

# TPC Benchmark™ H Full Disclosure Report

---

---

## Sun Microsystems Sun Fire™ X4100 Server Using Sybase IQ 12.6 Single Application Server

Submitted for Review  
Report Date: Jun 23, 2006

TPC Benchmark H Full Disclosure Report

First Printing

---

---

© 2006 Sun Microsystems, Inc.

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

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

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

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

#### TRADEMARKS

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

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

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

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

Sybase IQ are registered trademarks of Sybase Inc.

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

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

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

The pricing information in this document is believed to accurately reflect prices in effect on Report Date: Jun 23, 2006. However, Sun Microsystems and Sybase Corporation provide no warranty on the pricing information in this document.

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

---



**Sun Fire™ X4100 Server  
with Sybase IQ 12.6 Single  
Application Server**

TPC-H Rev. 2.3.0

Report Date: Jun 23, 2006

Total System Cost

Composite Query per Hour Metric

Price/Performance

\$19057.05

**4132.1**  
QphH@100GB

**\$4.61**  
per QphH@100GB

Database Size

Database Manager

Operating System

Other Software

Availability Date

**100GB**

**Sybase IQ 12.6  
Single Application  
Server**

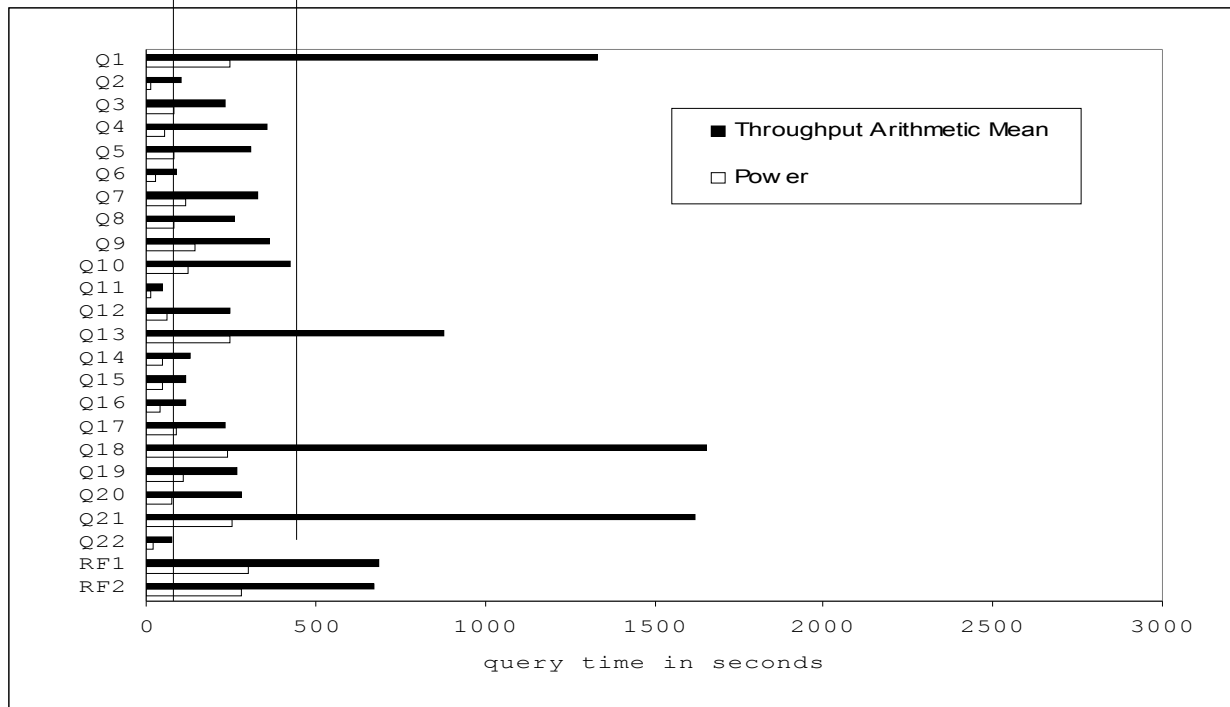
**Solaris 10**

**Solaris Volume  
Manager**

**Jun 23, 2006**

81.7 = power geometric mean

430.1 = throughput arithmetic mean



Database Load Time = 01:38:53

Load Includes Backup: N

Total Storage/Database Size=2.72

RAID (Base tables): RAID 1

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

RAID (All): N

**System Configuration:**

SunFire X4100 Server with

2 AMD Opteron 3.0GHz processors (each processor is 1 chip, 1 core, 1 thread)

16 GB memory

4 x 73 GB (10K RPM) internal SAS disks

**Total Storage:** 272 GB

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



**Sun Fire™ X4100 Server  
with Sybase IQ 12.6 Single  
Application Server**

TPC-H Rev. 2.3.0

Report Date: Jun 23, 2006

Description	Part Number	Source	Unit Price	Qty	Ext. Price	3 Yr. maint.
<b>Server Hardware</b>						
SunFire X4100 x64 Server: XATO base chassis	A64-ZD		1,495.00	1	1,495.00	
73GB 10K RPM 2.5" SAS hard disk drive	RA-SS2CD-73G10KZ		445.00	4	1,780.00	
AMD Opteron 256 (3.0 Ghz/1MB) processor	8035A-Z		995.00	2	1,990.00	
4GB Memory kit DDR1-400	8023A-Z		1,090.00	4	4,360.00	
X4100 Server upgrade to 3 year Gold Warranty Upgrade	W9D-A64-3G		2,772.00	1		2,772.00
Sun Discount				1	-673.75	-970.20
<i>Server Hardware Subtotal</i>					8,951.25	1,801.80
<b>External Storage</b>						
None						
<b>Server Software</b>						
Sybase IQ-M Single App Svr, per cpu core	12841		2,595.00	2	5,190.00	
Sybase IQ 3 Years Extended Support 24 x 7	98480		1,557.00	2		3,114.00
Sybase Discount				2	0.00	
<i>Server Software Subtotal</i>					5,190.00	3,114.00
<b>Total</b>					14,141.25	4,915.80
<b>3 Yr. Cost</b>					19,057.05	
<b>QphH@ 1000GB</b>					4,132.10	
<b>\$/QphH@ 100GB</b>					\$4.61	

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

Service for Sybase products is from Sybase Inc.

**Notes (Source):**

1. Sun Microsystems Inc.
2. Sybase Inc.

PriceQuotes provided in Appendix G

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

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



**Sun Fire™ X4100 Server  
with Sybase IQ 12.6 Single  
Application Server**

TPC-H Rev. 2.3.0

Report Date: Jun 23, 2006

**Numerical Quantities**

**Measurement Results:**

Database Scale Factor	= 100GB
Total Data Storage / Database Size	=2.72
Start of database load time	= 2006-05-31 05:31:09
End of database load time	= 2006-05-31 07:10:02
Database Load Time	= 01:38:53
Query Streams for Throughput Test	= 5
TPC-H Power	= 4408.6
TPC-H Throughput	= 3872.9
TPC-H Composite Query-per-Hour Rating (QphH@100GB)	= 4132.1
Total System Price Over 3 Years	= \$19057.05
TPC-H Price/Performance Metric (\$/QphH@100GB)	= \$4.61

**Measurement Intervals:**

Measurement Interval in Throughput Test (Ts)	= 10,225 seconds
--	------------------

**Duration of Stream Execution:**

Stream ID	Seed	Start Date	Start Time	End Date	End Time	Duration
Stream 0	531071002	2006-05-31	10:44:31	2006-05-31	11:31:21	46mins:50secs
Stream 1	531071003	2006-05-31	11:31:22	2006-05-31	13:57:42	2hrs:26mins:20secs
Stream 2	531071004	2006-05-31	11:31:22	2006-05-31	14:18:40	2hrs:47mins:18secs
Stream 3	531071005	2006-05-31	11:31:22	2006-05-31	14:02:27	2hrs:31mins:5secs
Stream 4	531071006	2006-05-31	11:31:22	2006-05-31	14:04:58	2hrs:33mins:36secs
Stream 5	531071007	2006-05-31	11:31:22	2006-05-31	14:21:47	2hrs:50mins:25secs
Refresh		2006-05-31	11:31:22	2006-05-31	14:09:45	2hrs:38mins:23secs



Sun Fire™ X4100 Server  
with Sybase IQ 12.6 Single  
Application Server

TPC-H Rev. 2.3.0

Report Date: Jun 23, 2006

TPC-H Timing Intervals (in seconds)

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Stream 00	248.8	14.3	79.0	55.3	80.5	27.9	115.2	82.8	145.7	125.0	12.7	59.6
Stream 01	706.1	110.7	222.3	338.2	288.0	54.9	248.8	224.8	364.2	441.9	34.8	140.4
Stream 02	2263.9	59.7	230.4	445.5	209.8	137.9	250.6	201.1	508.1	554.2	24.6	436.6
Stream 03	864.1	25.4	267.8	408.9	402.8	101.0	306.2	501.4	424.9	388.7	43.7	279.8
Stream 04	1692.0	193.8	233.6	269.0	425.1	54.5	428.6	214.8	368.3	555.8	89.4	291.7
Stream 05	1123.9	129.5	210.3	320.7	207.4	84.2	421.5	162.2	164.8	189.7	37.9	74.2
Minimum	706.1	25.4	210.3	269.0	207.4	54.5	248.8	162.2	164.8	189.7	24.6	74.2
Average	1330.0	103.8	232.9	356.5	306.6	86.5	331.1	260.9	366.1	426.1	46.1	244.6
Maximum	2263.9	193.8	267.8	445.5	425.1	137.9	428.6	501.4	508.1	555.8	89.4	436.6

	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream 00	244.2	50.5	44.7	38.7	87.6	239.2	112.0	77.4	257.4	19.6	303.1	284.1
Stream 01	510.3	109.8	74.8	53.6	169.0	1510.6	218.0	121.3	2805.1	30.7	783.6	699.3
Stream 02	699.4	123.3	166.9	121.7	476.1	1532.4	356.4	399.4	757.4	80.0	910.6	743.6
Stream 03	1055.8	165.6	105.9	121.0	215.3	1734.2	256.9	450.4	820.0	120.1	510.4	586.4
Stream 04	926.8	138.9	85.6	120.4	200.1	1587.3	272.3	318.1	679.0	70.6	489.4	592.7
Stream 05	1204.7	110.5	152.2	156.7	102.2	1891.5	239.4	105.6	3054.3	78.5	736.3	734.7
Minimum	510.3	109.8	74.8	53.6	102.2	1510.6	218.0	105.6	679.0	30.7	489.4	586.4
Average	879.4	129.6	117.1	114.7	232.5	1651.2	268.6	279.0	1623.1	76.0	686.1	671.3
Maximum	1204.7	165.6	166.9	156.7	476.1	1891.5	356.4	450.4	3054.3	120.1	910.6	743.6

# Table of Contents

1. General Items.....	12
1.1 Benchmark Sponsor.....	12
1.2 Parameter Settings.....	12
1.3 Configuration Diagram.....	13
2. Clause 1 Logical Database Design.....	14
2.1 Database Definition Statements.....	14
2.2 Physical Organization.....	14
2.3 Horizontal Partitioning.....	14
2.4 Replication.....	14
3. Clause 2 Queries and Refresh Functions.....	15
3.1 Query Language.....	15
3.2 Verifying Method for Random Number Generation.....	15
3.3 Generating Values for Substitution Parameters.....	15
3.4 Query Text and Output Data from Qualification Database.....	15
3.5 Query Substitution Parameters and Seeds Used.....	15
3.6 Query Isolation Level.....	16
3.7 Source Code of Refresh Functions.....	16
4. Clause 3 Database System Properties.....	17
4.1 ACID Properties.....	17
4.2 Atomicity.....	17
4.2.1 Completed Transaction.....	17
4.2.2 Aborted Transaction.....	17
4.3 Consistency.....	17
4.3.1 Consistency Test.....	18
4.4 Isolation.....	18
4.4.1 Read-Write Conflict with Commit.....	18
4.4.2 Read-Write Conflict with Rollback.....	18
4.4.3 Write-Write Conflict with Commit.....	18
4.4.4 Write-Write Conflict with Rollback.....	19
4.4.5 Concurrent Progress of Read and Write Transactions.....	19
4.4.6 Read-Only Query Conflict with Update Transaction.....	19
4.5 Durability.....	19
4.5.1 Failure of a Durable Medium.....	20
4.5.2 System Crash.....	20
4.5.3 Memory Failure.....	20
5. Clause 4 Scaling and Database Population.....	21
5.1 Ending Cardinality of Tables.....	21
5.2 Distribution of Tables and Logs Across Media.....	21
5.3 Database partition/replication mapping.....	22
5.4 RAID Feature.....	23
5.5 Modifications to the DBGEN.....	23
5.6 Database Load Time.....	23
5.7 Data Storage Ratio.....	23
5.8 Database Load Mechanism Details and Illustration.....	24
5.9 Qualification Database Configuration.....	24

6. Clause 5 Performance Metrics and Execution Rules.....	25
6.1 System Activity Between Load and Performance Tests.....	25
6.2 Steps in the Power Test.....	25
6.3 Timing Intervals for Each Query and Refresh Functions.....	25
6.4 Number of Streams for the Throughput Test.....	25
6.5 Start and End Date/Times for Each Query Stream.....	25
6.6 Total Elapsed Time of the Measurement Interval.....	25
6.7 Refresh Function Start Date/Time and Finish Date/Time.....	26
6.8 Timing Intervals for Each Query and Each Refresh Function for Each Stream.....	26
6.9 Performance Metrics.....	26
6.10 The Performance Metric and Numerical Quantities from Both Runs.....	26
6.11 System Activity Between Performance Tests.....	27
7. Clause 6 SUT and Driver Implementation.....	28
7.1 Driver.....	28
7.2 Implementation-Specific Layer.....	28
7.3 Profile-Directed Optimization.....	28
8. Clause 7 Pricing.....	29
8.1 Hardware and Software Used.....	29
8.2 Total Three Year Price.....	29
8.3 Availability Date.....	29
9. Auditor's Information and Attestation Letter.....	30
Appendix A. Solaris 9 and Sybase IQ 12.5 Parameters.....	31
Appendix B. Programs and Scripts.....	32
Appendix C. Query Text and Query Output.....	50
Appendix D. Seed and Query Substitution Parameters.....	59
Appendix E. Implementation-Specific Layer/Driver Code.....	61
Appendix F. Misc database scripts.....	63
Appendix G. Pricing information.....	65



June 22, 2006

Benchmark Sponsors:	Brad Carlile Director, Enterprise Benchmarking Sun Microsystems, Inc. 8305 S. W. Creekside Place Beaverton, OR 97008	Paul Kreneta Chief Technologist, Sybase IQ Sybase, Inc. 1 Sybase Drive Dublin, CA 94568
---------------------	--	---

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

Platform:	<b>Sun Fire X4100 Server</b>
Database Manager:	<b>Sybase IQ 12.6 Single Application Server</b>
Operating System:	<b>Solaris 10</b>

The results were:

CPU (Speed)	Memory	Disks	<b>QphH@100GB</b>
<b>One (1) Sun Fire X4100</b>			
2 x AMD Opteron (3.0 GHz)	16 GB Main	4 x 73 GB	<b>4132.1</b>

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:

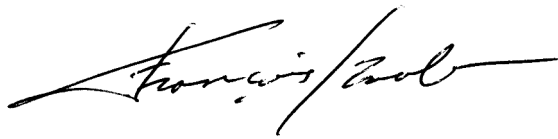
- The database records were defined with the proper layout and size
- The database population was generated using DBGEN
- The database was properly scaled to 100GB and populated accordingly
- The compliance of the database auxiliary data structures was verified
- The database load time was correctly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

Additional Audit Notes:

None.

Respectfully Yours,



François Raab, President



Bradley J. Askins, Auditor

## TPC Benchmark H Overview

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

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

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

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

---

## 1. General Items

---

### 1.1 Benchmark Sponsor

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

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

### 1.2 Parameter Settings

*Settings must be provided for all customer-tunable parameters and options that have been changed from the defaults found in actual products, including but not limited to:*

- *Database Tuning Options*
- *Optimizer/Query execution options*
- *Query processing tool/language configuration parameters*
- *Recovery/commit options*
- *Consistency/locking options*
- *Operating system and configuration parameters*
- *Configuration parameters and options for any other software component incorporated into the pricing structure*
- *Compiler optimization options*

Appendix A contains the Solaris and Sybase IQ parameters used in this benchmark.

---

### 1.3 Configuration Diagram

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

The priced and measure configurations are as follow:

#### **Priced Configuration**



#### **SUN Fire X4100 Server**

2 X 3.0 GHz AMD Opteron processors

16 GB Memory

4 X 73 GB internal disks

**NO EXTERNAL STORAGE REQUIRED**

---

## 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 2.3.1 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 2.3.1 was used to generate the substitution parameters.

### 3.4 Query Text and Output Data from Qualification Database

*The executable query text used for query validation must be disclosed along with the corresponding output data generated during the execution of the query text against the qualification database. If minor modifications (see Clause 2.2.3) have been applied to any functional query definitions or approved variants in order to obtain executable query text, these modifications must be disclosed and justified. The justification for a particular minor query modification can apply collectively to all queries for which it has been used. The output data for the power and throughput tests must be made available electronically upon request.*

Appendix C contains the query text and query output. The standard queries were used throughout with the following modifications:

- In Q1, Q4, Q5, Q6, Q10, Q12, Q14, Q15 and Q20, the "dateadd" function is used to perform date arithmetic.
- In Q7, Q8 and Q9, the "datepart" function is used to extract part of a date (e.g., "year").
- In Q2, Q3, Q10, Q18 and Q21, the "top" function is used to restrict the number of output rows.
- The semicolon (;) is used as a command delimiter.

### 3.5 Query Substitution Parameters and Seeds Used

*The query substitution parameters used for all performance tests must be disclosed in tabular format, along with the seeds used to generate these parameters.*

---

Appendix D contains the seed and query substitution parameters.

### **3.6 Query Isolation Level**

*The isolation level used to run the queries must be disclosed. If the isolation level does not map closely to the levels defined in Clause 3.4, additional descriptive detail must be provided.*

The queries and transactions were run with isolation level 3 (repeatable read).

### **3.7 Source Code of Refresh Functions**

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

Appendix B contains the source code for the refresh functions.



---

## 4. Clause 3 Database System Properties

---

### 4.1 ACID Properties

*The ACID (Atomicity, Consistency, Isolation and Durability) properties of transaction processing systems must be supported by the system under test during the timed portion of this benchmark. Since TPC-H is not a transaction processing benchmark, the ACID properties must be evaluated outside the timed portion of the test.*

Source code for the ACID test is included in Appendix B.

### 4.2 Atomicity

*The system under test must guarantee that transactions are atomic; the system will either perform all individual operations on the data, or will assure that no partially-completed operations leave any effects on the data.*

#### 4.2.1 Completed Transaction

*Perform the ACID Transaction for a randomly selected set of input data and verify that the appropriate rows have been changed in the ORDERS, LINEITEM, and HISTORY tables*

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

#### 4.2.2 Aborted Transaction

*Perform the ACID Transaction for a randomly selected set of input data, substituting a ROLLBACK of the transaction for the COMMIT of the transaction. Verify that the appropriate rows have not been changed in the ORDERS, LINEITEM, and HISTORY tables.*

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

### 4.3 Consistency

*Consistency is the property of the application that requires any execution of transactions to take the database from one consistent state to another.*

---

#### 4.3.1 Consistency Test

*Verify that ORDERS and LINEITEM tables are initially consistent, submit the prescribed number of ACID Transactions with randomly selected input parameters, and re-verify the consistency of the ORDERS and LINEITEM.*

1. The consistency of the ORDERS and LINEITEM tables was verified based on a sample of order keys.
2. 100 ACID Transactions were submitted by each of six execution streams.
3. The consistency of the ORDERS and LINEITEM tables was re-verified.

#### 4.4 Isolation

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

##### 4.4.1 Read-Write Conflict with Commit

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

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

##### 4.4.2 Read-Write Conflict with Rollback

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

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

##### 4.4.3 Write-Write Conflict with Commit

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

1. An ACID Transaction, T1, was started for a randomly selected O\_KEY, L\_KEY, and DELTA. T1 was suspended prior to COMMIT.
2. Another ACID Transaction, T2, was started using the same O\_KEY and L\_KEY and a randomly selected DELTA.
3. T2 waited.

- 
4. T1 was allowed to COMMIT and T2 completed.
  5. It was verified that  $T2.L\_EXTENDEDPRICE = T1.L\_EXTENDEDPRICE + (DELTA1 * (T1.L\_EXTENDEDPRICE / T1.L\_QUANTITY))$

#### 4.4.4 Write-Write Conflict with Rollback

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

1. An ACID Transaction, T1, was started for a randomly selected O\_KEY, L\_KEY, and DELTA. T1 was suspended prior to ROLLBACK.
2. Another ACID Transaction, T2, was started using the same O\_KEY and L\_KEY and a randomly selected DELTA.
3. T2 waited.
4. T1 was allowed to ROLLBACK and T2 completed.
5. It was verified that  $T2.L\_EXTENDEDPRICE = T1.L\_EXTENDEDPRICE$ .

#### 4.4.5 Concurrent Progress of Read and Write Transactions

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

1. An ACID Transaction, T1, was started for a randomly selected O\_KEY, L\_KEY, and DELTA. T1 was suspended prior to ROLLBACK.
2. Another Transaction, T2, was started which did the following:  
  
For random values of PS\_PARTKEY and PS\_SUPPKEY, all columns of the PARTSUPP table for which PS\_PARTKEY and PS\_SUPPKEY are equal, are returned.
3. T2 completed.
4. T1 was allowed to COMMIT.
5. It was verified that appropriate rows in ORDERS, LINEITEM and HISTORY tables were changed.

#### 4.4.6 Read-Only Query Conflict with Update Transaction

*Demonstrate that the continuous submission of arbitrary (read-only) queries against one or more tables of the database does not indefinitely delay update transactions affecting those tables from making progress.*

1. A Transaction, T1, executing Q1 against the qualification database, was started using a randomly selected DELTA.
2. An ACID Transaction T2, was started for a randomly selected O\_KEY, L\_KEY and DELTA.
3. T2 completed and appropriate rows in the ORDERS, LINEITEM and HISTORY tables had been changed.
4. Transaction T1 completed executing Q1.

---

## 4.5 Durability

*The SUT must guarantee durability: the ability to preserve the effects of committed transactions and insure database consistency after recovery from any one of the failures listed in Clause 3.5.3.*

### 4.5.1 Failure of a Durable Medium

*Guarantee the database and committed updates are preserved across a permanent irrecoverable failure of any single durable medium containing TPC-H database tables or recovery log tables.*

All disks containing TPC-H tables, TPC-H indexes, the Sybase IQ utility db file and the Sybase IQ log file are housed on RAID1 volumes. A permanent irrecoverable failure of one of these disks was simulated by removing it from the X4100 Server. Shortly after this was done, the SybaseIQ server died because a TEMP db segment housed on the removed disk was no longer accessible to IQ. After an interval of about 60 seconds, power to the X4100 Server was cut off. The outcome is described in section 4.5.2.

### 4.5.2 System Crash

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

A system crash was produced by cutting off power to the X4100 Server. When power was restored, the system automatically rebooted and the database was restarted. The durability success file and the HISTORY table were compared successfully.

### 4.5.3 Memory Failure

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

A memory failure was simulated by cutting off power to the X4100 Server. The outcome is described in section 4.5.2.

---

## 5. Clause 4 Scaling and Database Population

---

### 5.1 Ending Cardinality of Tables

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

<i>Table</i>	<i>Rows</i>
<i>Lineitem</i>	600037902
<i>orders</i>	150000000
<i>Partsupp</i>	80000000
<i>Part</i>	20000000
<i>Customer</i>	15000000
<i>Supplier</i>	1000000
<i>Nation</i>	25
<i>Region</i>	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 2 RAID 1 volumes. Each volume was constructed from two raw partitions using the Solaris Volume Manager.
- The Temp database for Sybase IQ was configured using four raw partitions, one on each of four internal disks. The Temp databases were not mirrored.

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

<b>Partition</b>	<b>Use</b>	<b>Size (in GB)</b>
c1t0d0s0	root partition	8
c1t1d0s6	swap	1
c1t2d0s6	swap	1
c1t3d0s6	swap	1
/export/home/swap/swap1	swap	3
c1t0d0s1	tempdb [via symbolic link /sybase2/T01]	2
c1t1d0s4	tempdb [via symbolic link /sybase2/T02]	9
c1t2d0s4	tempdb [via symbolic link /sybase2/T03]	9
c1t3d0s4	tempdb [via symbolic link /sybase2/T04]	9
c1t1d0s7	SVM state replica	0.3
c1t2d0s7	SVM state replica	0.3
C1t3d0s7	SVM state replica	0.3

---

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

**SVM Device Details:**

Raw Partition Name	SVM Device Name	SVM Contained Device	Symbolic Link	Database Usage	Database Device Size GB	RAID
c1t1d0s5	d1	d2	none	/sybase2	1	RAID 1
c1t2d0s5	d1	d3	none	/sybase2	1	RAID 1
c1t0d0s7	d15	d10	/sybase2/M01	IQ Main	58.33	RAID 1
c1t1d0s3	d15	d11	/sybase2/M01	IQ Main	57	RAID 1
c1t2d0s3	d25	d20	/sybase2/M02	IQ Main	57	RAID 1
c1t3d0s3	d25	d21	/sybase2/M02	IQ Main	57	RAID 1

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

### 5.3 Database partition/replication mapping

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

Database partitioning/replication was not used.

---

## 5.4 RAID Feature

*Implementations may use some form of RAID to ensure high availability. If used for data, auxiliary storage (e.g. indexes) or temporary space, the level of RAID must be disclosed for each device.*

RAID 1 was used for all base tables and auxiliary data structures. In addition, the Sybase IQ utility db file and log file also resided on a RAID 1 devices.

## 5.5 Modifications to the DBGEN

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

The supplied DBGEN version 2.3.1 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 =01:38:53

## 5.7 Data Storage Ratio

*The data storage ratio must be disclosed. It is computed as the ratio between the total amount of priced disk space, and the chosen test database size as defined in Clause 4.1.3.*

The data storage ratio is computed from the following information:

Disk Type	# of Disks	GB* per disk	Total Disk Space in GB**
Internal	4	73	271.95
		<b>Total Space</b>	271.95
		<b>Data Storage Ratio</b>	2.72

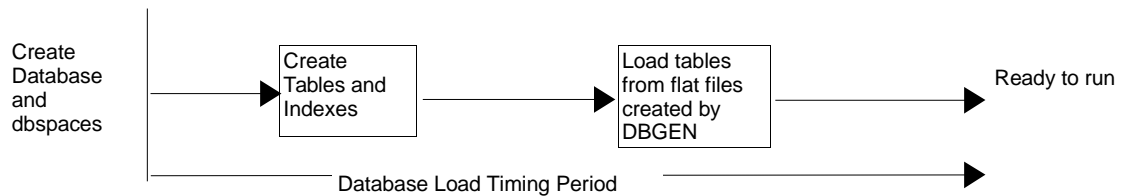
\* Disk manufacturer definition of one GB is  $10^9$  bytes

\*\*In this calculation one GB is defined as  $2^{30}$  bytes

---

## 5.8 Database Load Mechanism Details and Illustration

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



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

## 5.9 Qualification Database Configuration

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

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



---

## 6. Clause 5 Performance Metrics and Execution Rules

---

### 6.1 System Activity Between Load and Performance Tests

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

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

All scripts and queries used are included in Appendix F

### 6.2 Steps in the Power Test

*The details of the steps followed to implement the power test (e.g., system boot, database restart, etc.) must be disclosed.*

The following steps were used to implement the power test:

1. RF1 Refresh Transaction
2. Stream 00 Execution
3. RF2 Refresh Transaction

### 6.3 Timing Intervals for Each Query and Refresh Functions

*The timing intervals for each query and for both refresh functions must be reported for the power test.*

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

### 6.4 Number of Streams for the Throughput Test

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

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	2006-05-31	11:31:22	2006-05-31	11:44:26
RF2	2006-05-31	11:44:26	2006-05-31	11:56:05
RF1	2006-05-31	12:01:14	2006-05-31	12:16:24
RF2	2006-05-31	12:21:25	2006-05-31	12:33:49
RF1	2006-05-31	12:33:50	2006-05-31	12:42:20
RF2	2006-05-31	12:47:20	2006-05-31	12:57:07
RF1	2006-05-31	13:07:09	2006-05-31	13:15:18
RF2	2006-05-31	13:20:19	2006-05-31	13:30:11
RF1	2006-05-31	13:45:12	2006-05-31	13:57:28

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

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

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

## 6.9 Performance Metrics

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

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

## 6.10 The Performance Metric and Numerical Quantities from Both Runs

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

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

---

Run ID	QppH@1000GB	QthH@1000GB	QphH@1000GB
Run 1	4412.9	3947.4	4173.7
Run 2	4408.6	3872.9	4132.1
% Difference	-0.10%	-1.92%	-1.01%

## 6.11 System Activity Between Performance Tests

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

No activity occurred between Run 1 and Run 2. Moreover Sybase IQ was not restarted after the database load or between the two runs.

---

## 7. Clause 6 SUT and Driver Implementation

---

### 7.1 Driver

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

The entire test is run by executing the ntest shellscrip.

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

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

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

The Throughput Test is performed when ntest concurrently executes the 5 query stream scripts, stream[1-5].sql, together with the script update\_throughput5.sql. The latter executes the refresh functions in parallel with the query streams according to the rules set out in the TPC-H specification document.

### 7.2 Implementation-Specific Layer

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

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

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

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

### 7.3 Profile-Directed Optimization

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

Profile-directed optimization was not used.

---

## 8. Clause 7 Pricing

---

### 8.1 Hardware and Software Used

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

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

### 8.2 Total Three Year Price

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

The total 3-year price of the configuration is \$19057.05. 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 generally available as of Jun 23, 2006.

---

## **9. Auditor's Information and Attestation Letter**

---

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

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

## Appendix A. Solaris 10 and Sybase IQ 12.6 Parameters

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

### Sybase IQ Server Configuration Parameters

#### utility.cfg

```
#
# these are the default suggested startup parameters
for the asiq
#
# note that you will still need to provide the
# server with the server name on startup "-n", and
# possibly a different system port number
#
-n tpchsfsgal
-c 22m
-gd all
-gl all
-gm 15
-gp 4096
-ti 4400
-tl 300
-iqtc 128
-iqmc 128
```

#### tpch.cfg

```
# note the port number here must matche the one in
# .odbc.ini
-c 12m
-gd all
-gl all
-gm 30
-gp 4096
-ti 4400
-tl 300
-iqmc 10200
-iqtc 4400
-iqmt 800
-iggovern 12
-iqpartition 2
-x tcpip{port=3002}
-n tpch_sfsgal
```

### Sybase IQ Database Options

(altered from default)

#### options.sql

```
SET OPTION PUBLIC.Allow_Nulls_By_Default='Off';
SET OPTION PUBLIC.Flatten_Subqueries = 'On';
SET OPTION PUBLIC.Force_No_Scroll_Cursors='On';
SET OPTION PUBLIC.Load_Memory_Mb=0;
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.Query_Plan='off';
SET OPTION PUBLIC.Hash_Thrashing_Percent=100;
SET OPTION PUBLIC.Main_Reserved_DBSpace_MB=300;
SET OPTION
PUBLIC.SignificantDigitsForDoubleEquality=15;
```

```
SET OPTION PUBLIC.Sort_Phase1_Helpers=3;
```

```
SET OPTION PUBLIC.Max_IQ_Threads_Per_Connection=100;
SET OPTION PUBLIC.Append_Load='On';
SET OPTION PUBLIC.Default_Having_Selectivity = 1;
SET OPTION PUBLIC.Default_Like_Range_Selectivity = 1;
SET OPTION PUBLIC.Garray_Fill_Factor_Percent=3;
```

```
SET OPTION PUBLIC.Max_Hash_Rows = 10000000;
SET OPTION PUBLIC.Prefetch_Threads_Percent=18;
SET OPTION PUBLIC.Sweeper_Threads_Percent=12 ;
SET OPTION PUBLIC.Wash_Area_Buffers_Percent = '18';
```

### Sybase IQ Environment Variables

```
SYBASE="/export/home/sybase"
export SYBASE
SYBASE_OCS="OCS-12_5"
export SYBASE_OCS
ASDIR="${SYBASE}/ASIQ-12_6"
export ASDIR
PATH="${ASDIR}/bin:${SYBASE}/${SYBASE_OCS}/bin:${PATH}:/etc:."
export PATH
IQLIB="${ASDIR}/usr/lib:${ASDIR}/lib:${SYBASE}/${SYBASE_OCS}/lib"
LD_LIBRARY_PATH_64="${IQLIB}:${LD_LIBRARY_PATH_64}"
export LD_LIBRARY_PATH_64
LD_LIBRARY_PATH="${IQLIB}:${LD_LIBRARY_PATH}"
export LD_LIBRARY_PATH
unset IQLIB
```

#### .odbc.ini

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

[tpch]
Driver=/export/home/sybase/asiq12/lib/dbodbc7_r.so.1
EngineName=tpch_sfsgal
CommLinks=tcpip{host=10.8.4.38;Port=3002}
DatabaseName=tpch
UserID=DBA
Password=SQL
DBG=yes
LOG=/tmp/tpch_odbc.log

[utility_db]
Driver=/export/home/sybase/asiq12/lib/dbodbc7_r.so.1
EngineName=tpchsfsgal
CommLinks=tcpip{host=10.8.4.38 ;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

### check\_query1.bash

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

### create\_database.sql

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

### create\_dbspaces.sql

```
create dbspace main2 as '/sybase2/M02' iq store;

create dbspace iqtemp2 as '/sybase2/T02' iq
temporary store;
create dbspace iqtemp3 as '/sybase2/T03' iq
temporary store;
create dbspace iqtemp4 as '/sybase2/T04' iq
temporary store;
```

### create\_tables.sql

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

```
CREATE TABLE nation
(
n_nationkey unsigned int,
```

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

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

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

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

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

```
CREATE TABLE orders
(
o_orderkey unsigned int,
o_custkey unsigned int,
o_orderstatus char(1),
o_totalprice double precision,
o_orderdate date,
o_orderpriority char(15),
o_clerk char(15),
```

```

o_shippriority      int,
o_comment           varchar(79),
PRIMARY KEY (o_orderkey)
);
CREATE HG INDEX o_custkey_hg ON orders(o_custkey) ;
CREATE DATE INDEX o_orderdate_date ON orders
(o_orderdate) ;

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

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

```

## tpch\_rf.sql

```

=====
create table refresh_control ( rf1_data_set int not
null, rf2_data_set int not null);
insert into refresh_control values (0,0);
commit;
CREATE PROCEDURE DBA.tpch_rf1 (IN c_directory varchar
(128),
                               IN c_stream varchar(3))
ON EXCEPTION RESUME
BEGIN
  DECLARE delim_ascii integer;
  DECLARE c_data_set varchar(3);
  DECLARE i_data_set integer;
  DECLARE c_cmd long varchar;
  DECLARE outfile_name varchar(128); -- Debug
  DECLARE outfile_name2 varchar(128); -- Debug
  DECLARE c_lf varchar(2);
  DECLARE t_qstart timestamp;
  DECLARE t_qstop timestamp;
  DECLARE n_seconds numeric(12,5);
  DECLARE c_sqlstate CHAR(5);
  SET t_qstart = now(*);
  SET c_lf=char(10);
  SELECT rf1_data_set INTO i_data_set FROM
refresh_control;
  SET c_data_set=CAST(i_data_set+1 AS varchar(3));
  SET c_cmd='load table orders ( '+c_lf;
  SET c_cmd=c_cmd+' o_orderkey '+char(39)+'|'+char(39)
+', '+c_lf;
  SET c_cmd=c_cmd+' o_custkey '+char(39)+'|'+char(39)
+', '+c_lf;
  SET c_cmd=c_cmd+' o_orderstatus '+char(39)+'|'+char

```

```

(39)+'|'+c_lf;
  SET c_cmd=c_cmd+' o_totalprice '+char(39)+'|'+char
(39)+'|'+c_lf;
  SET c_cmd=c_cmd+' o_orderdate date('+char(39)+'YYYY-
MM-DD'+char(39)+')', filler(1), '+c_lf;
  SET c_cmd=c_cmd+' o_orderpriority '+char(39)+'
|'+char(39)+'|'+c_lf;
  SET c_cmd=c_cmd+' o_clerk '+char(39)+'|'+char(39)+'|'+
c_lf;
  SET c_cmd=c_cmd+' o_shippriority '+char(39)+'|'+char
(39)+'|'+c_lf;
  SET c_cmd=c_cmd+' o_comment '+char(39)+'|'+char(39)
+'|'+c_lf;
  SET c_cmd=c_cmd+'from '+char(39)
+c_directory+'orders.tbl.u'+c_data_set+char(39)+c_lf;
  SET c_cmd=c_cmd+'row delimited by '+char(39)
+'\\x0a'+char(39)+' quotes off escapes off preview
on;';
EXECUTE IMMEDIATE c_cmd;
SELECT SQLSTATE INTO c_sqlstate;
IF c_sqlstate != '00000' THEN
  ROLLBACK;
  RAISERROR 23002 'RF1 failed at Step 1 with
SQLSTATE: ', c_sqlstate;
  RETURN(1);
END IF;
SET c_cmd='load table lineitem ( '+c_lf;
SET c_cmd=c_cmd+' l_orderkey '+char(39)+'|'+char(39)
+', '+c_lf;
SET c_cmd=c_cmd+' l_partkey '+char(39)+'|'+char(39)
+', '+c_lf;
SET c_cmd=c_cmd+' l_suppkey '+char(39)+'|'+char(39)
+', '+c_lf;
SET c_cmd=c_cmd+' l_linenum '+char(39)+'|'+char
(39)+'|'+c_lf;
SET c_cmd=c_cmd+' l_quantity '+char(39)+'|'+char(39)
+', '+c_lf;
SET c_cmd=c_cmd+' l_extendedprice '+char(39)+'
|'+char(39)+'|'+c_lf;
SET c_cmd=c_cmd+' l_discount '+char(39)+'|'+char(39)
+', '+c_lf;
SET c_cmd=c_cmd+' l_tax '+char(39)+'|'+char(39)+'|'+
c_lf;
SET c_cmd=c_cmd+' l_returnflag '+char(39)+'|'+char
(39)+'|'+c_lf;
SET c_cmd=c_cmd+' l_linestatus '+char(39)+'|'+char
(39)+'|'+c_lf;
SET c_cmd=c_cmd+' l_shipdate date('+char(39)+'YYYY-
MM-DD'+char(39)+')', filler(1), '+c_lf;
SET c_cmd=c_cmd+' l_commitdate date('+char(39)
+'YYYY-MM-DD'+char(39)+')', filler(1), '+c_lf;
SET c_cmd=c_cmd+' l_receiptdate date('+char(39)
+'YYYY-MM-DD'+char(39)+')', filler(1), '+c_lf;
SET c_cmd=c_cmd+' l_shipinstruct '+char(39)+'|'+char
(39)+'|'+c_lf;
SET c_cmd=c_cmd+' l_shipmode '+char(39)+'|'+char(39)
+', '+c_lf;
SET c_cmd=c_cmd+' l_comment '+char(39)+'|'+char(39)
+'|'+c_lf;
SET c_cmd=c_cmd+'from '+char(39)
+c_directory+'lineitem.tbl.u'+c_data_set+char(39)
+c_lf;
SET c_cmd=c_cmd+'row delimited by '+char(39)
+'\\x0a'+char(39)+c_lf+'quotes off escapes off preview
on;';
EXECUTE IMMEDIATE c_cmd;
SELECT SQLSTATE INTO c_sqlstate;
IF c_sqlstate != '00000' THEN
  rollback;
  RAISERROR 23002 'RF1 failed at Step 2 with
SQLSTATE: ', c_sqlstate;
  RETURN(1);
END IF;
UPDATE refresh_control SET rf1_data_set=cast
(c_data_set AS integer);
COMMIT;
SET t_qstop = now(*);
SET n_seconds=cast(datediff

```

```

(millisecond,t_qstart,t_qstop) AS numeric(12,5))/1000;
  SET c_cmd='Stream updates Update
update_'+c_stream+'_RF1 LENGTH -- '+cast(n_seconds AS
varchar(20))+ ' seconds' ;
  SELECT c_cmd;
  RETURN(0);
END;
CREATE PROCEDURE DBA.tpch_rf2 (in c_directory varchar
(128),
                                in c_stream varchar(3))
ON exception resume
BEGIN
  DECLARE delim_ascii integer;
  DECLARE c_data_set varchar(3);
  DECLARE i_data_set integer;
  DECLARE c_cmd long varchar;
  DECLARE outfile_name varchar(128); -- Debug
  DECLARE c_lf varchar(2);
  DECLARE t_qstart timestamp;
  DECLARE t_qstop timestamp;
  DECLARE n_seconds numeric(12,5);
  DECLARE c_sqlstate CHAR(5);
  SET t_qstart = now(*);
  SET c_lf=char(10);
  SELECT rf2_data_set INTO i_data_set FROM
refresh_control;
  SET c_data_set=CAST(i_data_set+1 AS varchar(3));
  CREATE TABLE #delete_table ( d_orderkey UNSIGNED
INT, PRIMARY KEY (d_orderkey) );
  SET c_cmd='load table #delete_table (d_orderkey
'+char(39)+'\x0a'+char(39)+' ' +c_lf;
  SET c_cmd=c_cmd+'from '+char(39)
+c_directory+'delete.'+c_data_set+char(39)+c_lf;
  SET c_cmd=c_cmd+'quotes off '+c_lf;
  SET c_cmd=c_cmd+'escapes off; '+c_lf;
  EXECUTE IMMEDIATE c_cmd;
  SELECT SQLSTATE INTO c_sqlstate;
  IF c_sqlstate != '00000' THEN
    ROLLBACK;
    SET c_cmd='RF2 failed at Step 1 with SQLSTATE:
'+c_sqlstate;
    RAISERROR 23002 c_cmd;
    RETURN(1);
  END IF;
  DELETE lineitem FROM lineitem, #delete_table WHERE
l_orderkey = d_orderkey;
  SELECT SQLSTATE INTO c_sqlstate;
  IF c_sqlstate != '00000' THEN
    ROLLBACK;
    SET c_cmd='RF2 failed at Step 2 with SQLSTATE:
'+c_sqlstate;
    RAISERROR 23002 c_cmd;
    RETURN(1);
  END IF;
  DELETE orders FROM orders, #delete_table WHERE
o_orderkey = d_orderkey;
  SELECT SQLSTATE INTO c_sqlstate;
  IF c_sqlstate != '00000' THEN
    ROLLBACK;
    SET c_cmd='RF2 failed at Step 3 with SQLSTATE:
'+c_sqlstate;
    RAISERROR 23002 c_cmd;
    RETURN(1);
  END IF;
  UPDATE refresh_control SET rf2_data_set=CAST
(c_data_set AS integer);
  COMMIT;
  DROP TABLE #delete_table;
  SET t_qstop = now(*);
  SET n_seconds=cast(datediff
(millisecond,t_qstart,t_qstop) as numeric(12,5))/1000;
  SET c_cmd='Stream updates Update
update_'+c_stream+'_RF2 LENGTH -- '+cast(n_seconds as
varchar(20))+ ' seconds' ;
  SELECT c_cmd;
  RETURN(0);
END;

```

## load\_region.sql

```

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

```

## load\_nation.sql

```

=====
LOAD TABLE NATION (
N_NATIONKEY          | |
N_NAME                | |
N_REGIONKEY          | |
N_COMMENT            | |
)
FROM '/sybase_stage/nation.tbl'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
=====

```

## load\_customer.sql

```

=====
LOAD TABLE CUSTOMER (
C_CUSTKEY            | |
C_NAME                | |
C_ADDRESS            | |
C_NATIONKEY          | |
C_PHONE              | |
C_ACCTBAL            | |
C_MKTSEGMENT         | |
C_COMMENT            | |
)
FROM '/sybase_stage/customer.tbl'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
=====

```

## load\_part.sql

```

=====
LOAD TABLE PART (
P_PARTKEY            | |
P_NAME                | |
P_MFGR               | |
P_BRAND              | |
P_TYPE               | |
P_SIZE               | |
P_CONTAINER          | |
P_RETAILPRICE        | |
P_COMMENT            | |
)
FROM '/sybase_stage/part.tbl'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
=====

```

## load\_supplier.sql

```
=====
LOAD TABLE SUPPLIER (
S_SUPPKEY          | | ,
S_NAME            | | ,
S_ADDRESS         | | ,
S_NATIONKEY       | | ,
S_PHONE           | | ,
S_ACCTBAL         | | ,
S_COMMENT         | | ,
)
FROM '/sybase_stage/supplier.tbl'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
=====
```

## load\_partsupp.sql

```
=====
LOAD TABLE PARTSUPP (
PS_PARTKEY        | | ,
PS_SUPPKEY        | | ,
PS_AVAILQTY       | | ,
PS_SUPPLYCOST     | | ,
PS_COMMENT        | | ,
)
FROM '/sybase_stage/partsupp.tbl'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
=====
```

## load\_orders.sql

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

## load\_lineitem.sql

```
=====
LOAD TABLE LINEITEM (
L_ORDERKEY        | | ,
```

```
L_PARTKEY          | | ,
L_SUPPKEY          | | ,
L_LINENUMBER      | | ,
L_QUANTITY        | | ,
L_EXTENDEDPRICE   | | ,
L_DISCOUNT       | | ,
L_TAX             | | ,
L_RETURNFLAG      | | ,
L_LINestatus      | | ,
L_SHIPDATE        | | ,
L_COMMITDATE      | | ,
L_RECEIPTDATE     | | ,
L_SHIPINSTRUCT    | | ,
L_SHIPMODE        | | ,
L_COMMENT         | | ,
)
FROM
```

```
'/sybase_stage/lineitem.tbl.1',
'/sybase_stage/lineitem.tbl.2',
'/sybase_stage/lineitem.tbl.3',
'/sybase_stage/lineitem.tbl.4',
'/sybase_stage/lineitem.tbl.5',
'/sybase_stage/lineitem.tbl.6',
'/sybase_stage/lineitem.tbl.7',
'/sybase_stage/lineitem.tbl.8'
```

```
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
checkpoint;
commit;
=====
```

## update\_power.sql

```
=====
create variable qstart timestamp;
create variable qstop timestamp;
create variable c_sqlstate CHAR(5);
create variable c_path varchar(128);
set c_path='/sybase_stage/';
set qstart=now(*);
select 'Stream 0 RF1 START -- ', qstart ;
call tpch_rf1 (c_path,'0');
set qstop=now(*);
select 'Stream 0 Update RF1 LENGTH -- ',cast(datediff
(millisecond,qstart,qstop) as numeric)/1000, '
seconds';
select 'Stream 0 RF1 FINISH -- ', qstop ;
-- Sleep Until the query stream completes
set qstart = now(*);
select 'Stream 0 RF WAITING -- ', qstart;
xp_cmdshell
('/export/home/sybase/run/scripts/check_query1.bash');
set qstart = now(*);
select 'Stream 0 RF CONTINUING -- ', qstart;
set qstart = now(*);
select 'Stream 0 RF2 START -- ', qstart ;
call tpch_rf2 (c_path,'0');
set qstop=now(*);
select 'Stream 0 Update RF2 LENGTH -- ',cast(datediff
(millisecond,qstart,qstop) as numeric)/1000, '
seconds';
select 'Stream 0 RF2 FINISH -- ', qstop ;
=====
```

## update\_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();
call tpch_rf1 (c_path,'1');
commit;
tpch_wait;
call tpch_rf2 (c_path,'1');
commit;
tpch_wait;
call tpch_rf1 (c_path,'2');
commit;
tpch_wait;
call tpch_rf2 (c_path,'2');
commit;
tpch_wait;
call tpch_rf1 (c_path,'3');
commit;
tpch_wait;
call tpch_rf2 (c_path,'3');
commit;
tpch_wait;
call tpch_rf1 (c_path,'4');
commit;
tpch_wait;
call tpch_rf2 (c_path,'4');
commit;
tpch_wait;
call tpch_rf1 (c_path,'5');
commit;
tpch_wait;
call tpch_rf2 (c_path,'5');
commit;
tpch_wait;

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

```

## gen\_streams\_new.ksh

```

#!/bin/ksh

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

PATH=/export/home/sybase/ASIQ-
12_5/bin:/export/home/sybase/OCS-
12_5/bin:/usr/openwin/bin:/bin:./usr/dist/pkgs/forte_
dev/SUNWspro/bin:/usr/ccs/bin:/usr/dt/bin:/usr/dist/pk
gs/devpro,v4.0/5.x-
sparc/bin:/usr/dist/local/exe:/usr/dist/exe:/usr/ucb:/
usr/sbin:/net/josie/export/home18/rgostan/bin:/export/
home/sybase/run/scripts:/etc:./export/home/sybase/run
/tpch/appendix/dbgen
export PATH
export DSS_PATH=/export/home/sybase/run/scripts;
export
DSS_CONFIG=/export/home/sybase/run/tpch/appendix/dbgen
;
export DSS_DIST=dists.dss;
export
DSS_QUERY=/export/home/sybase/run/tpch/appendix/templa
tes/queries;
#export
DSS_QUERY=/export/home/sybase/run/tpch/appendix/templa
tes/queries.debug;

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

i=0

```

```

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

echo $seed
=====
ACID Test Execution Code
=====

atomicity test
=====

dbtest acid_atomic_main.tst > acid_atomic_main.out

=====

consistency test
=====

dbtest acid_consistency_main.tst >
acid_consistency_main.out

```

## isolation tests

```

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

```

## durability test

```

dbtest acid_durability_main.tst >
acid_durability_main.out

```

## ACID Test Source Code

### acid\_atomic\_main.tst

```

stringconnect "dsn=tpch;"

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

include 'acid_functions.tst'
commit

%
% Atomicity test with rollback

```

```

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

run test 'acid_atomic_setup.tst'

stringconnect "dsn=tpch;"
let counter=0

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

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

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

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

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

commit

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

stringconnect "dsn=tpch;"

open curl {select ordr, line, delta from aa_whattodo}
LOOP {
fetch curl into ordr, line, delta
if ROWSTATUS != FOUND then { BREAK LOOP } endif
print 'Acid transaction for: o_key-',ordr,' l_key-',
line,' delta-',delta
execute {select o_totalprice, l_quantity,
l_extendedprice

```

```

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

execute {call acid_transaction(^, ^, ^, rprice,
quantity,
    tax, disc, extprice,
ototal)
    } substitute ordr, line, delta
commit
execute {select now(*)} into times
print 'commit : ', times

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

close curl
commit

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

End Test

=====
acid_atomic_setup.tst
=====

stringconnect "dsn=tpch;"

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

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

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

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

print 'Number of rows before load: ',ROWS

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

```

```

linenumber, delta
  print counter, ' ',orderkey, ' ',linenumber,' ',
delta
}
ENDLOOP

commit

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

print 'Number of rows after load: ',ROWS

disconnect

End Test

```

### acid\_consistency\_main.tst

```

=====
stringconnect "dsn=tpch;"

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

include 'acid_functions.tst'

run test 'acid_consistency_setup.tst'

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

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

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

% let the log flush...
sleep 150000

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

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

```

End Test

### acid\_consistency\_query.tst

```

=====
stringconnect "dsn=tpch;"

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

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

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

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

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

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

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

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

disconnect

END Test

```

### acid\_consistency\_setup.tst

```

=====
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,
ord int null,
line int null,
delta int null)

on SYSTEM
}

execute {checkpoint}

```

```

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 <= 9}; {let j = j + 1})
{
    execute { insert into latest(stream,last)
values (^,0) }
    substitute j
} endloop
commit

```

```
print 'latest created'
```

```

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

```

```
commit
```

```

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

```

```
End Test
```

### acid\_consistency\_txn.tst

```
stringconnect "dsn=tpch;"
```

```
execute {set temporary option chained='on'}
```

```

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

```

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

```

```

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=1200*(i-smallest)
if i+4>=biggest
then {let num=num+8000}
endif
sleep 14000
}
ENDLOOP

```

```
disconnect
```

```
End Test
```

### acid\_durability\_main.tst

```
stringconnect "dsn=tpch;"
```

```

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

```

```

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

```

```

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

```

```

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

```

```

start test '-o' 'acid_dura_user1.ot' 'stream=1'
'acid_durability_txn.tst'

```

```

sleep 1000
start test '-o' 'acid_dura_user2.ot' 'stream=2'
'acid_durability_txn.tst'

```

```

sleep 1000
start test '-o' 'acid_dura_user3.ot' 'stream=3'
'acid_durability_txn.tst'

```

```

sleep 1000
start test '-o' 'acid_dura_user4.ot' 'stream=4'
'acid_durability_txn.tst'

```

```

sleep 1000
start test '-o' 'acid_dura_user5.ot' 'stream=5'
'acid_durability_txn.tst'

```

```

sleep 1000
start test '-o' 'acid_dura_user6.ot' 'stream=6'
'acid_durability_txn.tst'

```

```

sleep 1000
start test '-o' 'acid_dura_user7.ot' 'stream=7'
'acid_durability_txn.tst'

```

```

sleep 1000
start test '-o' 'acid_dura_user8.ot' 'stream=8'
'acid_durability_txn.tst'

```

```

sleep 1000
start test '-o' 'acid_dura_user9.ot' 'stream=9'
'acid_durability_txn.tst'

```

```
synchronize 10
```

```

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

```

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



```
'acid_durability_query.tst'

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

End Test

=====
acid_durability_query.tst
=====

Test 'tpcd_acid_query'
Description 'perform the acid query.'

stringconnect "dsn=tpch;"

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

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

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

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

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

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

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

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

disconnect

END Test
```

### acid\_durability\_setup.tst

```
=====
stringconnect "dsn=tpch;"

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

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

=====
```

```
execute {checkpoint}

print 'acid_table CREATED!!'

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

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

commit
execute {checkpoint}

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

print 'Number of rows after load: ',ROWS

End Test
```

### acid\_durability\_txn.tst

```
=====
stringconnect "dsn=tpch;"

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

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

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

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

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

  assert SQLCODE=0 then { DIE } endassert
  commit

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

=====
```

```

sleep 1000
}
ENDLOOP

synchronize 10

End Test

=====
acid_functions.tst
=====

print 'creating the sleep procedure'

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

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

print 'creating the Acid Transaction'

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

execute{ CREATE PROCEDURE acid_transaction(
IN o_key INT,
IN l_key INT,
IN delta INT,
OUT rprice Numeric(18,8),
OUT quantity INT,
OUT tax Numeric(18,8),
OUT disc Numeric(18,8),
OUT extprice Numeric(18,8),
OUT ototal Numeric(18,8)
)
ON EXCEPTION RESUME
BEGIN
DECLARE pkey INT ;
DECLARE skey INT ;
DECLARE cost NUMERIC(18,8) ;
DECLARE new_extprice NUMERIC(18,8) ;
DECLARE new_ototal NUMERIC(18,8) ;
DECLARE new_quantity INT ;
DECLARE c_sqlstate char(5);
LOOP1: LOOP
rollback;
call dbo.sleep(1);
COMMIT;
SELECT o_totalprice
INTO ototal
FROM orders
WHERE o_orderkey = o_key ;
SELECT l_quantity,
l_extendedprice,
l_partkey,
l_suppkey,
l_tax,
l_discount
INTO quantity,
extprice,
pkey,
skey,
tax,
disc
FROM lineitem
WHERE l_orderkey = o_key

```

```

AND l_linenumber = l_key;
-- CLEAN UP IMPRECICE NUMBERS
SET ototal = ototal -
"TRUNCATE"("truncate"(extprice*(1-disc),2)*(1+tax),2);
SET rprice = "TRUNCATE"((extprice / quantity),2);
SET cost = "TRUNCATE"((rprice * delta),2);
SET new_extprice = extprice + cost;
SET new_ototal = "TRUNCATE"(new_extprice * (1.0 -
disc),2);
SET new_ototal = "TRUNCATE"(new_ototal * (1.0 +
tax),2);
SET new_ototal = ototal + new_ototal ;
SET new_quantity = quantity + delta ;
--
-- Update LineItem
--
UPDATE lineitem
SET l_quantity = new_quantity,
l_extendedprice = new_extprice
WHERE l_orderkey=o_key
AND l_linenumber=l_key;
SELECT SQLSTATE INTO c_sqlstate;
IF c_sqlstate = '00000' THEN
--
-- Update Orders
--
UPDATE orders
SET o_totalprice = new_ototal
WHERE o_orderkey = o_key;
SELECT SQLSTATE INTO c_sqlstate;
IF c_sqlstate = '00000' THEN
INSERT INTO history VALUES ( pkey, skey,
o_key, l_key, delta, now(*) ) ;
SELECT SQLSTATE INTO c_sqlstate;
IF c_sqlstate = '00000' THEN
LEAVE LOOP1;
END IF;
END IF;
END IF;
END LOOP LOOP1;
RETURN(0);
END;
}

print 'Acid transaction created'
print ' '

print 'Creating Acid query'

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

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

print 'Acid query created'
print ' '

print 'Creating Generate_acid_values function'

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

```

```

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

BEGIN

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

declare times cursor for select datediff
(millisecond,convert(char(10),getdate(), 116),now(*));
declare random1 cursor for select rand(seed);
declare random cursor for select rand();
declare get_order cursor for
select o_orderkey from orders where o_orderkey
= rand_int;
declare get_linenum 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_linenum;
fetch next get_linenum into linenumber;
close get_linenum;

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

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

print 'Creating Generate_Ps_Values function'

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

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

BEGIN

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

```

```

declare counter int;

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

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

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

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

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

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

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

set partkey = out_key;
close random;
close get_part;

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

print 'Creating history table'

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

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

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

```

---

### acid\_isolation\_main1.tst

```
=====
stringconnect "dsn=tpch;"

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

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

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

End Test
=====
```

### acid\_isolation\_main2.tst

```
=====
stringconnect "dsn=tpch;"

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

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

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

End Test
=====
```

### acid\_isolation\_main3.tst

```
=====
stringconnect "dsn=tpch;"

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

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

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

End Test
=====
```

### acid\_isolation\_main4.tst

```
=====
stringconnect "dsn=tpch;"

execute {select now(*)} into times
print ' '
print ' '
=====
```

```
print 'Isolation test 4'
print 'start = ', times
print ' '
print 'Isolation test start = ', times
```

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

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

End Test

### acid\_isolation\_main5.tst

```
=====
stringconnect "dsn=tpch;"

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

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

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

End Test

### acid\_isolation\_main6.tst

```
=====
stringconnect "dsn=tpch;"

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

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

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

End Test

### acid\_isolation\_setup.tst

```
=====
stringconnect "dsn=tpch;"

% Drop Table if found

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

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

execute {checkpoint}
=====
```

```

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

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

print 'Number of rows before load: ',ROWS

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

commit

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

print 'Number of rows after load: ',ROWS

disconnect

End Test

```

### acid\_isolation\_test1.tst

```

=====
stringconnect "dsn=tpch;"

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

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

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

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

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

End Test

```

### acid\_isolation\_test1\_query.tst

```

=====
stringconnect "dsn=tpch;"

synchronize 2
print ' '

```

```

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

execute {select ordr from acid_isolation_table}
into ordr

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

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

disconnect

END Test

```

### acid\_isolation\_test2.tst

```

=====
stringconnect "dsn=tpch;"

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

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

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

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

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

End Test

```

### acid\_isolation\_test2\_query.tst

```

=====
stringconnect "dsn=tpch;"

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

execute {select ordr from acid_isolation_table}
into ordr

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

```

```

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

disconnect

END Test

=====
acid_isolation_test3_transaction1.tst
=====

stringconnect "dsn=tpch;"

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

execute {select ord, line, delta from
acid_isolation_table}
into ord, 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 ord, 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 ' txnl_extendedprice + (delta1 *
(txnl_extendedprice/txnl_quantity))'
print 'User 1 -- result1= ', result2
print ' '

disconnect

```

End Test

```

=====
acid_isolation_test3_transaction2.tst
=====

```

```

stringconnect "dsn=tpch;"

execute {select ord, line, delta from
acid_isolation_table}
into ord, line, delta
% generate a new set of values; we only use delta2
execute { call generate_acid_values()} into ord2,
line2, delta2

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

SYNCHRONIZE 2
sleep 5000

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

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

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

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

print ' '

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

End Test

```

```

=====
acid_isolation_test4_transaction1.tst
=====

```

```

stringconnect "dsn=tpch;"

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

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

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

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

```

```

times

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

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

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

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

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

disconnect
End Test

```

#### acid\_isolation\_test4\_transaction2.tst

```

stringconnect "dsn=tpch;"

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

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

SYNCHRONIZE 2
sleep 5000

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

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

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

execute {select now(*)} into times

```

```

print 'User 2 -- transaction commit complete: ', times
print ' '
print 'USER 2 -- original extendedprice = ', extprice2
print 'USER 2 -- original quantity = ', quantity
print ' '

```

End Test

#### acid\_isolation\_test5\_query.tst

```

stringconnect "dsn=tpch;"

synchronize 2

execute { call generate_ps_values() } into ps_ptky,
ps_spky
print ' '
print 'user 2 ps_partkey = ', ps_ptky
print 'user 2 ps_suppkey = ', ps_spky
print ' '

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

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

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

print ' '

End Test

```

#### acid\_isolation\_test5\_transaction1.tst

```

stringconnect "dsn=tpch;"

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

print ' '
print 'The following are the input values for the
users1 ACID Transaction.'
print 'o_key = ',ordr, ' l_key = ',line, '
delta = ',delta
print ' '
execute {select now(*)} into times
print 'User 1 isolation test time = ', times
print ' '
print ' '
execute {select o_totalprice from orders where
o_orderkey=^}
substitute ordr into o_tprice

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

End Test

### acid\_isolation\_test6\_query.tst

```
stringconnect "dsn=tpch;"
```

```

print 'User1 Query: '
print ' '
print 'User1 starts its query (Q1) here.'
execute {select now(*)} into qstart

```

```

print 'Start time for User1 Q1 =', qstart
print ' '
compare fetchall {select
      l_returnflag,
      l_linestatus,
      sum(l_quantity) as sum_qty,
      sum(l_extendedprice) as sum_base_price,
      sum(l_extendedprice * (1 - l_discount)) as
sum_disc_price,
      sum(l_extendedprice * (1 - l_discount) * (1 +
l_tax)) as sum_charge,
      avg(l_quantity) as avg_qty,
      avg(l_extendedprice) as avg_price,
      avg(l_discount) as avg_disc,
      count(*) as count_order
      from lineitem
      where l_shipdate <= dateadd(day, -1, '1998-12-01')
      group by l_returnflag,l_linestatus
      order by l_returnflag,l_linestatus
      } in 'queryresult'

```

```

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

```

End Test

### acid\_isolation\_test6\_transaction1.tst

```
stringconnect "dsn=tpch;"
```

```

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

```

```

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

```

```

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

```

```

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

```

```

execute {select o_totalprice from orders where
o_orderkey=^ }
      substitute ordr

```



```

        into o_tprice
execute {select l_extendedprice, l_quantity
        from lineitem where l_orderkey=^ and
l_linenumber=^}
        substitute ordr, line
        into l_price, l_quant
print 'User 2 o_totalprice = ', o_tprice
print 'User 2 l_extendedprice = ', l_price,'
l_quantity = ', l_quant
print 'User 2 l_partkey          = ', l_ptky,'
l_suppkey = ', l_spky
print ' '

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

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

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

End Test

```

## Disk Configuration Details

### Solaris Volume Manager Setup

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

Perform the following steps to create the RAID devices:

- 1) create the following 300MB slices
 

```

c1t1d0s7
c1t2d0s7
c1t3d0s7

```

 using the Solaris format command
- 2) create the SVM state replicas via
 

```

metadb -a -f -c 3 c1t1d0s7 c1t2d0s7 c1t3d0s7

```
- 3) create 58.33 GB s7 slice on c1t0d0 and 57GB s3 slices on c1t1d0, c1t2d0, c1t3d0 disks using the Solaris format command
- 4) create 2 mirrored devices via
 

```

metainit d10 1 1 c1t0d0s7
metainit d11 1 1 c1t1d0s3
metainit d15 -m d10 d11

metainit d20 1 1 c1t2d0s3
metainit d21 1 1 c1t3d0s3
metainit d25 -m d20 d21

```

- 5) create the following 1.5 GB slices:
 

```

c1t1d0s5
c1t2d0s5

```

 using the Solaris format command

- 6) create the mirrored volume d1 via the following:

```

metainit d2 1 1 c1t1d0s5
metainit d3 1 1 c1t2d0s5
metainit d1 -m d2 d3

```

### File System Setup

- 7) create a filesystem on /dev/md/rdisk/d1
- 8) mount the filesystem on /sybase2

### Database Device Links

# Create the symbolic links for the mirrored # devices by running the following shellsript

```

cd /sybase2
ln -s /dev/md/rdisk/d15 M01
ln -s /dev/md/rdisk/d25 M02

```

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

```

cd /sybase2
ln -s /dev/rdisk/c1t0d0s1 T01
ln -s /dev/rdisk/c1t1d0s4 T02
ln -s /dev/rdisk/c1t2d0s4 T03
ln -s /dev/rdisk/c1t3d0s4 T04

```

# issue the following commands to change permissions  
chown -hR sybase T\* M\*  
chmod 777 M\* T\*

## Appendix C. Query Text and Query Output

### qualification query 1

```

=====
% select
% l_returnflag,
% l_linestatus,
% sum(l_quantity) as sum_qty,
% sum(l_extendedprice) as sum_base_price,
% sum(l_extendedprice * (1 - l_discount)) as
sum_disc_price,
% sum(l_extendedprice * (1 - l_discount) * (1 +
l_tax)) as sum_charge,
% avg(l_quantity) as avg_qty,
% avg(l_extendedprice) as avg_price,
% avg(l_discount) as avg_disc,
% count(*) as count_order
% from
% lineitem
% where
% l_shipdate <= dateadd(day,-90,'1998-12-01')
% group by
% l_returnflag,
% l_linestatus
% order by
% l_returnflag,
% l_linestatus;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.32000 seconds - current
time 16:40:13
'A','F',37734107,56586554400.7292032,53758257134.86945
63,55909065222.8284717,25.5220058532573342,38273.12973
46211374,.0499852958383577168,1478493
'N','F',991417,1487504710.38000107,1413082168.05409968
,1469649223.19436967,25.5164719205229819,38284.4677608
483374,.0500934266742134809,38854
'N','O',74476040,111701729697.737336,106118230307.6073
83,110367043872.495174,25.5022267695849895,38249.11798
8907361,.049996586053555131,2920374
'R','F',37719753,56568041380.8983326,53741292684.60453
75,55889619119.8339581,25.5057936126907617,38250.85462
60985255,.0500094058300870121,1478870
% total of 4 rows written
=====

```

### qualification query 2

```

=====
% select top 100
% s_acctbal,
% s_name,
% n_name,
% p_partkey,
% p_mfgr,
% s_address,
% s_phone,
% s_comment
% from
% part,
% supplier,
% partsupp,
% nation,
% region
% where
% p_partkey = ps_partkey
% and s_suppkey = ps_suppkey
% and p_size = 15

```

```

% and p_type like 'BRASS'
% and s_nationkey = n_nationkey
% and n_regionkey = r_regionkey
% and r_name = 'EUROPE'
% and ps_supplycost = (
% select
% min(ps_supplycost)
% from
% partsupp,
% supplier,
% nation,
% region
% where
% p_partkey = ps_partkey
% and s_suppkey = ps_suppkey
% and s_nationkey = n_nationkey
% and n_regionkey = r_regionkey
% and r_name = 'EUROPE'
% )
% order by
% s_acctbal desc,
% n_name,
% s_name,
% p_partkey;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.76000 seconds - current
time 16:40:25
9938.53,'Supplier#000005359','UNITED KINGDOM
',185358,'Manufacturer#4
','QKuHYh,vZGiwu2FWEJoLDx04','33-429-790-
6131','blithely silent pinto beans are furiously.
slyly final deposits across'
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
','y30D9UywSTok','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
','t5L67YdBYH6o,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
','VSt3rzK3qG698u6ld8HhObYvvrTcSTsvQlDQDag','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
3459808,405838.6988999999917,'1995-03-04',0
492164,390324.061,'1995-02-19',0
1188320,384537.9358999999976,'1995-03-09',0
2435712,378673.0557999999952,'1995-02-26',0
4878020,378376.7952000000048,'1995-03-12',0
5521732,375153.9215,'1995-03-13',0
2628192,373133.3093999999976,'1995-02-22',0
993600,371407.4594999999994,'1995-03-05',0
2300070,367371.1452000000107,'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              ',1997,50805741.752300003
'ARGENTINA              ',1996,51923746.5754999459
% total of 175 rows written

```

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

```

% select top 20
% c_custkey,% c_name,
% sum(l_extendedprice * (1 - l_discount)) as revenue,
% c_acctbal,
% n_name,
% c_address,
% c_phone,
% c_comment
% from
% customer,
% orders,
% lineitem,
% nation
% where
% c_custkey = o_custkey
% and l_orderkey = o_orderkey
% and o_orderdate >= '1993-10-01'
% and o_orderdate < dateadd(month,3,'1993-10-01')
% and l_returnflag = 'R'
% and c_nationkey = n_nationkey
% group by
% c_custkey,
% c_name,
% c_acctbal,
% c_phone,
% n_name,
% c_address,
% c_comment
% order by
% revenue desc;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.50000 seconds - current
time 16:40:55
57040,'Customer#000057040',734235.2455,632.87,'JAPAN
','Eioyzjf4pp','22-895-641-3466','requests sleep
blithely about the furiously i'
143347,'Customer#000143347',721002.694799999952,2557.4
700000000003,'EGYPT
','laReFYv,Kw4','14-742-935-3718','fluffily bold
excuses haggle finally after the u'
60838,'Customer#000060838',679127.307700000048,2454.77
,'BRAZIL
','64EaJ5vMAHWJlBOxJklpNc2RJiWE','12-913-494-
9813','furiously even pinto beans integrate under the
ruthless foxes; ironic, even dolphins across the slyl'
101998,'Customer#000101998',637029.566699999809,3790.8
9,'UNITED KINGDOM
','01c9CILnNtfOQYmZj','33-
593-865-6378','accounts doze blithely! enticing, final
deposits sleep blithely special accounts. slyly
express accounts pla'
125341,'Customer#000125341',633508.086,4983.5100000000
006,'GERMANY
','S29ODD6bceU8QSuuEJznkNaK','17-582-695-
5962','quickly express requests wake quickly blithely'
25501,'Customer#000025501',620269.784899999976,7725.04
,'ETHIOPIA
','W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ','15-874-808-
6793','quickly special requests sleep evenly among the

```

```

special deposits. special deposi'
115831,'Customer#000115831',596423.867200000167,5098.1
,'FRANCE
','rFeBbEEyk dl
ne7zV5fDrmiqloK09wV7pxqCgIc','16-715-386-
3788','carefully bold excuses sleep alongside of the
thinly idle'
84223,'Customer#000084223',594998.0238999999976,528.65,
'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',585603.3917999999952,5583.02
,'IRAN
','vXCxoCsU0Bad5JQI
,oobkz','20-834-292-4707','express requests sublute
blithely regular requests. regular, even ideas solve.'
39922,'Customer#000039922',584878.1133999999976,7321.10
999999999881,'GERMANY
','Zgy4s5012GKN4pLDPBU8m342gIw6R','17-147-757-
8036','even pinto beans haggle. slyly bold accounts
inte'
6226,'Customer#00006226',576783.7605999999905,2230.09,
'UNITED KINGDOM
','8gPu8,NPGkfyQQ0hcIYUGPIBwC,ybP5g','33-657-701-
3391','quickly final requests against the regular
instructions wake blithely final instructions. pa'
922,'Customer#00000922',576767.5332999999833,3869.25,'
GERMANY
','Az9RFaut7NkPnc5zSD2PwHgVwr4jRzq','17-945-916-
9648','boldly final requests cajole blith'
147946,'Customer#000147946',576455.132,2030.1300000000
003,'ALGERIA
','iANyZHjghyy7AjahOpTrYyhJ','10-886-956-
3143','furiously even accounts are blithely above the
furiouslyl'
115640,'Customer#000115640',569341.1932999999952,6436.1
,'ARGENTINA
','Vtgfia9qI
7EpHgecUlX','11-411-543-4901','final instructions are
slyly according to the'
73606,'Customer#000073606',568656.8577999999952,1785.67
,'JAPAN
','xuR0Tro5yChDfOCrjkd2ol','22-437-653-
6966','furiously bold orbits about the furiously busy
requests wake across the furiously quiet theodolites.
d'
110246,'Customer#000110246',566842.9814999999881,7763.3
5,'VIETNAM
','7KzflgX MDOq7sOkI','31-
943-426-9837','dolphins sleep blithely among the slyly
final'
142549,'Customer#000142549',563537.2367999999952,5085.9
899999999994,'INDONESIA
','ChqEoK43OysjdHbtKCp6dKqjNyvvi9','19-955-562-
2398','regular, unusual dependencies boost slyly;
ironic attainments nag fluffily into the unusual
packages?'
146149,'Customer#000146149',557254.9865,1791.55,'ROMAN
IA
','s87fvzFQpU','29-744-164-
6487','silent, unusual requests detect quickly slyly
regul'
52528,'Customer#000052528',556397.3508999999976,551.79,
'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
129760,17538456.85999999994
166726,16503353.91999999988
191287,16474801.96999999988
161758,16101755.53999999976
34452,15983844.72000000018
139035,15907078.34000000006
9403,15451755.61999999988
154358,15212937.87999999982
38823,15064802.85999999994
85606,15053957.1500000003
% 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#23'
% 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
','Br7elnnntlyxrw6ImgpJ7YdhFDjuBf'
'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#0000002829      ','20
'Supplier#0000005808      ','18
'Supplier#000000262      ','17
'Supplier#0000000496      ','17
'Supplier#0000002160      ','17
'Supplier#0000002301      ','17
'Supplier#0000002540      ','17
'Supplier#0000003063      ','17
'Supplier#0000005178      ','17
'Supplier#0000008331      ','17
```

```
% total of 100 rows written
=====
qualification query 22
=====
% select
% ctrycode,
% count(*) as numcust,
% sum(c_acctbal) as totacctbal
% from
% (
% select
% substring(c_phone,1,2) as ctrycode,
% 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
% ctrycode
% order by
% ctrycode;
% Estimated 1 rows in query (I/O estimate 1010)
% PLAN> vt_1 (seq)
%
%
% 1 record(s) selected -- actual I/O 0
% select time including I/O 0.18000 seconds - current
time 16:42:07
'13',888,6737713.98999999881
'17',861,6460573.72
'18',964,7236687.40000001431
'23',892,6701457.95000000954
'29',948,7158866.62999999642
'30',909,6808436.13000000119
'31',922,6806670.17999998569
% total of 7 rows written
Appendix D. Seed and Query Substitution Parameters
```

This Appendix contains Seed values and substitution parameters for each stream

## Seed Values

```
stream0 0531071002
stream1 0531071003
stream2 0531071004
stream3 0531071005
stream4 0531071006
stream5 0531071007
```

# Query Parameters

## stream0:0531071002

```
=====
14 1993-07-01
2 29 COPPER EUROPE
9 salmon
20 rosy 1995-01-01 VIETNAM
6 1995-01-01 0.05 25
17 Brand#35 SM BAG
18 315
8 RUSSIA EUROPE ECONOMY BRUSHED NICKEL
21 JAPAN
13 pending packages
3 AUTOMOBILE 1995-03-19
22 29 23 20 21 27 11
16
16 Brand#53 ECONOMY BRUSHED 23
39 6 44 34 41 48 1
4 1996-01-01
11 EGYPT 0.0000010000
15 1995-03-01
1 117
10 1994-03-01
19 Brand#14 Brand#35 Brand#33 9
17 25
5 ASIA 1995-01-01
7 IRAN RUSSIA
12 SHIP RAIL 1993-01-01
=====
```

## stream1:0531071003

```
=====
21 EGYPT
3 FURNITURE 1995-03-05
18 312
5 EUROPE 1995-01-01
11 PERU 0.0000010000
7 BRAZIL KENYA
6 1995-01-01 0.02 25
20 cornflower 1993-01-01 IRAQ
17 Brand#32 SM PACK
12 FOB RAIL 1993-01-01
16 Brand#33 SMALL BURNISHED 7
36 45 24 25 42 29 39
15 1997-09-01
13 pending packages
10 1994-12-01
2 17 STEEL AFRICA
8 KENYA AFRICA ECONOMY PLATED NICKEL
14 1993-10-01
19 Brand#11 Brand#13 Brand#32 5
18 22
9 purple
22 21 22 16 20 15 29
30
1 64
4 1993-10-01
=====
```

## stream2:0531071004

```
=====
6 1995-01-01 0.08 24
17 Brand#33 SM DRUM
14 1994-01-01
16 Brand#23 LARGE PLATED 28 31
47 44 37 1 48 3
19 Brand#14 Brand#51 Brand#31
10 19 29
10 1993-09-01
9 peach
2 5 NICKEL EUROPE
15 1995-06-01
=====
```

```
8 FRANCE EUROPE ECONOMY ANODIZED NICKEL
5 MIDDLE EAST 1995-01-01
22 28 20 16 33 19 25
12
12 MAIL TRUCK 1993-01-01
7 ROMANIA FRANCE
13 pending packages
18 314
1 73
4 1996-04-01
20 navy 1997-01-01 ARGENTINA
3 AUTOMOBILE 1995-03-21
11 ETHIOPIA 0.0000010000
21 VIETNAM
=====
```

## stream3:0531071005

```
=====
8 UNITED KINGDOM EUROPE LARGE POLISHED BRASS
5 AFRICA 1996-01-01
4 1994-01-01
6 1996-01-01 0.05 25
17 Brand#35 LG BAG
7 IRAQ UNITED KINGDOM
1 81
18 315
22 17 24 20 21 12 15
32
14 1994-05-01
9 navy
10 1994-07-01
15 1993-03-01
11 CHINA 0.0000010000
20 aquamarine 1995-01-01 MOZAMBIQUE
2 43 TIN AMERICA
21 JORDAN
19 Brand#21 Brand#34 Brand#25 5
20 25
13 pending requests
16 Brand#53 PROMO BRUSHED 29 21
12 33 18 14 30 6
12 TRUCK AIR 1993-01-01
3 FURNITURE 1995-03-07
=====
```

## stream4:0531071006

```
=====
5 AMERICA 1996-01-01
21 FRANCE
14 1994-08-01
19 Brand#23 Brand#22 Brand#24 1
10 21
15 1995-10-01
17 Brand#32 LG PACK
12 RAIL TRUCK 1994-01-01
6 1996-01-01 0.03 25
4 1996-08-01
9 metallic
8 MOROCCO AFRICA LARGE BURNISHED BRASS
16 Brand#33 MEDIUM ANODIZED 45
26 15 12 14 40 17 39
11 FRANCE 0.0000010000
2 30 COPPER EUROPE
10 1993-04-01
18 313
1 89
13 unusual requests
7 CANADA MOROCCO
22 30 21 29 10 32 18
33
3 MACHINERY 1995-03-23
20 lavender 1994-01-01 ETHIOPIA
=====
```

## stream5:0531071007

---

21 UNITED KINGDOM  
 15 1993-06-01  
 4 1994-05-01  
 6 1996-01-01 0.08 24  
 7 SAUDI ARABIA GERMANY  
 16 Brand#23 ECONOMY PLATED 17 2 7  
 42 15 28 27 12  
 19 Brand#25 Brand#55 Brand#24 6  
 11 28  
 18 314  
 14 1994-11-01  
 22 13 29 19 16 23 10  
 18  
 11 ROMANIA 0.0000010000  
 13 unusual requests  
 3 FURNITURE 1995-03-09  
 1 97  
 2 18 STEEL AMERICA  
 5 ASIA 1996-01-01  
 8 GERMANY EUROPE MEDIUM BRUSHED BRASS  
 20 slate 1997-01-01 SAUDI ARABIA  
 12 REG AIR TRUCK 1994-01-01  
 17 Brand#34 LG DRUM  
 10 1994-01-01  
 9 light

## Appendix E. Implementation-Specific Layer/Driver Code

```
=====
=====
nctest
=====

dbisqlc -c "DSN=tpch" -q load_lineitem.sql >
load_lineitem.out &
loadlpid=$!
sleep 2200
dbisqlc -c "DSN=tpch" -q load_region.sql
dbisqlc -c "DSN=tpch" -q load_nation.sql
dbisqlc -c "DSN=tpch" -q load_customer.sql
dbisqlc -c "DSN=tpch" -q load_part.sql
dbisqlc -c "DSN=tpch" -q load_supplier.sql
dbisqlc -c "DSN=tpch" -q load_partsupp.sql
dbisqlc -c "DSN=tpch" -q load_orders.sql
loadopid=$!
wait $loadopid
wait $loadlpid
end_load.out
echo " "
seed=`date +%m%d%H%M%S`;
echo $seed;
./gen_streams.ksh $seed 1000
dbisqlc -c "DSN=tpch" -q dbtables-syb.sql >
rdtablest.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 -c "DSN=tpch" -q check_options.sql >
check_options_start.out

touch /export/home/sybase/run/scripts/rf1.lock
touch /export/home/sybase/run/scripts/rf2.lock
dbisqlc -c "DSN=tpch" -q update_power.sql >
update_power.out &
rfspid=$!
while [ -f /export/home/sybase/run/scripts/rf1.lock ]
do
sleep 10
done

dbisqlc -c "DSN=tpch" -q stream0.sql > stream0.out

rm -f /export/home/sybase/run/scripts/rf2.lock
wait $rfspid

dbisqlc -c "DSN=tpch" -q stream1.sql > stream1.out &
dbisqlc -c "DSN=tpch" -q stream2.sql > stream2.out &
dbisqlc -c "DSN=tpch" -q stream3.sql > stream3.out &
dbisqlc -c "DSN=tpch" -q stream4.sql > stream4.out &
dbisqlc -c "DSN=tpch" -q stream5.sql > stream5.out &

dbisqlc -c "DSN=tpch" -q update_throughput5.sql >
update_throughput.out &
wait

mv stream0.out mls00q.out
mv update_power.out mls00rf.out
mv stream1.out mls01q.out
mv stream2.out mls02q.out
mv stream3.out mls03q.out
mv stream4.out mls04q.out
mv stream5.out mls05q.out
```

```
mv update_throughput.out mls01rf.out

touch /export/home/sybase/run/scripts/rf1.lock
touch /export/home/sybase/run/scripts/rf2.lock

dbisqlc -c "DSN=tpch" -q update_power.sql >
update_power.out &
rfspid=$!
while [ -f /export/home/sybase/run/scripts/rf1.lock ]
do
sleep 10
done

dbisqlc -c "DSN=tpch" -q stream0.sql > stream0.out
rm -f /export/home/sybase/run/scripts/rf2.lock
wait $rfspid

dbisqlc -c "DSN=tpch" -q stream1.sql > stream1.out &
dbisqlc -c "DSN=tpch" -q stream2.sql > stream2.out &
dbisqlc -c "DSN=tpch" -q stream3.sql > stream3.out &
dbisqlc -c "DSN=tpch" -q stream4.sql > stream4.out &
dbisqlc -c "DSN=tpch" -q stream5.sql > stream5.out &

dbisqlc -c "DSN=tpch" -q update_throughput20.sql >
update_throughput.out &
wait

dbisqlc -q -c "DSN=tpch" -q dbtables-syb.sql >
rdtablest_end.out
dbisqlc -q -c "DSN=tpch" -q dew_cat1.sql >
dew_cat1_end.out
dbisqlc -q -c "DSN=tpch" -q dew_cat2.sql >
dew_cat2_end.out
dbisqlc -q -c "DSN=tpch" -q dew_cat3.sql >
dew_cat3_end.out
dbisqlc -q -c "DSN=tpch" check_options.sql >
check_options_end.out

mv stream0.out m2s00q.out
mv update_power.out m2s00rf.out
mv stream1.out m2s01q.out
mv stream2.out m2s02q.out
mv stream3.out m2s03q.out
mv stream4.out m2s04q.out
mv stream5.out m2s05q.out

mv update_throughput.out m2s01rf.out

exit

=====
tpch_wait.sql
=====

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

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

```
declare local temporary table t_iq_spaceused(
mainKB unsigned bigint,
mainKBUsed unsigned bigint,
tempKB unsigned bigint,
```

```

tempKBUsed unsigned bigint,
)
in SYSTEM on commit preserve rows;

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

select 'xp_cmdshell 'sleep 120'' into command;

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

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

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

if ( mainfree > 12400000 )
then leave waitloop;
end if;

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

END LOOP waitloop;

drop table t_iq_spaceused;
commit;
end
;

```

## Appendix F. Misc database scripts

The dbtables-syb.sql script was run to validate the correctness of the database after the database load. Three other scripts were used to extract basic information about tables and indexes from the database dew\_cat1.sql, dew\_cat2.sql, dew\_cat3.sql.

### Auditor Scripts

#### dbtables-syb.sql

```
=====
-- FILENAME
-- DBTABLES.SQL
-- DESCRIPTION
-- CHECK ROW COUNT AND ROW STRUCTURE/CONTENT FOR
EACH TABLE
-- IN THE TPC-H DATABASE.
--
-- =====
--
-- GET TIMESTAMP
SELECT 'START TIME', CONVERT(CHAR(30), GETDATE(),
120);
go
--
-- TABLE: LINEITEM
-- =====
SELECT COUNT(*) FROM LINEITEM;
go
SELECT * FROM LINEITEM
WHERE L_ORDERKEY IN
( 4, 26598, 148577, 387431, 56704, 517442,
600000)
AND L_LINENUMBER = 1
ORDER BY L_ORDERKEY;
go
--
-- TABLE: ORDERS
-- =====
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM ORDERS;
go
SELECT * FROM ORDERS
WHERE O_ORDERKEY IN ( 7, 44065, 287590, 411111,
483876, 599942 )
ORDER BY O_ORDERKEY;
go
--
-- TABLE: PART
-- =====
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM PART;
go
SELECT * FROM PART
WHERE P_PARTKEY IN (1,984,8743,9028,13876,17899,20000)
ORDER BY P_PARTKEY;
go
--
-- TABLE: PARTSUPP
-- =====
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM PARTSUPP;
go
SELECT* FROM PARTSUPP
```

```
WHERE PS_PARTKEY = 3398
AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
FROM PARTSUPP WHERE PS_PARTKEY = 3398);
go
SELECT* FROM PARTSUPP
WHERE PS_PARTKEY =15873
AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
FROM PARTSUPP WHERE PS_PARTKEY = 15873);
go
SELECT* FROM PARTSUPP
WHERE PS_PARTKEY = 11394
AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
FROM PARTSUPP WHERE PS_PARTKEY = 11394);
go
SELECT* FROM PARTSUPP
WHERE PS_PARTKEY = 6743
AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
FROM PARTSUPP WHERE PS_PARTKEY = 6743);
go
SELECT* FROM PARTSUPP
WHERE PS_PARTKEY = 19763
AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY) FROM
PARTSUPP WHERE PS_PARTKEY =19763);
go
--
-- TABLE: SUPPLIER
-- =====
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM SUPPLIER;
go
SELECT * FROM SUPPLIER
WHERE S_SUPPKEY IN (83,265,492,784,901,1000)
ORDER BY S_SUPPKEY;
go
--
-- TABLE: CUSTOMER
-- =====
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT COUNT(*) FROM CUSTOMER;
go
SELECT * FROM CUSTOMER
WHERE C_CUSTKEY IN (832,2653,4924,7845,92016,108070)
ORDER BY C_CUSTKEY;
go
--
-- TABLE: NATION & REGION
-- =====
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
SELECT * FROM REGION;
go
SELECT COUNT(*) FROM NATION;
go
SELECT * FROM NATION
WHERE N_NATIONKEY IN (3,10,14,20)
ORDER BY N_NATIONKEY;
go
--
-- CHECK KEY VALUES
-- =====
-- GET TIMESTAMP
SELECT 'TIME', CONVERT(CHAR(30), GETDATE(), 120);
go
if exists (select name from sysobjects where
name='MINMAX')
drop table MINMAX
go
CREATE TABLE MINMAX
(TNAME CHAR(15),
KEYMIN INTEGER,
KEYMAX INTEGER);
go
INSERT INTO MINMAX
```

```

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

```

```

=====
dew_cat1.sql
=====

```

```

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

```

```

=====
dew_cat2.sql
=====

```

```

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

```

```

=====
dew_cat3.sql
=====

```

```

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

```

## **Appendix G. Pricing information**

---

Price quotes from Sybase Inc and Sun Microsystems Inc are included below.



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

**Quotation for Software and Support**

**SYBASE Sales** IHollie Nash  
**Phone:** 972-687-6412  
**Fax:** 972-687-6409

CBSS#

	Catalogue Number	Product Description	License Type	Machine	P/S	List Price Per Unit	Quantity	Discount	Extended Price	Extended Support Fees
										3 YEARS
1	12841	Sybase IQ Single App Svr, per cpu core	CP	Sun	P	2,595	2		5,190.00	
3	98480	3 yr support Single App Svr, per cpu core				1,557	2			3,114.00
4		Discount						0		
5		10% if license + support > 50000								
6		5% if license + support > 25000								
7		0 otherwise								
8										
9										
10										
11										
12										

**Quote Date:** 6/21/06  
**Valid thru:**

**Total** 8,304.00

Licence + 3 year support

Payment terms : Net 30 Days

5400 LBJ Freeway, Suite 1500, Dallas, TX 75240



# Sales Quotation

Quote Number: T-US-853735-A

Quote Date: 6/21/06

Customer : RICHARD GOSTANIAN  
 SUN MICROSYSTEMS INC  
 1 NETWORK DRIVE  
 BURLINGTON MA 01803  
 Tel / Fax : 7814423063 /

Sun : Holli Babb  
 Sun Microsystems, Inc.  
 1617 Southwood Drive  
 Nashua NH 03063  
 Tel / Fax : 781-442-0906/781-394-0478

We are pleased to quote as follows:

Validity Period
60 Days

Credit Terms
Net 30 Days

Shipping Terms
Origin

Item	Product Number	Description	Qty	Unit List Price	Disc	Unit Net Price	Extended Net Price
SunFire X4100 Server							
1	Config ID 4540351	Configuration: A64-ZD	1	\$12,397.00	N/A	\$10,753.05	\$10,753.05
1.1	A64-ZD	Sun Fire X4100 x64 Server: XATO base chassis for 4-disk system configuration with no DVD, including motherboard, 1x PSU, redundant fans and Service Processor. RoHS-5.	1	\$1,495.00	7.00%	\$1,390.35	\$1,390.35
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	RA-SS2CD-73G10KZ	73GB 10K RPM 2.5" SAS hard disk drive. XATO & FACTORY INTEGRATION. RoHS-6.	4	\$445.00	7.00%	\$413.85	\$1,655.40
1.4	8048A-Z	Power supply unit filler panel for Sun Fire X4100 and X4200 x64 servers. XATO option. ROHS-5.	1	N/C	7.00%	N/C	N/C

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

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





# Sales Quotation

Quote Number: T-US-853735-A

Quote Date: 6/21/06

Item	Product Number	Description	Qty	Unit List Price	Disc	Unit Net Price	Extended Net Price
1.5	8035A-Z	AMD Opteron 256 (3.0 GHz/1MB) processor for Sun Fire X4100 and X4200 x64 servers.	2	\$995.00	7.00%	\$925.35	\$1,850.70
1.6	8023A-Z	4 GB Memory kit DDR1-400 Registered ECC DIMMs (2x2GB) for Sun Fire X4100 and X4200 x64 servers. RoHS-5. XATO & FACTORY INTEGRATION.	4	\$1,090.00	7.00%	\$1,013.70	\$4,054.80
1.7	W9D-A64-3G	Sun Fire X4100 Server upgrade to 3 years of Gold support.	1	\$2,772.00	35.00%	\$1,801.80	\$1,801.80

List Price Total:	\$12,397.00
-------------------	-------------

Total:	\$10,753.05
--------	-------------

