                  ',108438,'Manufacturer#1
','ANDENSOSmk,miq23Xfb5RWt6dvUcvt6Qa','29-520-692-3537','carefully slow deposits use furiously.
slyly ironic platelets above the ironic'
9936.22,'Supplier#000005250       ','UNITED KINGDOM           ',249,'Manufacturer#4
','B3rqp0xbSEim4Mpy2RH J','33-320-228-2957','blithely special packages are. stealthily express
deposits across the closely final instructi'
9923.77000000000119,'Supplier#000002324       ','GERMANY                  ',29821,'Manufacturer#4
','y3OD9UywSTOk','17-779-299-1839','quickly express packages breach quiet pinto beans. requ'
9871.22,'Supplier#000006373       ','GERMANY                  ',43868,'Manufacturer#5
','J8fcXWsTqM','17-813-485-8637','never silent deposits integrate furiously blit'
9870.78,'Supplier#000001286       ','GERMANY                  ',81285,'Manufacturer#2
','YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH','17-516-924-4574','final theodolites cajole slyly
special,'
9870.78,'Supplier#000001286       ','GERMANY                  ',181285,'Manufacturer#4
','YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH','17-516-924-4574','final theodolites cajole slyly
special,'
9852.52000000000119,'Supplier#000008973       ','RUSSIA                   ',18972,'Manufacturer#2
','t5L67YdBYYH6o,Vz24jpDyQ9','32-188-594-7038','quickly regular instructions wake-- carefully
unusual braids into the expres'
9847.83,'Supplier#000008097       ','RUSSIA                   ',130557,'Manufacturer#2
','xMe97bpE69NzdwLoX','32-375-640-3593','slyly regular dependencies sleep slyly furiously express
dep'
9847.57,'Supplier#000006345       ','FRANCE                   ',86344,'Manufacturer#1
','VSt3rzk3qG698u6ld8HhOByvrTcSTSvQlDQDag','16-886-766-7945','silent pinto beans should have to
snooze carefully along the final reques'
% total of 100 rows written
```

==========

qualification query 3

==========

```
% select top 10
% l_orderkey,
% sum(l_extendedprice * (1 - l_discount)) as revenue,
% o_orderdate,
```

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

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

qualification query 4

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

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

```
'5-LOW          ',10487
% total of 5 rows written

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

qualification query 5

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

% select
% n_name,
% sum(l_extendedprice * (1 - l_discount)) as revenue
% from
% customer,
% orders,
% lineitem,
% supplier,
% nation,
% region
% where
% c_custkey = o_custkey
% and l_orderkey = o_orderkey
% and l_suppkey = s_suppkey
% and c_nationkey = s_nationkey
% and s_nationkey = n_nationkey
% and n_regionkey = r_regionkey
% and r_name = 'ASIA'
% and o_orderdate >= '1994-01-01'
% and o_orderdate < dateadd(year,1,'1994-01-01')
% group by
% n_name
% order by
% revenue desc;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.65000 seconds - current time 16:40:36
'INDONESIA                ',55502041.1696999431
'VIETNAM                  ',55295086.9966999531
'CHINA                    ',53724494.2565999746
'INDIA                    ',52035512.000200057
'JAPAN                    ',45410175.6954000235
% total of 5 rows written

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

qualification query 6

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

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

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

qualification query 7

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

```
% select
% supp_nation,
% cust_nation,
% l_year,
% sum(volume) as revenue
% from
% (
% select
% n1.n_name as supp_nation,
% n2.n_name as cust_nation,
% datepart(year, l_shipdate) as l_year,
% l_extendedprice * (1 - l_discount) as volume
% from
% supplier,
% lineitem,
% orders,
% customer,
% nation n1,
% nation n2
% where
% s_suppkey = l_suppkey
% and o_orderkey = l_orderkey
% and c_custkey = o_custkey
% and s_nationkey = n1.n_nationkey
% and c_nationkey = n2.n_nationkey
% and (
% (n1.n_name = 'FRANCE' and n2.n_name = 'GERMANY')
% or (n1.n_name = 'GERMANY' and n2.n_name = 'FRANCE')
% )
% and l_shipdate between '1995-01-01' and '1996-12-31'
% ) as shipping
% group by
% supp_nation,
% cust_nation,
% l_year
% order by
% supp_nation,
% cust_nation,
% l_year;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.67000 seconds - current time 16:40:43
'FRANCE                  ','GERMANY                 ',1995,54639732.7335999489
'FRANCE                  ','GERMANY                 ',1996,54633083.3075999737
'GERMANY                 ','FRANCE                  ',1995,52531746.6696999669
'GERMANY                 ','FRANCE                  ',1996,52520549.0223998487
% total of 4 rows written
```

==========

qualification query 8

==========

```
% select
% o_year,
% sum(case
% when nation = 'BRAZIL' then volume
% else 0
% end) / sum(volume) as mkt_share
% from
% (
% select
% datepart(year, o_orderdate) as o_year,
% l_extendedprice * (1 - l_discount) as volume,
% n2.n_name as nation
% from
% part,
% supplier,
% lineitem,
```

```
% orders,
% customer,
% nation n1,
% nation n2,
% region
% where
% p_partkey = l_partkey
% and s_suppkey = l_suppkey
% and l_orderkey = o_orderkey
% and o_custkey = c_custkey
% and c_nationkey = n1.n_nationkey
% and n1.n_regionkey = r_regionkey
% and r_name = 'AMERICA'
% and s_nationkey = n2.n_nationkey
% and o_orderdate between '1995-01-01' and '1996-12-31'
% and p_type = 'ECONOMY ANODIZED STEEL'
% ) as all_nations
% group by
% o_year
% order by
% o_year;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.86000 seconds - current time 16:40:47
1995,.0344358904066548347
1996,.041485521293530345
% total of 2 rows written
```

qualification query 9

```
% select
% nation,
% o_year,
% sum(amount) as sum_profit
% from
% (
% select
% n_name as nation,
% datepart(year, o_orderdate) as o_year,
% l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity as
%     amount
% from
% part,
% supplier,
% lineitem,
% partsupp,
% orders,
% nation
% where
% s_suppkey = l_suppkey
% and ps_suppkey = l_suppkey
% and ps_partkey = l_partkey
% and p_partkey = l_partkey
% and o_orderkey = l_orderkey
% and s_nationkey = n_nationkey
% and p_name like 'green'
% ) as profit
% group by
% nation,
% o_year
% order by
% nation,
% o_year desc;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
```

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

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

qualification query 10

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

% select top 20
% c_custkey,% c_name,
% sum(l_extendedprice * (1 - l_discount)) as revenue,
% c_acctbal,
% n_name,
% c_address,
% c_phone,
% c_comment
% from
% customer,
% orders,
% lineitem,
% nation
% where
% c_custkey = o_custkey
% and l_orderkey = o_orderkey
% and o_orderdate >= '1993-10-01'
% and o_orderdate < dateadd(month,3,'1993-10-01')
% and l_returnflag = 'R'
% and c_nationkey = n_nationkey
% group by
% c_custkey,
% c_name,
% c_acctbal,
% c_phone,
% n_name,
% c_address,
% c_comment
% order by
% revenue desc;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.50000 seconds - current time 16:40:55
57040,'Customer#000057040',734235.2455,632.87,'JAPAN                     ','Eioyzjf4pp','22-895-
641-3466','requests sleep blithely about the furiously i'
143347,'Customer#000143347',721002.694799999952,2557.4700000000003,'EGYPT
','1aReFYv,Kw4','14-742-935-3718','fluffily bold excuses haggle finally after the u'
60838,'Customer#000060838',679127.307700000048,2454.77,'BRAZIL
','64EaJ5vMAHWJlBOxJklpNc2RJiWE','12-913-494-9813','furiously even pinto beans integrate under
the ruthless foxes; ironic, even dolphins across the slyl'
101998,'Customer#000101998',637029.566699999809,3790.89,'UNITED KINGDOM
','01c9CILnNtfOQYmZj','33-593-865-6378','accounts doze blithely! enticing, final deposits sleep
blithely special accounts. slyly express accounts pla'
125341,'Customer#000125341',633508.086,4983.5100000000006,'GERMANY
','S29ODD6bceU8QSuuEJznkNaK','17-582-695-5962','quickly express requests wake quickly blithely'
25501,'Customer#000025501',620269.784899999976,7725.04,'ETHIOPIA                  ','
W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ','15-874-808-6793','quickly special requests sleep evenly
among the special deposits. special deposi'
115831,'Customer#000115831',596423.867200000167,5098.1,'FRANCE                   ','rFeBbEEyk dl
ne7zV5fDrmiq1oK09wV7pxqCgIc','16-715-386-3788','carefully bold excuses sleep alongside of the
```

```
thinly idle'
84223,'Customer#000084223',594998.023899999976,528.65,'UNITED KINGDOM           ','nAVZCs6BaWap
rrM27N 2qBnzc5WBauxbA','33-442-824-8191','pending, final ideas haggle final requests. unusual,
regular asymptotes affix according to the even foxes.'
54289,'Customer#000054289',585603.391799999952,5583.02,'IRAN
','vXCxoCsU0Bad5JQI ,oobkZ','20-834-292-4707','express requests sublate blithely regular
requests. regular, even ideas solve.'
39922,'Customer#000039922',584878.113399999976,7321.10999999999881,'GERMANY
','Zgy4s50l2GKN4pLDPBU8m342gIw6R','17-147-757-8036','even pinto beans haggle. slyly bold accounts
inte'
6226,'Customer#000006226',576783.760599999905,2230.09,'UNITED KINGDOM
','8gPu8,NPGkfyQQ0hcIYUGPIBWc,ybP5g,','33-657-701-3391','quickly final requests against the
regular instructions wake blithely final instructions. pa'
922,'Customer#000000922',576767.533299999833,3869.25,'GERMANY
','Az9RFaut7NkPnc5zSD2PwHgVwr4jRzq','17-945-916-9648','boldly final requests cajole blith'
147946,'Customer#000147946',576455.132,2030.1300000000003,'ALGERIA
','iANyZHjqhyy7Ajah0pTrYyhJ','10-886-956-3143','furiously even accounts are blithely above the
furiousl'
115640,'Customer#000115640',569341.193299999952,6436.1,'ARGENTINA               ','Vtgfia9qI
7EpHgecU1X','11-411-543-4901','final instructions are slyly according to the'
73606,'Customer#000073606',568656.857799999952,1785.67,'JAPAN
','xuR0Tro5yChDfOCrjkd2ol','22-437-653-6966','furiously bold orbits about the furiously busy
requests wake across the furiously quiet theodolites. d'
110246,'Customer#000110246',566842.981499999881,7763.35,'VIETNAM                 ','7KzflgX
MDOq7sOkI','31-943-426-9837','dolphins sleep blithely among the slyly final'
142549,'Customer#000142549',563537.236799999952,5085.9899999999994,'INDONESIA
','ChqEoK43OysjdHbtKCp6dKqjNyvvi9','19-955-562-2398','regular, unusual dependencies boost slyly;
ironic attainments nag fluffily into the unusual packages?'
146149,'Customer#000146149',557254.9865,1791.55,'ROMANIA                 ','s87fvzFQpU','29-744-
164-6487','silent, unusual requests detect quickly slyly regul'
52528,'Customer#000052528',556397.350899999976,551.79,'ARGENTINA
','NFztyTOR10UOJ','11-208-192-3205','unusual requests detect. slyly dogged theodolites use slyly.
deposit'
23431,'Customer#000023431',554269.536000000119,3381.86,'ROMANIA
','HgiV0phqhaIa9aydNoIlb','29-915-458-2654','instructions nag quickly. furiously bold accounts
cajol'
% total of 20 rows written
```

==========

qualification query 11

==========

```
% select
% ps_partkey,
% sum(ps_supplycost * ps_availqty) as value
% from
% partsupp,
% supplier,
% nation
% where
% ps_suppkey = s_suppkey
% and s_nationkey = n_nationkey
% and n_name = 'GERMANY'
% group by
% ps_partkey having
% sum(ps_supplycost * ps_availqty) > (
% select
% sum(ps_supplycost * ps_availqty) * 0.0001000000
% from
% partsupp,
% supplier,
% nation
% where
% ps_suppkey = s_suppkey
% and s_nationkey = n_nationkey
% and n_name = 'GERMANY'
% )
% order by
% value desc;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
```

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

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

qualification query 12

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

% select
% l_shipmode,
% sum(case
% when o_orderpriority = '1-URGENT'
% or o_orderpriority = '2-HIGH'
% then 1
% else 0
% end) as high_line_count,
% sum(case
% when o_orderpriority <> '1-URGENT'
% and o_orderpriority <> '2-HIGH'
% then 1
% else 0
% end) as low_line_count
% from
% orders,
% lineitem
% where
% o_orderkey = l_orderkey
% and l_shipmode in ('MAIL', 'SHIP')
% and l_commitdate < l_receiptdate
% and l_shipdate < l_commitdate
% and l_receiptdate >= '1994-01-01'
% and l_receiptdate < dateadd(year,1,'1994-01-01')
% group by
% l_shipmode
% order by
% l_shipmode;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.26000 seconds - current time 16:41:03
'MAIL      ',6202,9324
'SHIP      ',6200,9262
% total of 2 rows written

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

qualification query 13

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

% select
% c_count,
% count(*) as custdist
% from
% (
% select
% c_custkey,
% count(o_orderkey)
```

```
% from
% customer left outer join orders on
% c_custkey = o_custkey
% and o_comment not like 'specialrequests'
% group by
% c_custkey
% ) as c_orders (c_custkey, c_count)
% group by
% c_count
% order by
% custdist desc,
% c_count desc;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.17000 seconds - current time 16:41:06
0,50004
9,6641
10,6566
11,6058
8,5949
12,5553
13,4989
19,4748
7,4707
18,4625
% total of 42 rows written
```

==========

qualification query 14

==========

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

==========

qualification query 15

==========

```
 Executing command:
% create view revenue0 (supplier_no, total_revenue) as
% select
% l_suppkey,
% sum(l_extendedprice * (1 - l_discount))
% from
% lineitem
% where
% l_shipdate >= '1996-01-01'
% and l_shipdate < dateadd(month,3,'1996-01-01')
```

```
% group by
% l_suppkey;
% execution time 0.81000 seconds - current time 16:41:21

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

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

qualification query 16

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

% select
% p_brand,
% p_type,
% p_size,
% count(distinct ps_suppkey) as supplier_cnt
% from
% partsupp,
% part
% where
% p_partkey = ps_partkey
% and p_brand <> 'Brand#45'
% and p_type not like 'MEDIUM POLISHED'
% and p_size in (49, 14, 23, 45, 19, 3, 36, 9)
% and ps_suppkey not in (
% select
% s_suppkey
% from
% supplier
% where
% s_comment like 'CustomerComplaints'
% )
% group by
% p_brand,
% p_type,
% p_size
% order by
% supplier_cnt desc,
% p_brand,
% p_type,
% p_size;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
```

```
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.27000 seconds - current time 16:41:22
'Brand#41  ','MEDIUM BRUSHED TIN',3,28
'Brand#54  ','STANDARD BRUSHED COPPER',14,27
'Brand#11  ','STANDARD BRUSHED TIN',23,24
'Brand#11  ','STANDARD BURNISHED BRASS',36,24
'Brand#15  ','MEDIUM ANODIZED NICKEL',3,24
'Brand#15  ','SMALL ANODIZED BRASS',45,24
'Brand#15  ','SMALL BURNISHED NICKEL',19,24
'Brand#21  ','MEDIUM ANODIZED COPPER',3,24
'Brand#22  ','SMALL BRUSHED NICKEL',3,24
'Brand#22  ','SMALL BURNISHED BRASS',19,24

% total of 18314 rows written

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

qualification query 17

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

% select
% sum(l_extendedprice) / 7.0 as avg_yearly
% from
% lineitem,
% part
% where
% p_partkey = l_partkey
% and p_brand = 'Brand#23'
% and p_container = 'MED BOX'
% and l_quantity < (
% select
% 0.2 * avg(l_quantity)
% from
% lineitem
% where
% l_partkey = p_partkey
% );
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.22000 seconds - current time 16:41:28
348406.054285713732
% total of 1 rows written

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

qualification query 18

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

% select top 100
% c_name,
% c_custkey,
% o_orderkey,
% o_orderdate,
% o_totalprice,
% sum(l_quantity)
% from
% customer,
% orders,
% lineitem
% where
% o_orderkey in (
% select
% l_orderkey
% from
% lineitem
% group by
% l_orderkey having
% sum(l_quantity) > 300
```

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

==========

qualification query 19

==========

```
% select
% sum(l_extendedprice* (1 - l_discount)) as revenue
% from
% lineitem,
% part
% where
% (
% p_partkey = l_partkey
% and p_brand = 'Brand#12'
% and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
% and l_quantity >= 1 and l_quantity <= 1 + 10
% and p_size between 1 and 5
% and l_shipmode in ('AIR', 'AIR REG')
% and l_shipinstruct = 'DELIVER IN PERSON'
% )
% or
% (
% p_partkey = l_partkey
% and p_brand = 'Brand#23'
% and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
% and l_quantity >= 10 and l_quantity <= 10 + 10
% and p_size between 1 and 10
% and l_shipmode in ('AIR', 'AIR REG')
% and l_shipinstruct = 'DELIVER IN PERSON'
% )
% or
% (
% p_partkey = l_partkey
% and p_brand = 'Brand#34'
% and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
% and l_quantity >= 20 and l_quantity <= 20 + 10
% and p_size between 1 and 15
% and l_shipmode in ('AIR', 'AIR REG')
% and l_shipinstruct = 'DELIVER IN PERSON'
% );
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
```

```
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.35000 seconds - current time 16:41:46
3083843.05780000031
% total of 1 rows written
```

==========

qualification query 20

==========

```
% select
% s_name,
% s_address
% from
% supplier,
% nation
% where
% s_suppkey in (
% select
% ps_suppkey
% from
% partsupp
% where
% ps_partkey in (
% select
% p_partkey
% from
% part
% where
% p_name like 'forest'
% )
% and ps_availqty > (
% select
% 0.5 * sum(l_quantity)
% from
% lineitem
% where
% l_partkey = ps_partkey
% and l_suppkey = ps_suppkey
% and l_shipdate >= '1994-01-01'
% and l_shipdate < dateadd(year,1,'1994-01-01')
% )
% )
% and s_nationkey = n_nationkey
% and n_name = 'CANADA'
% order by
% s_name;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.37000 seconds - current time 16:41:51
'Supplier#000000020       ','iybAE,RmTymrZVYaFZva2SH,j'
'Supplier#000000091       ','YV45D7TkfdQanOOZ7q9QxkyGUapU1oOWU6q3'
'Supplier#000000197       ','YC2Acon6kjY3zj3Fbxs2k4Vdf7X0cd2F'
'Supplier#000000226       ','83qOdU2EYRdPQAQhEtn GRZEd'
'Supplier#000000285       ','Br7e1nnt1yxrw6ImgpJ7YdhFDjuBf'
'Supplier#000000378       ','FfbhyCxWvcPrO8ltp9'
'Supplier#000000402       ','i9Sw4DoyMhzhKXCH9By,AYSgmD'
'Supplier#000000530       ','0qwCMwobKY OcmLyfRXlagA8ukENJv,'
'Supplier#000000688       ','D fw5ocppmZpYBBIPI718hCihLDZ5KhKX'
'Supplier#000000710       ','f19YPvOyb QoYwjKC,oPycpGfieBAcwKJo'
% total of 204 rows written
```

==========

qualification query 21

==========

```
% select top 100
% s_name,
% count(*) as numwait
% from
% supplier,
% lineitem l1,
% orders,
% nation
% where
% s_suppkey = l1.l_suppkey
% and o_orderkey = l1.l_orderkey
% and o_orderstatus = 'F'
% and l1.l_receiptdate > l1.l_commitdate
% and exists (
% select
% *
% from
% lineitem l2
% where
% l2.l_orderkey = l1.l_orderkey
% and l2.l_suppkey <> l1.l_suppkey
% )
% and not exists (
% select
% *
% from
% lineitem l3
% where
% l3.l_orderkey = l1.l_orderkey
% and l3.l_suppkey <> l1.l_suppkey
% and l3.l_receiptdate > l3.l_commitdate
% )
% and s_nationkey = n_nationkey
% and n_name = 'SAUDI ARABIA'
% group by
% s_name
% order by
% numwait desc,
% s_name;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.54000 seconds - current time 16:41:53
'Supplier#000002829       ',20
'Supplier#000005808       ',18
'Supplier#000000262       ',17
'Supplier#000000496       ',17
'Supplier#000002160       ',17
'Supplier#000002301       ',17
'Supplier#000002540       ',17
'Supplier#000003063       ',17
'Supplier#000005178       ',17
'Supplier#000008331       ',17
% total of 100 rows written
```

==========

qualification query 22

==========

```
% select
% cntrycode,
% count(*) as numcust,
% sum(c_acctbal) as totacctbal
% from
% (
% select
% substring(c_phone,1,2) as cntrycode,
% c_acctbal
% from
```

```
% customer
% where
% substring(c_phone,1,2) in
% ('13', '31', '23', '29', '30', '18', '17')
% and c_acctbal > (
% select
% avg(c_acctbal)
% from
% customer
% where
% c_acctbal > 0.00
% and substring(c_phone,1,2) in
% ('13', '31', '23', '29', '30', '18', '17')
% )
% and not exists (
% select
% *
% from
% orders
% where
% o_custkey = c_custkey
% )
% ) as custsale
% group by
% cntrycode
% order by
% cntrycode;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.18000 seconds - current time 16:42:07
'13',888,6737713.98999999881
'17',861,6460573.72
'18',964,7236687.40000001431
'23',892,6701457.95000000954
'29',948,7158866.62999999642
'30',909,6808436.13000000119
'31',922,6806670.17999998569
% total of 7 rows written
Appendix D. Seed and Query Substitution Parameters
```

Appendix D. Query Parameters

This Appendix contains Seed values and substitution parameters for each stream

Seed Values

```
stream0 1104160842
stream1 1104160843
stream2 1104160844
stream3 1104160845
stream4 1104160846
stream5 1104160847
```

Query Parameters

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

stream0: seed = 1104160842

```
14      1997-10-01
2       39      BRASS   ASIA
9       yellow
20      blanched        1994-01-01      RUSSIA
6       1993-01-01      0.05    24
17      Brand#44        LG BAG
18      315
8       IRAN    MIDDLE EAST     PROMO BRUSHED COPPER
21      INDIA
13      unusual         deposits
3       MACHINERY       1995-03-14
22      14      10      31      16      34      32
19
16      Brand#14        ECONOMY POLISHED        29      5
        27      40      17      15      16      3
4       1995-04-01
11      CHINA   0.0000010000
15      1995-04-01
1       115
10      1994-08-01
19      Brand#21        Brand#25        Brand#32        8
        12      20
5       ASIA    1993-01-01
7       INDONESIA       IRAN
12      AIR     MAIL    1997-01-01
==================================================
```

stream1: seed =1104160843

```
21      ALGERIA
3       BUILDING        1995-03-31
18      312
5       EUROPE  1993-01-01
11      FRANCE  0.0000010000
7       ARGENTINA       BRAZIL
6       1993-01-01      0.02    24
20      lime    1993-01-01      JAPAN
17      Brand#41        LG PKG
12      REG AIR         MAIL    1993-01-01
16      Brand#44        STANDARD BRUSHED        17
13      4       2       38      8       9       46
15      1993-01-01
13      unusual         packages
10      1993-06-01
2       27      TIN     MIDDLE EAST
8       BRAZIL  AMERICA         PROMO PLATED COPPER
14      1993-02-01
19      Brand#23        Brand#13        Brand#21        3
        13      27
9       thistle
22      11      15      13      16      20      10
23
1       62
4       1997-11-01
==================================================
```

stream2: seed = 1104160844

```
6       1993-01-01      0.07    25
17      Brand#43        LG DRUM
14      1993-05-01
16      Brand#24        LARGE BURNISHED         21
35      24      25      16      50      2       3
19      Brand#25        Brand#41        Brand#25        8
        14      23
10      1994-03-01
9       slate
2       15      COPPER  ASIA
15      1995-08-01
8       ROMANIA         EUROPE  PROMO ANODIZED COPPER
5       MIDDLE EAST     1993-01-01
22      27      30      34      23      19      10
12
12      SHIP    FOB     1993-01-01
7       CHINA   ROMANIA
13      unusual         packages
18      314
1       70
4       1995-07-01
20      tan     1996-01-01      ARGENTINA
3       HOUSEHOLD       1995-03-17
11      ROMANIA         0.0000010000
21      PERU
==================================================
```

stream3: seed = 1104160845

```
8       IRAQ    MIDDLE EAST     ECONOMY POLISHED COPPER
5       AFRICA  1993-01-01
4       1993-04-01
6       1993-01-01      0.05    24
17      Brand#45        MED BAG
7       IRAN    IRAQ
1       78
18      315
22      32      11      30      17      22      34
18
14      1993-08-01
9       saddle
10      1994-12-01
15      1993-05-01
11      GERMANY         0.0000010000
20      gainsboro       1995-01-01      MOZAMBIQUE
2       3       STEEL   AFRICA
21      IRAN
19      Brand#32        Brand#34        Brand#25        4
        15      30
13      express         packages
16      Brand#14        PROMO PLATED    24      26
21      13      48      46      36      25
12      FOB     REG AIR         1993-01-01
3       BUILDING        1995-03-02
==================================================
```

stream4: seed = 1104160846

```
5       ASIA    1994-01-01
21      BRAZIL
14      1993-11-01
19      Brand#34        Brand#12        Brand#14        9
        16      26
15      1995-11-01
17      Brand#42        MED PKG
12      TRUCK   FOB     1993-01-01
6       1994-01-01      0.02    24
4       1995-11-01
9       puff
8       CANADA  AMERICA         ECONOMY BURNISHED COPPER
```

```
16      Brand#44        SMALL BRUSHED   1       20
50      13      24      22      43      33
11      SAUDI ARABIA    0.0000010000
2       40      BRASS   ASIA
10      1993-09-01
18      313
1       86
13      express         packages
7       BRAZIL  CANADA
22      33      14      13      31      24      25
10
3       HOUSEHOLD       1995-03-19
20      red     1993-01-01      FRANCE
==================================================
```

stream5: seed = 1104160847

```
21      ROMANIA
15      1993-08-01
4       1993-08-01
6       1994-01-01      0.08    25
7       ROMANIA         SAUDI ARABIA
16      Brand#25 ECONOMY ANODIZED       20      9       39
        18      46      21      17      16
19      Brand#31 Brand#45 Brand#13 4    17      23
18      314
14      1994-02-01
22      21      12      33      11      28      30      20
11      INDIA   0.0000010000
13      express packages
3       AUTOMOBILE      1995-03-04
1       94
2       28      NICKEL  AFRICA
5       EUROPE  1994-01-01
8       SAUDI ARABIA    MIDDLE EAST     ECONOMY
ANODIZED TIN
20      chocolate 1996-01-01            VIETNAM
12      RAIL    FOB     1994-01-01
17      Brand#54 MED DRUM
10      1994-06-01
9       papaya
```

Appendix E. Implementation-Specific Layer/Driver Code

==========

==========

ntest15

```
#!/bin/bash

# Notes:
# Notes:  the default behavior of ntest15 is to load tables serially in the optimal order
# Notes:  for parallel loads with restart after load, use a negative value for the number of disks
# Notes:

RESTART_IQ_AFTER_LOAD=

cur_dir=$(pwd)
if [[ $cur_dir != /export/home/sybase/run/scripts ]]
then
     echo
     echo ERROR: $(basename $0) must be run from $HOME/run/scripts
     echo
     echo "       the current dir is $cur_dir "
     echo
     exit
fi


export audit_file_dir="/sybase_stage/results/"


if (( $# < 1 ))
then
  echo
  echo "usage: `basename $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) (without restarting IQ)                 "
  echo "          - stream0       restart IQ and do all 22 queries without any refreshes            "
  echo "          - refresh       restart IQ and do a single rerfesh pair                           "
  echo "          - power         restart IQ and do a single power run                              "
  echo "          - tpworf        restart IQ and do a single throughput run without any refreshes   "
  echo "          - throughput    restart IQ and do a single throughput run with concurrent refresh "
  echo "          - throughputtr  restart IQ and do a single throughput run with trailing refreshes "
  echo "          - lthroughput   restart IQ and do a load-throughput cycle                         "
  echo "          - run1          restart IQ and do a single power-throughput cycle (without a load) "
  echo "          - lrun1         restart IQ and do a load-power-throughput cycle                   "
  echo "          - all           do full audit run (load + 2 power-throughput cycles)"
  echo "          - alltr         do full audit run (load + 2 power-throughput with trailing refresh cycles)"
  echo "          - mall          do mpx audit run (load + 2 power-throughput cycles), assumes running IQ"
  echo
  exit
fi

scope=$1

 echo scope=$scope     # remove

case $scope in
      restart)  if (( $# < 1 || $# > 1 ))
                then
                    echo
                    echo "usage: `basename $0`    restart"
                    echo
                    exit
                fi
                ;;

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

                sf=$2 ;;

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

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

                sf=$2  ;;

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

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

                sf=$2
                input_seed=$3   ;;

     refresh)  if (( $# < 2 || $# > 2 ))
                then
                    echo
```

```
            echo "usage: `basename $0`    refresh    scale_factor  "
            echo
            exit
        fi

        sf=$2 ;;

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

        sf=$2
        nqs=0
        input_seed=$3  ;;

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

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

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

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

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

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

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

        real_scope2=$scope     # i.e. real_scope2 is lthroughput
        scope=all              # pretend for now that scope = all
        sf=$2
        nqs=$3
        input_seed=$4   ;;

    run1)  if (( $# < 7 || $# > 7 ))
        then
            echo
            echo "usage: `basename $0`    run1    scale_factor  "
            echo "                num_query_streams                   "
            echo "                num_disks  disk_size(in GB)  system_cost    "
            echo "                input_seed                          "
            echo
            exit
            echo
        fi

        sf=$2
        nqs=$3

        let  nd=$4
        if (( nd < 0 ))
        then
           RESTART_IQ_AFTER_LOAD=yes
        fi

        let ds=$5
        let sc=$6
        input_seed=$7  ;;

   lrun1)  if (( $# < 7 || $# > 7 ))
        then
            echo
            echo "usage: `basename $0`    lrun1  scale_factor  "
            echo "                num_query_streams                   "
            echo "                num_disks  disk_size(in GB)  system_cost    "
            echo "                input_seed                          "
            echo
            exit
            echo
        fi

        real_scope=$scope     # i.e. real_scope is lrun1
        scope=all             # pretend for now that scope = all
        sf=$2
        nqs=$3
        let nd=$4
        if (( nd < 0 ))
        then
           RESTART_IQ_AFTER_LOAD=yes
        fi

        let ds=$5
        let sc=$6
        input_seed=$7
        echo input_seed = $input_seed ;;

     all)  if (( $# < 7 || $# > 7 ))
        then
            echo
            echo "usage: `basename $0`    all    scale_factor  "
            echo "                num_query_streams                   "
            echo "                num_disks  disk_size(in GB)  system_cost    "
            echo "                input_seed                          "
            echo
            exit
            echo
        fi

        sf=$2
echo scale=$sf
        nqs=$3
        let nd=$4
        if (( nd < 0 ))
        then
           RESTART_IQ_AFTER_LOAD=yes
        fi
```

```
        let ds=$5
        let sc=$6
        input_seed=$7
        echo input_seed = $input_seed ;;

    alltr)  if (( $# < 7 || $# > 7 ))
        then
            echo
            echo "usage: `basename $0`    all    scale_factor  "
            echo "                num_query_streams                    "
            echo "                num_disks  disk_size(in GB)  system_cost    "
            echo "                input_seed                           "
            echo
            exit
            echo
        fi

        sf=$2
        nqs=$3
        let nd=$4
        if (( nd < 0 ))
        then
            RESTART_IQ_AFTER_LOAD=yes
        fi

        let ds=$5
        let sc=$6
        input_seed=$7  ;;

    mall) if (( $# < 7 || $# > 7 ))
        then
            echo
            echo "usage: `basename $0`   mall    scale_factor  "
            echo "                num_query_streams                    "
            echo "                num_disks  disk_size(in GB)  system_cost"
            echo "                input_seed"
            echo
            exit
            echo
        fi
        sf=$2
        nqs=$3
        let nd=$4
        if (( nd < 0 ))
        then
            RESTART_IQ_AFTER_LOAD=yes
        fi

        let ds=$5
        let sc=$6
        input_seed=$7  ;;

     *)  echo
        echo "ERROR: scope (=$scope) must be one of:                    "
        echo
        echo "   restart,load,q1,q2,...,q22,stream0,refresh,power,"
        echo "   tpworf,throughput,lthroughput,run1,lrun1,all     "
        echo
        echo
        exit
esac

echo INPUT_SEED = $input_seed

# start_load.out gets created if there is a load; existance is tested in
tpch_report.bash
# to determine if load order should be displayed in the report
rm start_load.out


# we impose the minimum stream requirement only in the "all" case
# for all other scopes the min number of query streams is 1
# aminqs is thus the "allowed minimum" for the current run
aminqs=1
if [[ ($scope = all || $scope = mall) && -z $real_scope && -z $real_scope2 ]]  #
really all; not lrun1 or lthroughput1
then
  case $sf in
       .1)  aminqs=1
            ;;
        1)  aminqs=2
            ;;
       10)  aminqs=3
            ;;
       30)  aminqs=4
            ;;
       50)  aminqs=4
            ;;
      100)  aminqs=5
            ;;
      300)  aminqs=6
            ;;
     1000)  aminqs=7
            ;;
     3000)  aminqs=8
            ;;
    10000)  aminqs=9
            ;;
      * )
            echo
            echo "ERROR:  scale factor (=$sf) must be one of
(.1,1,10,30,50,100,300,1000,3000,10000)"
            echo
            exit
            echo;;
  esac
fi

# These are the minimum number of streams for a compliant run
case $sf in
      1)  cmin=2
          ;;
     10)  cmin=3
          ;;
     30)  cmin=4
          ;;
     50)  cmin=4
          ;;
    100)  cmin=5
          ;;
    300)  cmin=6
          ;;
   1000)  cmin=7
          ;;
   3000)  cmin=8
          ;;
  10000)  cmin=9
          ;;
    *)  cmin=1
          ;;
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=nqs+1))

echo "XXXXX   min_required_connections = $min_req_connections
input_seed=$input_seed"

if [[ $scope != mall ]]
then
```

```
  if ((max_connections<min_req_connections))
  then
echo "  XXXXX   $min_req_connections"
    echo
    echo "In order to run $nqs 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
else
  echo
  echo "WARNING: Make sure total no. of connections is >= no. of streams +
1"
  echo
fi


# make sure MPSS values are legal for the scopes where they have been supplied
# as commandline args (i.e. everything but the standalone queries)

PLATFORM=`uname -m`

if [[ $scope != q* ]]    # every other scope restarts IQ
then
    mpssheap=$(current_mpssheap_value.bash)
    if [[ $PLATFORM == "i86pc" ]]
    then
      if [[ $mpssheap != 4k && $mpssheap != 2m && \
           $mpssheap != 4K && $mpssheap != 2M ]]
      then
        echo
        echo "ERROR: mpssheap (=$mpssheap) defined in
$IQDIR15/bin64/start_iq"
        echo "       must be one of [4k,2m] where k and m are case-insensitive"
        echo
        exit
      fi
    else
      if [[ $mpssheap != 8k && $mpssheap != 64k && $mpssheap != 512k &&
\
        $mpssheap != 8K && $mpssheap != 64K && $mpssheap != 512K &&  \
        $mpssheap != 4m && $mpssheap != 32m && $mpssheap != 256m
&& \
        $mpssheap != 4M && $mpssheap != 32M && $mpssheap !=
256M       ]]
      then
        echo
        echo "ERROR: mpssheap (=$mpssheap) defined in
$IQDIR15/bin64/start_iq"
        echo "       must be one of [8k,64k,512k,4m,256m] where k and m are
case-insensitive"
        echo
        exit
      fi
    fi


    mpssstack=$(current_mpssstack_value.bash)
    if [[ $PLATFORM == "i86pc" ]]
    then
      if [[ $mpssstack != 4k && $mpssstack != 2m && \
           $mpssstack != 4K && $mpssstack != 2M ]]
      then
        echo
        echo "ERROR: mpssstack (=$mpssstack) defined in
$IQDIR15/bin64/start_iq"
        echo "       must be one of [4k,2m] where k and m are case-insensitive"
        echo
        exit
      fi
   else
      if [[ $mpssstack != 8k && $mpssstack != 64k && $mpssstack != 512k
&& \
        $mpssstack != 8K && $mpssstack != 64K && $mpssstack != 512K &&
\
        $mpssstack != 4m && $mpssstack != 32m && $mpssstack != 256m &&
\
        $mpssstack != 4M && $mpssstack != 32M && $mpssstack != 256M
]]
      then
        echo
        echo "ERROR: mpssstack (=$mpssstack) defined in
$IQDIR15/bin64/start_iq"
        echo "       must be one of [8k,64k,512k,4m,256m] where k and m are
case-insensitive"
        echo
        exit
      fi
    fi
fi


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

if [[ $scope = stream0    || $scope = power || $scope = tpworf ||  \
    $scope = throughputtr || $scope = throughput || $scope = run1  || \
    $scope = all || $scope = alltr  ]]
then
    if [[ $input_seed < 0 ]]
    then
        echo
        echo "ERROR: input_seed($input_seed) must be >= 0"
        echo
        exit
    fi
fi


# make sure disk_size is reasonable for the scopes where it is supplied
# as a commandline arg  (i.e. run1, lrun1 and all)

if [[ ( $scope = run1 || $scope = all || $scope = alltr || $scope = mall ) &&
$real_scope2 != lthroughput ]]
then
    if ((ds < 20))
    then
        echo
        echo "ERROR: disk_size (=$ds) appears to be too small"    #  this may
need to be changed
        echo
        exit
    fi

    if ((ds > 2000))
    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 `basename $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 != 1 && $sf != 10 && $sf != 30 && $sf != 50 && \
         $sf != 100 && $sf != 300 && $sf != 1000 && $sf != 3000 && $sf !=
10000 ]]
    then
```

```
        echo
        echo "ERROR: scale factor (=$sf) must be one of
[.1,1,10,30,50,100,300,1000,3000,10000]"
        echo
        exit
    fi
fi


# make sure the correct version of tpch_wait.sql is being used
if [[ $scope = throughput || $scope = throughputtr || $scope = lthroughput ||
$scope = run1 || $scope = lrun1 ||$scope = all || $scope = alltr || $scope = mall ||
$scope = load ]]
then
  if [[ -e tpch_wait_${sf}GB.sql ]]
  then
    cp tpch_wait_${sf}GB.sql tpch_wait.sql
  else
    echo
    echo "ERROR: tpch_wait_${sf}GB.sql does not exist"
    echo
    exit
  fi

  # generate update_throughputX.sql
  upd_tput_fname="update_throughput${nqs}.sql"

  sleep_time_in_secs=0
  #cat update_throughput_header > $upd_tput_fname
  sed "s/xxx/$sleep_time_in_secs/" update_throughput_header  >
$upd_tput_fname
  i=1
  while ((i<=nqs))
  do
    echo "call tpch_rf1 (c_path,'${i}');" >> $upd_tput_fname
    echo "commit;" >> $upd_tput_fname
    echo "tpch_wait;" >> $upd_tput_fname
    echo "call tpch_rf2 (c_path,'${i}');" >> $upd_tput_fname
    echo "commit;" >> $upd_tput_fname
    if ((i<nqs))
    then
      echo "tpch_wait;" >> $upd_tput_fname
    fi
    ((i=i+1))
  done
  cat update_throughput_footer >> $upd_tput_fname

  if [[ -f update_throughput${nqs}.sql ]]
  then
      echo
  else
      echo
      echo "ERROR: update_throughput${nqs}.sql must exist for a run of $nqs
streams"
      echo
      exit
  fi

fi

# make sure that the number of query streams >= minimum allowed num
streams
# aminqs has been previously set to 1 except when $scope is "all" in which
# case aminqs is the complient minimun

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

# 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 || $scope = alltr || $scope = mall) && $sf != .
1 ]]  # can't do sf=.1 without floating point support
then
    rm -f dbgen_files
    ln -s /sybase_stage/${sf}GB dbgen_files
    if [[ ! -x dbgen_files ]]
    then
        echo "ERROR: /sybase_stage/${sf}GB does not exist"
        echo
        exit
    fi
    cd dbgen_files

    if [[ ! -f dbgen_size ]]
    then
      echo "ERROR: /sybase_stage/${sf}GB appears to be incomplete"
      echo
      exit
    fi
    dbg=$(dbgen_size)
    if (( dbg < sf*1000000000 ))
    then
        echo
        echo "WARNING: dbgen files in /sybase_stage/${sf}GB contain only $
(show_dbgen_size)"
        echo "        --  not enough for a scale factor of $sf -- "
        echo
        #exit
    fi

    if (( dbg > 2*sf*1000000000 ))
    then
        echo
        echo "ERROR: dbgen files in /sybase_stage/${sf}GB contain $
(show_dbgen_size)"
        echo "       --  too much for a scale factor of $sf --  "
        echo
    fi
    cd /export/home/sybase/run/scripts
fi

#dbgen often sets strange premissions on customer
cust_file_name="dbgen_files/customer.tbl"
if [[ ! -r ${cust_file_name} ]]
then
  echo
  echo "      ERROR: File customer.tbl doesn't have correct permission"
  echo
  exit 1
fi

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

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

     if ((liss>lill))
     then
         echo
         echo $liss lineitem.tbl files in /sybase_stage/${sf}GB but only $lill in
load_lineitem.sql
         echo
         exit
```

```
    fi

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

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

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

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


# decide if orderkey should be bigint or int (regardless of scope)

export partitioned=
#export partitioned=_partitioned_by_months
#export partitioned=_partitioned_by_6months
#export partitioned=_partitioned_by_years


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

#if [[ $scope != q* && $scope != stream0 && $scope != restart && $sf != .1 ]]
#??? big can be set unconditionally
#then
#     if ((sf >= 1000))
#     then
#       big=big     #  make l_orderkey and o_orderkey bigint
#     else
#       big=        #  otherwise int
#     fi
#fi

# we want to make sure that /sybase2 is a mounted filesystem
# this will not guarantee that it is mounted on a mirrored device
# but at least it will prevent running with /sybase2 as a subdirectory
# of the root filesystem
if [[ ! $(df -kF ufs | grep /sybase2) ]]
then
  echo
  echo "ERROR: /sybase2 is not a mounted filesystem"
  echo
  exit
fi

# make sure that M* and T* referenced in create_databases.sql &
add_xxxx_files.sql
# are writable links
# if not, fix /sybase2

if [[ $scope = load || $scope = all || $scope = alltr ]]
then
    if [[ ! -L /sybase2/M01 ]]
    then
        echo "ERROR: /sybase2/M01 is not a symbolic link "
        echo
        exit
    fi
    if [[ ! -w  /sybase2/M01 ]]
    then
        echo "ERROR: /sybase2/M01 is not writable"
        echo
        exit
    fi

    if [[ ! -L /sybase2/T01 ]]
    then
        echo "ERROR: /sybase2/T01 is not a symbolic link "
        echo
        exit
    fi
    if [[ ! -w  /sybase2/T01 ]]
    then
        echo "ERROR: /sybase2/T01 is not writable"
        echo
        exit
    fi

for i in $(grep sybase add_main_files.sql | grep -v "\-\-" | tr -s " "  | tr -d "'" | tr -d
"," | tr -d ";" | cut -d" " -f3)
    do
      if [[ ! -L $i ]]
      then
        echo "ERROR(main): $i is not a symbolic link "
        echo
        exit
      fi
      if [[ ! -w $i ]]
      then
        echo "ERROR(main): $i is not writable"
        echo
        exit
      fi
done


for i in $(grep sybase add_temp_files.sql | grep -v "\-\-" | tr -s " "  | tr -d "'" | tr -d
"," | tr -d ";" | cut -d" " -f3)
    do
      if [[ ! -L $i ]]
      then
        echo "ERROR(temp): $i is not a symbolic link "
        echo
        exit
      fi
      if [[ ! -w $i ]]
      then
        echo "ERROR(temp): $i is not writable"
        echo
        exit
      fi
done

fi

if [[ ! -L /sybase_stage/dbgenrf ]]
then
   echo "ERROR: /sybase_stage/dbgenrf is not a symbolic link "
```

```
   echo
   exit
fi

# we have checked the commandline args for reasonableness, now
# describe the scope of the run and its configuration
echo
echo
echo  "  $(date)   "
echo
if  [[ -e benchmark_disks ]]
then
      echo benchmark disks
      echo ---------------
      cat benchmark_disks
else
      echo
      echo "   ------  benchmark_disks file does not exist  ------"
      echo
      exit
      echo
fi

echo
case $scope in
         q* )
              echo "DOING QUERY $query_num  with: " ;;
        all )
              if [[ $real_scope = lrun1 ]]
              then
                   echo "DOING LOAD, FOLLOWED BY POWER, FOLLOWED
BY THROUGHPUT with:"
              else
                   echo "DOING FULL AUDIT TEST (load, plus 2 full runs) with:
"
              fi ;;
       alltr )
              if [[ $real_scope = lrun1 ]]
              then
                   echo "DOING LOAD, FOLLOWED BY POWER, FOLLOWED
BY THROUGHPUT WITH TRAILING REFRESHES and:"
              else
                   echo "DOING FULL AUDIT TEST (load, plus 2 full runs) WITH
TRAILING REFRESHES and: "
              fi ;;
       mall )
              echo "DOING FULL AUDIT TEST FOR MULTIPLEX
ENVIRONMENT, ASSUMING IQ IS ALREADY STARTED" ;;
       load )
              echo "ONLY DOING LOAD (and create database) with: " ;;
      power )
              echo "ONLY DOING ONE POWER TEST with:" ;;
    refresh )
              echo "ONLY DOING ONE REFRESH PAIR with: " ;;
     tpworf )
              echo "ONLY DOING ONE THROUGHPUT TEST WITHOUT
REFRESHES with:" ;;
  throughput )
              echo "ONLY DOING ONE THROUGHPUT TEST with:" ;;
 throughputtr )
              echo "ONLY DOING ONE THROUGHPUT TEST WITH
TRAILING REFRESHES and:" ;;
 lthroughput )
              echo "ONLY DOING LOAD AND ONE THROUGHPUT TEST
with:" ;;
       run1 )
              echo "ONLY DOING ONE POWER FOLLOWED BY ONE
THROUGHPUT TEST with:" ;;
      lrun1 )
              echo "ONLY DOING LOAD, POWER FOLLOWED BY ONE
THROUGHPUT TEST with:" ;;
    stream0 )
              echo "ONLY DOING 22 SINGLE-USER QUERIES (NO
REFRESHES):" ;;
    restart )
              echo "ONLY RESTARTING IQ with:" ;;
          * )
              echo "ERROR:  scope (=$scope) must be one of"
              echo "          (all,restart,load,power,refresh,stream0,throughput,run1)"
              echo
              exit ;;
esac

echo " "



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

if [[ $scope = all || $scope = alltr || $scope = load || $scope = mall ]]
then
    cd dbgen_files
    s=$(show_dbgen_size)
    cd $cur_dir
    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* ]]  # every other scope restarts IQ; maybe some scopes should
not restart ???
then
    echo "      MPSSHEAP = $mpssheap    (for this node)"
    echo "      MPSSSTACK = $mpssstack  (for this node)"
    echo
    iqmc=$(grep iqmc tpch.cfg | grep -v "#" | tr -s ' '| cut -d' ' -f 2)
    iqmc_gt=$(round_to_tenths.pl $(my_calc.pl "$iqmc/1000"))
    iqtc=$(grep iqtc tpch.cfg | grep -v "#" | tr -s ' '| cut -d' ' -f 2)
    iqtc_gt=$(round_to_tenths.pl $(my_calc.pl "$iqtc/1000"))
    if [[ $iqmc_gt = *.0 ]]
    then
      iqmc_g=$(echo $iqmc_gt | cut -d '.' -f1)
    else
      iqmc_g=$iqmc_gt
    fi
    if [[ $iqtc_gt = *.0 ]]
    then
      iqtc_g=$(echo $iqtc_gt | cut -d '.' -f1)
    else
      iqtc_g=$iqtc_gt
    fi

    echo "      iqmc = $iqmc MB"
    echo "      iqtc = $iqtc MB"
    echo "      iqmt = $(grep iqmt tpch.cfg | grep -v "#" | tr -s ' '| cut -d' ' -f 2)
threads"
else
    echo "      MPSS values unchanged from existing values"
    echo "      iqmc unchanged from existing value"
    echo "      iqtc unchanged from existing value"
    echo "      iqmt unchanged from existing value"
fi

if [[ $scope = all || $scope = alltr || $scope = run1 || $scope = throughput || $scope
= throughputtr || $scope = tpworf || $scope = mall ]]
then
  echo "      num throughput streams = $nqs    (compliant minimum for this scale
factor: $cmin)"
  echo "      max connections = $(grep gm   tpch.cfg|cut -d' ' -f 2)  (for this node)
"
fi
```

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

     echo

fi


# seed is irrelevant for load, refresh and restart

if [[ $scope != load && $scope != refresh && $scope != restart ]]
then
  if [[ $input_seed == 1 ]]
  then
       seed=$input_seed        # testing for seed = 1 is used later when
gen_streams.ksh is invoked
       echo "      using newly gererated seed"
  elif  [[ $input_seed == 0 ]]
  then
       caution=yes     # print message below warning that this is not a auditable
run
       if [[ -f stream0.sql ]]
       then
           existing_seed=$(grep "as a seed" stream0.sql | cut -d 'g' -f2 | cut -d 'a'
-f1)
           seed=$existing_seed
           echo  "      using existing seed = $seed"
       else
           echo "ERROR: Cannot use existing seed when stream0.sql does not
exist"
           exit
       fi
  else
     caution=yes     # print message below warning that this is not a auditable run
     echo "      using $input_seed as seed"
     seed=$input_seed
  fi
fi

echo

echo "      Using the following devices"
echo "      ---------------------------"
echo "      $(grep "iq path" create_database.sql | grep -v "-" | tr -d """  | tr -s ' ' | cut
-d' ' -f3)"

for i in $(grep main add_main_files.sql  | grep -v alter  | tr -d "," | tr -d """ | tr -d
";" | tr -s " " | cut -d' ' -f3)
do
  echo "      $i"
done
echo

echo "      Using the following temporary devices (on this node)"
echo "      ----------------------------------------------------"

echo "      $(grep "temporary path" create_database.sql | grep -v "-" | tr -d """  | tr
-s ' '| cut -d' ' -f3)"

for i in $(grep temp add_temp_files.sql  | grep -v alter  | tr -d "," | tr -d """ | tr -d
";" | tr -s " " | cut -d' ' -f3)
do
  echo "      $i"
done
echo

((main_link_count=$(ls -l /sybase2/M* | wc -l)+0))
((main_db_count=$(grep main add_main_files.sql  | wc -l)+1))
if (( main_link_count > main_db_count ))
then
  echo "WARNING: There are more main links($main_link_count) in /sybase2 "
  echo "         than                                          "
  echo "         main devices($main_db_count) referenced in create_database.sql
and add_main_files.sql"
  echo
fi

((temp_link_count=$(ls -l /sybase2/T* | wc -l)+0))
((temp_db_count=$(grep temp add_temp_files.sql  | wc -l)+1))
if (( temp_link_count > temp_db_count ))
then
  echo "WARNING: There are more temp links($temp_link_count) than temp
devices($temp_db_count) (on this node)"
  echo
fi

# show which optional indexes are being used
# may not be accurate if a load is not being done and
# create_tables has been changed since the last load
ps_hg_index=$(grep "l_partsupp_hg" create_tables_${big}int$
{partitioned}.sql)
p_hg_index=$(grep "l_partkey_hg" create_tables_${big}int${partitioned}.sql)
echo "      optional lineiten indexes:   "
echo "      $ps_hg_index"
echo "      $p_hg_index"
echo


# show relevant /etc/system values
adb -k >/tmp/out <<END
tune_t_fsflushr/D
END
echo "      $(grep tune_t_fsflushr /tmp/out | tr ":" " " | 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)"

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

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


if [[ $scope = q* ]]
then
     echo "      options.sql unchanged "
else  # going to restart iq and rerun options.sql
     echo
     cat options.sql
fi

echo
iq_version=$(ls $HOME | grep IQ-)
# iq_version=$(my_transform.pl  "$iq_version" 's/ASIQ/IQ/') not needed in
IQ15
```

```
iqv=$(ls $HOME | grep IQ- | grep -v lic | cut -d'-' -f2)
echo Running $iq_version
echo

echo Load Order:
echo

if [[ $scope != mall ]]
then
  if [[ $RESTART_IQ_AFTER_LOAD = yes ]]
  then
      grep dbisql $0 | grep -v grep | grep load  | grep restart
  else
      grep dbisql $0 | grep -v grep | grep load  | grep optimized
  fi
else
  grep dbisql mpx | grep -v grep | grep load
fi
echo

if [[ $scope = mall ]]
then
  mpx_streams=$(grep dbisqlc mpx_throughput.bash | grep -v "#" | wc -l)
  if [[ $nqs -ne $mpx_streams ]]
  then
    echo "ERROR: Number of query streams($nqs) does not match"
    echo "       the number of streams($mpx_streams) exececued in"
    echo "       mpx_throughput.bash"
    echo
    exit
  fi
fi
echo

echo
echo "     the following are run immediately before stream0 or stream0 RF1    "
echo "     ------------------------------------------------------------------   "
grep  Junk  $0  | grep -v grep  | grep -v "#"
echo
echo
if [[ $RESTART_IQ_AFTER_LOAD = yes ]]
then
    echo
    echo "  Note:  this run will restart IQ after the load and auditor verification
"
    echo "        scripts complete, as indicated by the negative number of disks
"
    echo
    echo
fi

echo
echo "html query plans from gen_streams_new.ksh"
echo "-----------------------------------------"
 grep DSS_QUERY  gen_streams_new.ksh  |  grep export | grep -v "#"
echo
echo "    set DSS_QUERY in gen_streams_new.ksh to identify query plans     "
echo

grep -v "#" comments.txt
echo
   cp comments.txt /tmp/comments.txt    # this makes the /tmp copy the one
used for
                                # reporting this run.
echo


if   [[  -f $upd_tput_fname  ]]
then
    echo waiting -----\>\>\> $(grep  sleep  $upd_tput_fname | tr -d "'"  | \
        tr -d ";" | tr  -s  " "  | cut -d " "  -f3) \<\<\<----- \
 seconds after throughput query streams begin before starting refreshes
fi

echo
echo "   using create_tables_${big}int${partitioned}.sql "
echo


if [[ $caution = yes ]]
then
  banner "CAUTION:"  ; banner "this is"; banner "not a "; banner
"AUDITABLE ";  banner "RUN"
fi

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


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

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

rm JuNk   # file containing redirected output of restarting IQ after load


if [[ $scope = mall || $scope = all || $scope = alltr ]]
then
  rm -f ${audit_file_dir}* 2&>/dev/null
fi

if [[ $scope = mall ]]
then
  mpx $sf $mpssheap $mpssstack $nqs $nd $ds $sc $input_seed $iqmc_g
$iqtc_g $iqv $iq_version
  exit
fi

echo

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

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

echo
echo

# stop & restart IQ in all other 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
iq_running=`ps -eaf | grep iq | grep -v grep | wc -l`
if ((iq_running > 0))
then
    echo "Stopping IQ"
    #dbstop -c "DSN=tpch" -y     # replace this with
```

```
    stop_iq  -stop all
fi

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

    if [[ -f /sybase2/firstmain.iq ]]
    then
      #  rm -f /export/home/sybase/new_firstmain.iq
      rm -f /sybase2/firstmain.iq

      if [[ -f /sybase2/firstmain.iq ]]
      then
         echo
         echo unable to remove /sybase2/firstmain.iq
         echo
         exit
      fi

      echo "removed  /sybase2/firstmain.iq"
    fi

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

      # there have been times 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 times 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  1 Seconds
    echo "  "
    sleep 1

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

    start_iq @utility.cfg


    # start_iq @/export/home/sybase/IQ-15_0/demo/iqdemo.cfg
/export/home/sybase/IQ-15_0/demo/iqdemo.db

    #start_iq @utility.cfg /export/home/sybase/IQ-15_0/demo/iqdemo.db

    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 1
    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_iq @tpch.cfg /sybase2/tpch.db
echo "    IQ tpch database started: `date`   "
echo "  "


sleep 1

# add files to the database and create the 2 stored procs which control the refresh
operations
if [[ $scope = load || $scope = all || $scope = alltr ]]
then
    if [[ -e af.ooo  ]]
    then
       rm af.ooo
    fi

    dbisqlc -c "DSN=tpch" -q add_main_files.sql   >af.ooo

    if [[ -s af.ooo  ]]
    then
        echo "         main files added "
    else
        echo
```

```
        echo ERROR: alter database add main files  failed
        echo
        exit
    fi

    dbisqlc -c "DSN=tpch" -q add_temp_files.sql   >af.ooo
    if [[ -s af.ooo  ]]
    then
        echo "         temp files added "
    else
        echo
        echo ERROR: alter database add temp files  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

# set IQ options
if [[ -e o.ooo ]]
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 1

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

sleep 5

echo "Restarting IQ with tpch database and specified MPSS values: `date`   "
start_iq @tpch.cfg  /sybase2/tpch.db
echo "IQ restarted: `date`   "
echo "  "


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

# set $new_seed to the time at this instant
# if there is a load, new_seed will be reset to the end of the load time.
# without a load, new_seed gets it value here

new_seed=`date '+%m%d%H%M%S'`

if [[ $scope = load || $scope = all || $scope = alltr ]]
then
    sleep 1

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

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

    sleep 1

    #
    # Load the database
    #
    echo "  "

    show_iq_cpu  start_load  > start_load_cpu.out

    echo "   Load Started  `now_iq_format.bash`"  | tee start_load.out

    if [[ $RESTART_IQ_AFTER_LOAD = yes ]]
    then

      # use standard table load order (faster than optimized order)

      dbisqlc -c "DSN=tpch" -q    load_lineitem.sql > load_lineitem_restart.out &
      loadlpid=$!
      echo " "
      echo "      - lineitem   load started: `date` "

      dbisqlc -c "DSN=tpch" -q        load_region.sql    > load_region_restart.out
      echo "          - region        load completed: `date` "
```

```
    dbisqlc -c "DSN=tpch" -q       load_nation.sql    > load_nation_restart.out
    echo "         - nation       load completed: `date` "

    dbisqlc -c "DSN=tpch" -q       load_customer.sql  > customer_restart.out
    echo "         - customer     load completed: `date` "

    dbisqlc -c "DSN=tpch" -q       load_part.sql      > load_part_restart.out
    echo "         - part         load completed: `date` "

    dbisqlc -c "DSN=tpch" -q       load_supplier.sql  >
load_supplier_restart.out
    echo "         - supplier     load completed: `date` "

    dbisqlc -c "DSN=tpch" -q       load_partsupp.sql  >
load_partsupp_restart.out
    echo "         - partsupp     load completed: `date` "

    dbisqlc -c "DSN=tpch" -q       load_orders.sql    > load_orders_restart.out
    echo "         - orders       load completed: `date` "

    wait $loadlpid
    echo "      - lineitem   parallel load completed: `date` "

  else

    # use optimized table load order

    dbisqlc -c "DSN=tpch" -q   load_lineitem.sql  >
load_lineitem_optimized.out
    echo "         - lineitem load completed: `date` "

    dbisqlc -c "DSN=tpch" -q   load_region.sql    > load_region_optimized.out
    echo "         - region   load completed: `date` "

    dbisqlc -c "DSN=tpch" -q   load_nation.sql    > load_nation_optimized.out
    echo "         - nation   load completed: `date` "

    dbisqlc -c "DSN=tpch" -q   load_customer.sql  >
load_customer_optimized.out
    echo "         - customer load completed: `date` "

    dbisqlc -c "DSN=tpch" -q   load_orders.sql    > load_orders_optimized.out
    echo "         - orders   load completed: `date` "

    dbisqlc -c "DSN=tpch" -q   load_supplier.sql  >
load_supplier_optimized.out
    echo "         - supplier load completed: `date` "

    dbisqlc -c "DSN=tpch" -q   load_part.sql      > load_part_optimized.out
    echo "         - part     load completed: `date` "

    dbisqlc -c "DSN=tpch" -q   load_partsupp.sql  >
load_partsupp_optimized.out
    echo "         - partsupp load completed: `date` "

  fi

  echo
  echo "   Load Finished  `now_iq_format.bash` " | tee end_load.out
  new_seed=`date '+%m%d%H%M%S'`
  echo
  show_iq_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 `basename $0`
      echo
      exit
    fi
  fi

  # note: load has exited

  if [[ $scope = stream0 || $scope = power || $scope = tpworf || \
        $scope = throughput || $scope = run1 || $scope = all || \
        $scope = throughputtr || $scope = alltr          ]]
  then
    if [[ $seed = "1" ]]
    then
        echo "Generating $((nqs+1)) Query Streams "
        seed=$new_seed
        echo "Using the appropriate timestamp seed = $seed "
        gen_streams_new.ksh    $seed  $sf  $nqs
    else
      if [[ $input_seed = "1" ]]
      then
        echo  "Using existing seed = $seed"
      else
        echo  "Using provided seed = $seed"
        rm -f stream*.sql
        gen_streams_new.ksh    $seed  $sf  $nqs
      fi
    fi
    echo
  fi

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

    grep dbisqlc $0 | grep _start | grep -v grep

    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 Verification Scripts   `date` "
    echo

    copy_audit_db_data.bash start
  fi

  # delete all stream*.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 outfiles.


  # This is important because some of the report calculations
  # use all the stream*.out files.
  # If some of these files belong to prior runs, then some of
  # the report values will not be correct.

  # Also delete update_power.out and update_throughput.out so
  # that the stream0 and tpworf scopes will not report old
  # refresh information

  rm stream*.out

  echo
  # assume that there are always some stream?.out files
  echo "removed stream*.out"
  echo
  if rm update_power.out 2>/tmp/junk
```

```
then
    echo "removed update_power.out"
else
    echo "cannot remove update_power.out. file does not exist"
fi
echo
if rm update_throughput.out 2>/tmp/junk
then
    echo "removed update_throughput.out"
else
    echo "cannot remove update_throughput.out. file does not exist"
fi
echo
echo

# iq system monitoring (trace)
# iqsysmonitor.sh start
# echo "press any key to continue..."
# read xyzvar

echo "  scope = $scope    real_scope2 = $real_scope2  "

if [[ ( $scope = refresh || $scope = stream0 || $scope = power || $scope = run1 ||
$scope = all || $scope = alltr ) \
          && $real_scope2 != lthroughput ]]
then

    if [[ $RESTART_IQ_AFTER_LOAD = yes ]]
    then
        echo
        echo restarting IQ
        echo
        echo "     first shutdown IQ: `date`  "

        dbstop -c "DSN=tpch" -y

        echo
        echo

        sleep 60

        echo "     now restart IQ with tpch database and specified MPSS values:
`date`   "
        start_iq @tpch.cfg  /sybase2/tpch.db  > JuNk

        if  (( $(grep failed JuNk | wc -l) > 0 ))
        then
          echo
          echo "  --  IQ failed to restart "
          echo
          exit
        else
          echo
          echo "  IQ restarted after load: `date`   "
          echo
        fi
    fi

    echo

    # queries to prime the cache

    #dbisqlc -c "DSN=tpch" -q q6.sql  > Junk6.out
    #dbisqlc -c "DSN=tpch" -q q14.sql > Junk14.out
    #dbisqlc -c "DSN=tpch" -q q2.sql  > Junk2.out


    if [[ $scope = power || $scope = run1 || $scope = all || $scope = alltr ]]
    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 ||
$scope = alltr ]]
    then
        show_iq_cpu  start_power  > start_power_cpu.out
        show_iq_cpu  start_refresh1  > start_refresh1_cpu.out
        echo
        echo "     Start RF1   `date`  "
        dbisqlc -c "DSN=tpch" -q rf1.sql > rf1.out

        show_iq_cpu  end_refresh1  > end_refresh1_cpu.out
    fi


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

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


    if [[ $scope != stream0 ]]  # do refresh2
    then
         show_iq_cpu  start_refresh2  > start_refresh2_cpu.out
         echo
         echo "     Start RF 2   `date` "
         dbisqlc -c "DSN=tpch" -q rf2.sql > rf2.out
         echo "     End RF 2    `date`  "
         show_iq_cpu  end_refresh2  > end_refresh2_cpu.out
         show_iq_cpu  end_power    >end_power_cpu.out
         cat rf1.out  > update_power.out
         cat rf2.out >> update_power.out
    else
         echo
         echo "End Stream0   `date`"
         echo "-----------         "
         echo will now make report
         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

        mv /streamtmp/stream0.out .      # non timed interval

        tpch_power_response_times.bash  $scope
        power=$(tpch_power.bash  $sf)
        echo
        echo "     Power = $power"
        echo
        echo
        ps -eaf | grep iq | grep -v grep | awk -F' ' '{print $7,$8}'
        echo
    fi
fi

if [[ $scope = throughput || $scope = run1 || $scope = all || $scope = tpworf ||
$scope = alltr || $scope = throughputtr ]]
then
    show_iq_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<=nqs))  # run all query streams concurrently
    do
      dbisqlc -c "DSN=tpch" -q stream${i}.sql > /streamtmp/stream${i}.out &
      #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

    if [[ $scope = throughputtr || $scope = alltr ]]  # wait if it's only throughput
with trailing refreshes
    then
      echo "     Waiting for query streams to complete... `date`"
      wait   # for query streams
      echo "     All query streams have completed  `date`   "
    fi

    if [[ $scope != tpworf ]]  # tpworf does a throughput without refreshes
    then
        echo "     Start Refresh Streams  `date` "
        echo "  "

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

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

    wait   # for everything

    if [[ $scope != tpworf ]]  # tpworf does a throughput without refreshes
    then
        echo "     All query and refresh streams have completed  `date`   "
        echo
        echo "End Run 1 Throughput `date`  "
        echo "--------------------          "
        echo
    else
        echo "     All query streams have completed  `date`   "
        echo
        echo "End Throughput (without refreshes) `date`  "
        echo "--------------------          "
        echo
    fi
    ((i=1))
    while ((i<=nqs))  # copy throughput output files back to run/scripts
    do
      mv /streamtmp/stream${i}.out  .
      ((i=i+1))
    done

    throughput_interval=$(tpch_throughput_interval.bash  $nqs)
    throughput=$(tpch_throughput.bash  $sf  $nqs)
    echo
    echo "     Throughput Interval = $throughput_interval secs "
    echo
    echo "     Throughput = $throughput"
    echo
    if [[ $power != 0 ]]
    then
        composite=$(my_calc.pl  "sqrt($power*$throughput)")
        echo "     Composite = $composite"
        echo
    fi
    echo
    ps -eaf | grep iq | grep -v grep | awk -F' ' '{print $7,$8}'
    echo
    show_iq_cpu  end_throughput  >end_throughput_cpu.out
fi


if [[ $sf = .1 ]]     # this was intended for a quick Niagara run with scope =  all
then                  # why not use lrun1 ???? and forget this
  exit
fi

if [[ $scope = stream0 || $scope = power || $scope = tpworf ||
     $scope = throughput || $scope = throughputtr || $scope = run1 || $scope =
lrun1 ||\
     $scope = all || $scope = alltr       ]]
then
    dayHr=`date '+%m%d%H'`
    echo
    echo
    if [[ -z $real_scope2 ]]     # non_null means lthroughput
    then
      if [[ -z $real_scope ]]   # non_null means lrun1
      then
        if [[ $scope = stream0 ]]
        then
          composite=0
        fi
        if [[ $scope = power ]]
        then
          composite=$power
        fi
        if [[ $scope = throughput || $scope = throughputtr || $scope = tpworf ]]
        then
          composite=$throughput
        fi
        rpt_file_name="mrun1_${scope}_${sf}g_${nqs}s_${iqmc_g}m_$
{iqtc_g}t_${dayHr}_${iqv}_1n_$(round_to_tenths.pl ${composite}).r"
        echo "Producing ${rpt_file_name}"
        tpch_report.bash $scope $sf $mpssheap $mpssstack                    \
                  run1_${scope}_${sf}g_${nqs}s_${iqmc_g}m_${iqtc_g}t_$
{dayHr}_${iqv}_1n.r \
                  $seed $nqs $nd $ds $sc                                    \
                > ${rpt_file_name}

        cp ${rpt_file_name} /sybase_stage/results
      else
        # lrun1
        rpt_file_name="mrun1_${real_scope}_${sf}g_${nqs}s_${iqmc_g}m_$
{iqtc_g}t_${dayHr}_${iqv}_1n_$(round_to_tenths.pl ${composite}).r"
        echo "Producing ${rpt_file_name}"
        tpch_report.bash lrun1 $sf $mpssheap $mpssstack                     \
                  run1_${scope}_${sf}g_${nqs}s_${iqmc_g}m_${iqtc_g}t_$
```

```
{dayHr}_${iqv}_1n.r \
                    $seed $nqs $nd $ds $sc                                  \
                  > ${rpt_file_name}

        cp ${rpt_file_name} /sybase_stage/results

      fi
    else
      # lthroughput
      composite=$throughput
      rpt_file_name="mrun1_${real_scope2}_${sf}g_${nqs}s_${iqmc_g}m_$
{iqtc_g}t_${dayHr}_${iqv}_1n_$(round_to_tenths.pl ${composite}).r"
      echo "Producing ${rpt_file_name}"
      tpch_report.bash lthroughput $sf $mpssheap $mpssstack
\
                  run1_${scope}_${sf}g_${nqs}s_${iqmc_g}m_${iqtc_g}t_$
{dayHr}_${iqv}_1n.r \
                  $seed $nqs $nd $ds $sc                                  \
                  > ${rpt_file_name}

        cp ${rpt_file_name} /sybase_stage/results

    fi
    echo
fi

# iq system monitoring (trace)
#iqsysmonitor.sh stop
#echo "press any key to continue..."
#read xyzvar

if [[ $scope = all || $scope = alltr ]]
then
    echo
    ((i=0))
    while ((i<=nqs))  # move the streamX.out files to audit naming standard
    do
      mv stream${i}.out /sybase_stage/results/m1s0${i}q.out
      ((i=i+1))
    done

    mv update_power.out /sybase_stage/results/m1s00rf.out
    mv update_throughput.out /sybase_stage/results/m1s01rf.out

    echo "Moved  *.out files to audit naming standard in /sybase_stage/results"
    echo
    echo "Need To Copy all audit required files to audit dir --- NOT DONE
YET"
    echo
    echo "FINISHED Run1  "
    echo

# ---------------------------------------------------------------------


    if [[ $real_scope = lrun1 || $real_scope2 = lthroughput ]]
    then
        echo
        if [[ $real_scope = lrun1 ]]
        then
          real_scope_var=lrun1
        else
          real_scope_var=lthroughput
        fi
        echo Nothing More to Do with $real_scope_var
        echo
        exit
    fi

    echo
    echo STARTING RUN 2

    echo "Start Run 2 Power Test   `date` "
    echo "----------------------          "

    # Start the RF Stream in the Background

    show_iq_cpu  start_power  > start_power_cpu.out
    show_iq_cpu  start_refresh1  > start_refresh1_cpu.out
    echo
    echo "     Start RF1 in the background   `date` "
    dbisqlc -c "DSN=tpch" -q rf1.sql > rf1.out

    show_iq_cpu  end_refresh1  > end_refresh1_cpu.out

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

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

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

###  Finished Query Stream

    # Remove the lock file so that the RF will continue with RF2

    show_iq_cpu  start_refresh2  > start_refresh2_cpu.out
    echo
    echo "     Start RF 2   `date` "
    dbisqlc -c "DSN=tpch" -q rf2.sql > rf2.out

    echo "     End RF 2    `date`  "
    show_iq_cpu  end_refresh2  > end_refresh2_cpu.out
    show_iq_cpu  end_power    >end_power_cpu.out
    cat rf1.out  > update_power.out
    cat rf2.out >> update_power.out

    mv /streamtmp/stream0.out .      # non timed interval

    echo
    echo "End Run 2 Power  Test    `date` "
    echo "----------------------          "
    echo
    echo
    tpch_power_response_times.bash  $scope
    power=$(tpch_power.bash  $sf)
    echo
    echo "     Power = $power"
    echo
    echo
    ps -eaf | grep iq | grep -v grep | awk -F' ' '{print $7,$8}'
    echo
    show_iq_cpu  start_throughput  >start_throughput_cpu.out
    echo
    echo "Start Run 2 Throughput `date`  "
    echo "----------------------          "
    echo "  "
    echo "     Start Query Streams  `date` "
    echo "  "

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

    if [[ $scope = throughputtr || $scope = alltr ]]  # wait if it's only throughput
with trailing refreshes
    then
```

```
      wait   # for query streams
      echo "     All query streams have completed  `date`   "
    fi

    echo Start the refresh streams  `date`
    echo

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

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

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

    ((i=1))
    while ((i<=nqs))  # copy throughput output files back to run/scripts
    do
      mv /streamtmp/stream${i}.out  .
      ((i=i+1))
    done

    echo
    echo "End Run 2 Throughput `date`  "
    echo "--------------------          "
    echo
    echo
    throughput_interval=$(tpch_throughput_interval.bash  $nqs)
    throughput=$(tpch_throughput.bash  $sf  $nqs)
    echo
    echo "     Throughput Interval = $throughput_interval secs "
    echo
    echo "     Throughput = $throughput"
    echo
    composite=$(my_calc.pl  "sqrt($power*$throughput)")
    echo "     Composite = $composite"
    echo
    echo

    ps -eaf | grep iq | grep -v grep | awk -F' ' '{print $8,$9}'
    show_iq_cpu  end_throughput  >end_throughput_cpu.out

###

    echo

    dayHr=`date '+%m%d%H'`
    echo
    rpt_file_name="mrun2_${scope}_${sf}g_${nqs}s_${iqmc_g}m_$
{iqtc_g}t_${dayHr}_${iqv}_1n_$(round_to_tenths.pl ${composite}).r"
    echo "Producing ${rpt_file_name}"

    tpch_report.bash $scope $sf $mpssheap $mpssstack                          \
                run2_${scope}_${sf}g_${nqs}s_${iqmc_g}m_${iqtc_g}t_$
{dayHr}_${iqv}_1n.r \
                $seed $nqs $nd $ds $sc                                    \
              > ${rpt_file_name}

        cp ${rpt_file_name} /sybase_stage/results

    ((i=0))
    while ((i<=nqs))  # move the streamX.out files to audit naming standard
    do
      cp stream${i}.sql /sybase_stage/results/stream${i}.sql
      cp stream${i}.out /sybase_stage/results/m2s0${i}q.out
      ((i=i+1))
    done

    cp /sybase2/tpch.iqmsg /sybase_stage/results/tpch.iqmsg

    cp qparm*.txt /sybase_stage/results

    # copy these rather than move so that we can make reports
    # for debug purposes if necessary

    cp update_power.out /sybase_stage/results/m2s00rf.out
    cp update_throughput.out /sybase_stage/results/m2s01rf.out

    echo "Copied *.out files to audit naming standard in /sybase_stage/results"
    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

    copy_audit_db_data.bash end

    prtdiag -v > /sybase_stage/results/prtdiag.out

    for i in $(ls $IQDIR15/lib64/*so*)
    do
      strings ${i} > /sybase_stage/results/strings/`basename ${i}`.strings
    done

    strings $IQDIR15/bin64/iqsrv* > /sybase_stage/results/strings/iqsrv.strings

    tar cf /sybase_stage/results/strings.tar /sybase_stage/results/strings/*

    tar cf /sybase_stage/results/outfiles.tar /sybase_stage/results/*out
/sybase_stage/results/*.r /sybase_stage/results/tpch.iqmsg
/sybase_stage/results/qparm*.txt

    gzip -f /sybase_stage/results/*.tar

    echo "FINISHED Run2 "
fi
############zzzzzz
```

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

Sybase pricing:

For:

Quotation for Software and Support

Company Sybase Inc.
Contact Prasanta Gosh
Phone 925-236-5776
Fax
Address Sybase Drive, Dublin CA 94568

SYBASE Sales Rep: Pat Maloney
Phone: 925-236-5079
Fax: 925-236-6178
Sybase Inc. 1 Sybase Drive, Dublin, CA 94568

Catalogue Number	Product Description	License Type	Machine	P/S	List Price Per Unit	Quantity	Price	Discount	Extended Price
12193	Sybase IQ Single App Svr, per cpu core	CP	Sun	P	2,595	8	20,760	1,038	19,722.00
98477	3 yr support Single App Svr, per cpu core				1,713	8	13,704	685	13,018.80
	Discounts:								
	5% (if total undicounted order > $25,000)								
	10% (if total undicounted order > $50,000)								

Quote Date: 12/01/2009
Valid thru:

Total 32,740.80
License + 3 year support

Sales Quotation

Quote Number: T-US-1559714-B

Quote Date: 11/12/09

Customer : ROBERT HOOPER
AT&T CORPORATION
2301 W 120TH ST
HAWTHORNE CA 90250
Tel / Fax : 9733608737 / 9733608737

Sun : Wesley Kenison
Sun Microsystems, Inc.
500 Eldorado Blvd
Broomfield Colorado 80021
Tel / Fax : 303-272-5183/

We are pleased to quote as follows:

Validity Period	Credit Terms	Shipping Terms
60 Days	NET 30 FRM INV DATE	Origin

Item	Product Number	Description	Qty	Unit List Price	Disc	Unit Net Price	Extended Net Price
	Quote requested by Nobel Shelby. Contract # AR-50958						
1	Config ID 7960173	Configuration: X4270-S1-AA	1	$33,038.00	N/A	$27,091.16	$27,091.16
1.1	X4270-S1-AA	Sun Fire X4270 x64 Server: 2.5-inch HDD base chassis package including motherboard, no DVD, 1 x PSU, redundant fans and Service Processor for Factory Integration. RoHS-6.	1	$2,695.00	18.00%	$2,209.90	$2,209.90
1.2	X311L	Localized Power Cord Kit North American/Asian This Product is Hazard Class Y, RoHS compliant.	1	N/C	N/A	N/C	N/C
1.3	5868A	8 GB Memory kit DDR3-1066 Registered ECC DIMMs (1 x 8 GB) for Sun Fire X4170, X4270 & X4275 x64 servers. RoHS-6. For Factory Integration Only.	15	$1,050.00	18.00%	$861.00	$12,915.00

Sales Quotation

Quote Number: T-US-1559714-B

Quote Date: 11/12/09

Item	Product Number	Description	Qty	Unit List Price	Disc	Unit Net Price	Extended Net Price
1.4	RB-SS2CF-146G10K	146GB 10K RPM 2.5" SAS hard disk drive with Marlin bracket. RoHS-6. (ATO)	1	$329.00	18.00%	$269.78	$269.78
1.5	RA-ST2CF-32G2SSD	2.5" 32GB SATA SSD (Type: SLC) with Marlin bracket, RoHS-6 Compliant. For Factory Integration Only	8	$1,199.00	18.00%	$983.18	$7,865.44
1.6	SG-PCIE8SAS-I-Z	Sun StorageTek (TM) 8-Port internal SAS PCI-Express LSI 3081E Host Bus Adapter with RAID 0, 1, 1E support. RoHS-6. XATO.	1	$249.00	18.00%	$204.18	$204.18
1.7	5861A	1 Intel Xeon Model Number X5570 Quad-Core (2.93GHz/95W) Processor without Heatsink for Sun Fire X4170 & Sun Fire X4270 & Sun Fire X4275 Servers. RoHS-6. For Factory Integration Only.	2	$2,199.00	18.00%	$1,803.18	$3,606.36
1.8	6334A	Power supply unit filler panel for Sun Fire X4240/X4270/X4275/X4440/X4450 x64 servers. XATO. RoHS-6.	1	N/C	18.00%	N/C	N/C
1.9	6331A	Drive bay filler panel for Sun Fire X4140/X4170/X4240/X4270/X4440/X4150 /X4450/Sun Blade X6270 x64 servers. RoHS-6. XATO.	7	N/C	18.00%	N/C	N/C
1.10	6332A	DVD bay filler panel for Sun Fire X4140/X4240/X4440/X4150/X4170/X4250 /X4270/X4450 server. XATO. RoHS-5.	1	N/C	18.00%	N/C	N/C
1.11	5879A	Memory Filler Panel for Sun Fire X4170 and X4270; Sun Blade X6275 Server Module, Sun Blade X6270 Server Module. For Factory Integration Only. RoHS-6.	3	N/C	18.00%	N/C	N/C
1.12	5899A	CPU Heatsink for Sun Fire X4270 & X4275 Server. For Factory Integration Only. RoHS-6	2	N/C	18.00%	N/C	N/C
1.13	X5900A	Media and Documentation Kit for Sun Fire X4170, Sun Fire X4270 and Sun Fire X4275 x64 servers. X-Option.	1	$25.00	18.00%	$20.50	$20.50

List Price Total:	$33,038.00

Total:	$27,091.16

Sales Quotation

Quote Number: T-US-1559714-B

Quote Date: 11/12/09

WE CAN HELP YOU SAVE WHEN YOU FINANCE YOUR TECHNOLOGY SOLUTION THROUGH SUN MICROSYSTEMS GLOBAL FINANCIAL SERVICES (SMGFS). CHOOSE FROM A WIDE RANGE OF FLEXIBLE ,LOW RATE FINANCING AND LEASING PLANS. CONTACT US TODAY AT SMGFS_ Quote_ AMER @ sun. com

Sales Quotation

Quote Number: T-US-1559714-B

Quote Date: 11/12/09

THESE ARE THE GENERAL TERMS APPLICABLE TO YOUR ORDER. ADDITIONAL TERMS APPLY TO YOUR ORDER AND CAN BE FOUND AT http://www.sun.com/sales/salesterms/.

1. INTERPRETATION

The purpose of the General Terms is to create a single mechanism under which you and your Affiliated Companies, if any, ("Company") may form purchasing or other Agreements with Sun Microsystems and its Affiliated Companies ("Sun"). In the General Terms:

"Affiliated Company" means, in relation to either party, any entity: (a) which is owned 50% or more by that party; or (b) over which that party exercises management control; or (c) which is under common control with that party; or (d) which owns 50% or more of that party;

"Agreement" means each agreement entered into under the General Terms, comprising the General Terms and an Exhibit executed by Sun and Company referencing the General Terms;

"Confidential Information" means any information disclosed by one party to another under any Agreement which is, prior to or at the time of disclosure, identified in writing as confidential or proprietary;

"Equipment" means the hardware (including components), software media and spare parts listed in the standard product price lists published by Sun from time to time;

"Exhibit" means any exhibit to the General Terms as executed by the parties from time to time;

"IPR" means intellectual property rights, including patents, trademarks, design rights, copyrights, database rights, trade secrets and all rights of an equivalent nature anywhere in the world;

"Products" means Equipment or Software;

"Service Listing" means any offering in Sun's Enterprise Services Service List, which is located at *http://www.sun.com/service/servicelist* (a hard copy of each of which will be made available to

Company on request), together with such other standard service offerings as the parties may agree from time to time;

"Services" means the services described in any Service Listing or SOW;

"Software" means (i) any binary software programs listed in the standard price lists published by Sun from time to time, (ii) any Updates, and (iii) any related user manuals or other documentation;

"SOW" means any statement of work relating to Services;

"Sun Trademarks" means all names, marks, logos, designs, trade dress and other brand designations used by Sun in connection with Products and Services;

"Updates" means subsequent releases and error corrections for Software previously licensed, as listed in the standard price lists published by Sun from time to time.

2. CONFIDENTIAL INFORMATION

2.1 A party receiving Confidential Information ("the Recipient") may use it only for the purposes for which it was provided under the Agreement. Confidential Information may be disclosed only:) to employees or contractors obligated to the Recipient under similar confidentiality restrictions and in each case only for the purposes for which it was provided under the relevant Agreement.

2.2 The obligations set out in section 2.1 do not apply to information which: (a) is rightfully obtained by the Recipient without breach of any obligation to maintain its confidentiality; (b) is or becomes known to the public through no act or omission of the Recipient; (c) the Recipient develops independently without using Confidential Information of the other party; or (d) is disclosed in response to a valid court or governmental order, if the Recipient has given the other party prior written notice and provides reasonable assistance so as to afford it the opportunity to object.

3. RESTRICTED ACTIVITIES

3.1 Export laws. Products, Services and technical data delivered by Sun may be subject to US export controls or the trade laws of other countries. Company will comply with all such laws and obtain all licenses to export, re-export or import as may be required after delivery to Company. Company will not export or re-export to entities on the most current U.S. export exclusion lists or to any country subject to U.S. embargo or terrorist controls as specified in the U.S. export laws. Company will not use or provide Products, Services, or technical data for nuclear, missile, or chemical biological weaponry end uses.

3.2 Nuclear applications. Company acknowledges that Products and Services are not designed or intended for use in the design, construction, operation or maintenance of any nuclear facility.

4. SUN TRADEMARKS

4.1 Company may refer to Products and Services by their associated names, provided that such reference is not misleading and complies with Sun's Trademark and Logo Policies, which are located at *http://www.sun.com/policies/trademarks* (and a hard copy of which will be made available to Company on request).

4.2 Company may not remove or alter any Sun Trademarks, nor may it co-logo Products or Services. Company agrees that any use of Sun Trademarks by Company will inure to the sole benefit of Sun.

4.3 Company agrees not to incorporate any Sun Trademarks into Company's trademarks, service marks, company names, Internet addresses, domain names, or any other similar designations.

5. PUBLICITY

5.1 Sun may use Company's name in promotional materials, including press releases, presentations and customer references regarding the sale of Products or Services. These permissions are free of charge for worldwide use in any medium. Sun will obtain Company's prior approval for publicity that contains claims, quotes, endorsements or attributions by Company, such approval not to be unreasonably withheld.

6. INTELLECTUAL PROPERTY CLAIMS

6.1 Each party ("the Indemnifying Party") will defend or settle, at its option and expense, any legal proceeding brought against the other ("the Indemnified Party") to the extent that it is based on a claim that materials (which term includes Products) developed and provided by the Indemnifying Party infringe a third party's patent, trade secret or copyright. The Indemnifying Party will indemnify the Indemnified Party against all damages and costs attributable exclusively to such claim awarded by the court finally determining the case, provided that the Indemnified Party: (a) gives written notice of the claim promptly to the Indemnifying Party; (b) gives the Indemnifying Party sole control of the defense and settlement of the claim; (c) provides to the Indemnifying Party, at the expense of the Indemnifying Party, all available information and assistance; (d) does not compromise or settle such claim; and (e) is not in material breach of any Agreement.

6.2 If such materials or services are found to infringe, or in the reasonable opinion of the Indemnifying Party are likely to be the subject of a claim, the Indemnifying Party will at its option: (a) obtain for the Indemnified Party the right to use such materials; (b) replace or modify the materials so they become non-infringing; or (c) if neither (a) nor (b) is reasonably achievable, remove such materials and refund their net book value.

6.3 Neither party has any obligation to the extent any claim results from: (a) use of materials in combination with any third party equipment, software or data; (b) compliance by the Indemnifying Party with the designs or specifications of the Indemnified Party; (c) modification of materials other than at the direction of the Indemnifying Party; or (d) use of an allegedly infringing version of the materials, if the alleged infringement could have been avoided by the use of a different version made available to the Indemnified Party.

6.4 This section states the entire liability of each party (as Indemnifying Party) and the exclusive remedies of each party (as Indemnified Party) for claims that materials infringe a third party's IPR.

7. WARRANTY

7.1 Sun warrants Products and Services as provided at http://www.sun.com/service/support/warranty (the "Warranty Web Page") (a hard copy of which is available on request).

7.2 EXCEPT AS SPECIFIED IN THE WARRANTY WEB PAGE, ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OR CONDITION OF MERCHANTABILITY, SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT, ARE HEREBY EXCLUDED TO THE MAXIMUM EXTENT PERMITTED BY LAW.

8. LIMITATION OF LIABILITY

8.1 No limitation on certain categories of liability. Each party acknowledges the full extent of its own liability to the other, arising from: (a) death or personal injury resulting from negligent acts or omissions; (b) breach of any applicable license grant; or (c) claims for non payment; and the non-excludable statutory rights of consumers (for example, under laws providing for strict product liability) are not affected.

8.2 Limitations on other categories of liability. Subject to 7.1 above and to the extent not prohibited by applicable law: (a) each party's maximum aggregate liability for all claims relating to any Agreement, whether for breach of contract, breach of warranty or in tort, including negligence, will be limited to two million US dollars (U.S. $2,000,000); and (b) neither party will be liable for any indirect, punitive, special, incidental or consequential damages in connection with or arising out of the General Terms or any Agreement (including, without limitation, loss of business, revenue, profits, goodwill, use, data, electronically transmitted orders, or other economic advantage), however they arise, whether in breach of contract, breach of warranty or in tort, including negligence, and even if that party has previously been advised of the possibility of such damages.

8.3 Failure of essential purpose. Liability for damages will be limited and excluded, even if any exclusive remedy provided for in the Agreement fails of its essential purpose.

9. TERMINATION AND EXPIRATION

9.1 **T**ermination for cause. Either party may terminate the General Terms or any Exhibit immediately by written notice: (a) if the other party commits a non-remediable material breach of the General Terms or Exhibit (as the case may be); or (b) if the other party fails to cure any remediable material breach within thirty (30) days of being notified in writing of such breach.

9.2 Termination without cause. (a) Either party may terminate the General Terms immediately by written notice if no Exhibit is in effect. (b) Either party may terminate any Exhibit at any time upon expiration of ninety (90) days' written notice.

9.3 Actions following termination or expiration. On termination or expiration of the General Terms (for whatever reason), all Exhibits shall automatically terminate with immediate effect. Following termination or expiration of an Exhibit (for whatever reason), each party will deliver to the other any property of the other in its possession or control relating to that Exhibit, in good condition, reasonable wear and tear excepted.

9.4 Effect of termination. Neither party will be liable for any damages arising out of the termination or expiration of the General Terms or any Exhibit, provided that such termination or expiration will not affect any right to recover: (a) damages sustained by reason of material breach; or (b) any payments which may be owing in respect of any Agreement.

10. ASSIGNMENT AND SUBCONTRACTING

10.1 Neither party may assign or otherwise transfer any of its rights or obligations under the General Terms or any Exhibit without the prior written consent of the other party, which consent will not be unreasonably withheld, except that: (a) both parties may assign their right to receive payment; and (b) Sun may use subcontractors in the performance of its obligations, in which case Sun will remain responsible for the performance by such sub-contractors.

11. **DISPUTE RESOLUTION.** The parties will use reasonable efforts to resolve any dispute arising out of the General Terms or any Exhibit through a meeting of appropriate managers from each party. If the parties are unable to resolve the dispute, either party may escalate the dispute to its executives. If an executive level meeting fails to resolve the dispute within thirty (30) days after escalation, either party may seek any available legal relief. This provision will not affect either party's right to seek injunctive or other provisional relief at any time.

12. **GENERAL.** All disputes will be governed by the laws of the State of California The venue for litigation will be the appropriate courts located in the County of Santa Clara. Choice of law rules of any jurisdiction and the United Nations Convention on Contracts for the International Sale of Goods will not apply to any dispute under the Agreement. Force majeure. A party is not liable under any Agreement for non-performance caused by events or conditions beyond that party's reasonable control, if the party makes reasonable efforts to perform. This provision does not relieve either party of its obligation to make payments then owing. Notices. All written notices required by the General Terms or any Exhibit must be delivered in person or by means evidenced by a delivery receipt or acknowledgment and will be effective upon receipt. Notices communicated by electronic mail or facsimile will be deemed to be written. Relationship. Neither the General Terms nor any Agreement is intended to create a partnership, franchise, joint venture, agency, or a fiduciary or employment relationship. Neither party may bind the other party or act in a manner which expresses or implies a relationship other than that of independent contractor. **I**nvalidity. If any provision of the General Terms or any Agreement is held invalid by any law or regulation of any government or by any court or arbitrator, such invalidity will not affect the enforceability of other provisions. Survival. Rights and obligations under the General Terms and any Exhibit which by their nature should survive, will remain in effect after termination or expiration of the General Terms or the relevant Exhibit. No waiver. Any express waiver or failure to exercise promptly any right under the General Terms or any Exhibit will not create a continuing waiver or any expectation of non-enforcement. Modification. No modification to the General Terms or any Exhibit will be binding, unless in writing and manually signed by an authorized representative of each party. Entire agreement. Each Agreement constitutes the parties' entire agreement relating to its subject matter. It cancels and supersedes all prior or contemporaneous oral or written communications, proposals, conditions, representations and warranties and prevails over any conflicting or additional terms contained in any quote, purchase order, acknowledgment, or other communication between the parties relating to its subject matter during its term.

Quotation
Sun Microsystems, Inc.

Quote Nr.	US1071076
Quote Date	12-NOV-09
Quote valid until	
Duration/Period	12-DEC-09 To 11-DEC-12
Agreement Number	AR-50958:EU:123016

AT&T CORP
ROBERT, HOOPER

Email id - hooper@research.att.com
Phone No - 973-360-8737
Fax No -

THIS IS OUR QUOTATION TO YOU. IF YOU WISH TO BUY OUR PRODUCTS OR SERVICES, PLEASE SEND YOUR PURCHASE ORDER TO US USING THE CONTACT INFORMATION LISTED BELOW. TO RECEIVE COMPLETE AND TIMELY SERVICE, INCLUDE THE FOLLOWING ON YOUR ORDER: THE NUMBER OF THIS QUOTATION, YOUR INSTALLATION ADDRESS(ES), YOUR CONTACT NAME(S), YOUR PHONE NUMBER(S) AND EMAIL ADDRESS(ES). YOUR ORDER MUST BE MADE OUT TO US. THANK YOU.

PLEASE READ THE FOLLOWING:
THIS SUN QUOTATION AND ANY ORDER YOU SEND TO US FOR PRODUCTS OR SERVICES IS COVERED BY (1) THE TERMS OF ANY SALES CONTRACT YOU ALREADY HAVE WITH US, OR BY THE CONTRACT TERMS SET OUT ON OUR WEBSITE AT http://www.sun.com/sales/salesterms; AND (2) FOR SERVICES: (A) THE APPROPRIATE SERVICE LISTING(S) SET OUT ON OUR WEBSITE AT http://www.sun.com/service/servicelist; OR (B) THE APPROPRIATE STATEMENT OF WORK; OR (C) OTHER DOCUMENT(S) PROVIDED BY US DESCRIBING THE SERVICE(S) YOU WANT TO PURCHASE FROM US.

ALL YOUR ORDERS MUST REFER TO: (1) THE NUMBER OF THE SALES CONTRACT YOU ALREADY HAVE WITH US, OR (2) THIS SALES QUOTATION. BY SENDING AN ORDER TO US, YOU AGREE THAT OUR CONTRACT TERMS APPLY TO ANY ORDER WE ACCEPT. WE WILL SHOW THAT WE HAVE ACCEPTED YOUR ORDER BY: (1) SHIPPING THE PRODUCT; (2) STARTING TO PROVIDE THE SERVICE; OR (3) SENDING YOU A WRITTEN ACCEPTANCE. THIS QUOTATION WILL REMAIN IN EFFECT FOR THE PERIOD LISTED ABOVE, EXCEPT THAT WE CAN CHANGE THIS QUOTATION IF: (1) IT CONTAINS A TYPOGRAPHICAL ERROR, OR (2) THE AVAILABILITY OF PRODUCTS OR SERVICES CHANGE, OR (3) YOUR CREDIT STATUS WITH US CHANGES.

CHANGES TO ANY OF OUR CONTRACT TERMS OR TO THIS QUOTATION WILL TAKE EFFECT ONLY IF IN WRITING AND SIGNED BY OUR AUTHORIZED REPRESENTATIVE.

OUR EQUIPMENT OR PARTS OF IT MAY BE NEW OR USED, BUT ANY LIMITED WARRANTY WE GIVE WILL STILL APPLY.

IF THE SERIAL NUMBER OF A HARDWARE ITEM WAS NOT PROVIDED, THE PRICE LISTED ON THIS QUOTATION INCLUDES A FULL WARRANTY. WHEN THE SERIAL NUMBER OF THE HARDWARE ITEM IS PROVIDED, THE ACTUAL AMOUNT PAYABLE WILL BE ADJUSTED TO INCLUDE THE PRICE OF THE REMAINING WARRANTY FOR THAT HARDWARE ITEM.

IF YOUR PURCHASE ORDER REQUIRES THAT (1) SUN FEDERAL PROVIDE PERSONNEL WITH U.S. GOVERNMENT SECURITY CLEARANCES, OR (2) SUN FEDERAL OR ITS SUPPLIERS ACCESS CLASSIFIED INFORMATION OR SECURE FACILITIES, YOU MUST GIVE US A VALID DD FORM 254 OR OTHER APPLICABLE SECURITY SPECIFICATION(S). IF THE REQUIRED FORM OR SPECIFICATIONS ARE NOT GIVEN TO US, OR YOU DO NOT HAVE FACILITY CLEARANCE LEVELS TO GIVE THEM TO US, WE MAY NOT PROVIDE YOU WITH CLEARED PERSONNEL.

Total Quote Value (excluding taxes):	USD 844.20

AT&T CORP		Sun Microsystems, Inc.	
SIGNATURE	DATE	SIGNATURE	DATE
PRINTED NAME		PRINTED NAME	

Sun Microsystems, Inc.
4150 NETWORK CIRCLE
95054 SANTA CLARA
CA United States

ANY TAXES LISTED ON THIS QUOTATION ARE ESTIMATES ONLY AND THE ACTUAL TAX TO BE PAID WILL BE LISTED ON THE INVOICE WE SEND TO YOU. IF TAXES ARE NOT LISTED ON THIS QUOTATION, THEY ARE NOT INCLUDED IN THE TOTAL QUOTE VALUE. THE ACTUAL AMOUNT PAYABLE, INCLUDING ALL TAXES, WILL BE LISTED ON THE INVOICE WE SEND TO YOU.

Sun Contact : DELEO, JOHN R
Office: x63315/+1 732-537-3315
Fax:
Email: john.deleo@sun.com

Service Level Details:

Site Name:	AT&T CORP 200 LAUREL AVE A2-1A21 MIDDLETOWN NJ 07748-1914 United States

Service Line #	Service Item #	Service Description	Line Start Date	Line End Date	Currency	Total Net Price
1	GOLD-SYS-SVC	Gold system service plan	12-DEC-09	11-DEC-12	USD	844.20

Service Item #: GOLD-SYS-SVC [Gold system service plan]

Covered Product:												
#	Qty	Mktg part#	Description	Serial No. / Cust. Ref.	List Price	Discounts	%	Line Start Date	Line End Date	Warr (Y/N)	Warranty End Date	Total Net price
1	1	X4270-S1-AA	X4270 1 x Standard PSU	/	118.42	Ar-50958:Eu WTY_UP	64 45	12-DEC-09	11-DEC-12	Y	11-DEC-12	844.20

Billing Details:	
Total Period 1	USD 281.40
Total Period 2	USD 281.40
Total Period 3	USD 281.40
Total Quote Value:	USD 844.20

Sun's Purchase Order Guidelines
• Vendor Information (PO made out to Sun Microsystems, Inc. or Sun Federal)
• PO Number & Date
• Customer Bill-to Address
• Legal Company Name
• Payment Terms (Net 30)
• Fully funded for services
• Signed by Authorized Agent
• Buyers Name
• Phone Number
• Tax Status (if applicable)
• Billing Frequency