HP NetServer LXr 8500
using
Microsoft Windows 2000
and
Microsoft SQL Server 2000

TPC Benchmark® H
Full Disclosure Report

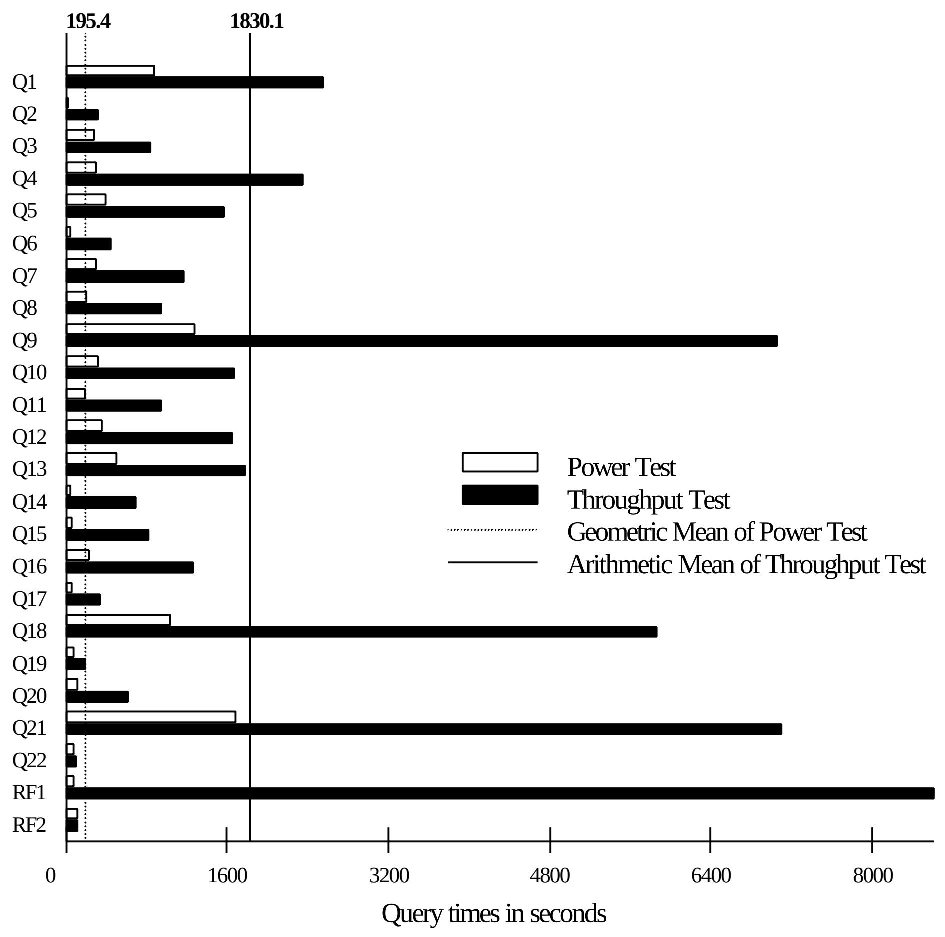
Second Edition

Submitted for Review
August 18, 2000

First Edition - August 18, 2000

HEWLETT® PACKARD	**HP NetServer LXr 8500 Microsoft SQL Server 2000**	**TPC-H Rev 1.2** / **Report Date: 18-Aug-2000**
Total System Cost **$251,537**	Composite Query per Hour Metric **1,291.4** **QphH @ 100 Gbyte**	Price Performance **$195** **QphH @ 100 Gbyte**

Database Size	Database Manager	Operating System	Other software	Availability date
100 Gbyte	**Microsoft SQL Server 2000**	**Microsoft Windows 2000**	**Microsoft Visual C++**	**14-August-2000**



Database Load Time = 10:35	Load Includes Backup: Y	Total Data Storage / Database Size =15.79
RAID (Base tables only): N	RAID (Base tables and auxiliary data structures): N	RAID (All): N

Number of nodes	1
Processors	8 Intel Pentium III Xeon 550Mhz each w/ 2Mbyte L2 cache
Memory	4 Gbyte
Disk Drives	180 x 8.678 Gbyte and 2 x 8.678 Gbyte
Controllers	7 Adaptec 39160 (2 port) SCSI, 1 HP Fiber Channel Disk Array Pair
Total Disk Storage	1.58 Tbyte
NIC	1 HP D5013A 10/100 TX Network Card
Tape Drives	n/a
CD ROM	Integrated

HP NetServer LXr 8500
Microsoft SQL Server 2000

TPC-H Rev 1.2

Report Date: 18-Aug-2000

Description	Part Number	Brand	Price Key	Unit Price	Qty	Extended Price	5 yr. Maint. Price
Server Hardware							
HP NetServer LXr 8500	D8542A	HP	1	16,290	1	16,290	
Intel Pentium III Xeon 550MHz 2Mbyte	D8531A	HP	1	5,590	7	39,130	
HP NetServer LXr8500 Memory Carrier Card	D7071A	HP	1	680	1	680	
256MB Dimm for LXr 8500	D9325A	HP	1	739	15	11,085	
Adaptec SCSI Card 39160	18223000	Adaptec	1	319	7	2,233	
HP Fiber Host Bus Adapter	D8602A	HP	1	1,349	1	1,349	
HP NetServer 10/100TX PCI LAN Adapter	D5013A	HP	1	82	1	82	
HP 9 GB 10K HotSwap Wide Ultra2 SCSI Disk	D6107A	HP	1	430	2	860	
HP 17" Display	D2828A	HP	1	185	1	185	
HP NetServer mini-DIN keyboard and mouse	D4950B/C3751B	HP	1	79	1	79	
HP Rack System/E33 (33 EIA units usable space)	J1501A	HP	1	1,680	2	3,360	
					Subtotal	75,333	0
Server Software							
Microsoft SQL Server 2000 Enterprise Edition (50 user license)		MS	2	10999	1	10,999	10475
Microsoft Windows 2000 Advanced Server		MS	2	3999	1	3,999	
Microsoft Visual C++ 6.0	716856	MS	2	549	1	549	
					Subtotal	15,547	10,475
Storage Devices							
HP NetServer Rack Storage/12FC	D5991A	HP	1	6,159	1	6,159	
HP Fibre Channel Controller	D5990A	HP	1	4,450	1	4,450	
HP Fibre Channel Hub	D6976A	HP	1	3,130	1	3,130	
HP NetServer Rack Storage/12	D5989B	HP	1	1,890	14	26,460	
SCSI Cable, 2.5m 68 pin UHD to 68 pin HD	D6020A	HP	1	97	14	1,358	
HP 9 GB 10K HotSwap Wide Ultra2 SCSI Disks	D6107A	HP	1	430	180	77,400	
APC SmartUPS 3000NS 208V 3000VA	588293	APC	1	1,725	1	1,725	
HP System Support 5 yrs of 4 hr response M-F includes server and storage subsystem	H2826VV	HP	1	29,500	1		29,500
					Subtotal	120,682	29,500
					Total	**$211,562**	**$39,975**

Notes:
Price key: 1=Software House International, 2=Microsoft
Software and hardware available now.

5-yr Cost of Ownership:	**$251,537**
QphH @ 100 Gbyte:	**1291.4**
$/QphH @ 100 Gbyte:	**$ 195**

Audited by Francois Raab, Infosizing. Inc.

Prices used in TPC benchmarks reflect actual prices a customer would pay for a one-time purchase of the stated components. Individually negotiated discounts are not permitted. Special prices based on assumptions about past or future purchases 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.

HEWLETT® PACKARD	HP NetServer LXr 8500 Microsoft SQL Server 2000	TPC-H Rev 1.2 / Report Date: 18-Aug-2000

Numerical Quantities

Measurement Results

Scale Factor	=	100
Total data storage/database size	=	15.79
Database load time	=	10:35
Query streams for throughput test	=	5
TPC-H Power Metric (QppH @ 100 Gbyte)	=	1842.7
TPC-H Throughput Metric (QthH @ 100 Gbyte)	=	905.0
Composite Metric (QphH @ 100 Gbyte)	=	1291.4
Total system price over 5 years	=	$251,537
TPC-H Price Performance Metric ($/QphH @ 100 Gbyte)	=	$195

Measurement Intervals

Measurement Interval in Throughput Test (Ts) = 43,757 seconds

Duration of stream execution:

	Seed	Query Start Date/Time Query End Date/Time	RF1 Start Date/Time RF1 End Date/Time	RF2 Start Date/Time RF2 End Date/Time	**Duration**
Stream 00	210105040	02/11/00 03:20:43 02/11/00 05:45:52	2/11/00 03:20:43 2/11/00 03:22:10	2/11/00 05:43:57 2/11/00 05:45:52	2:25
Stream 01	210105041	02/11/00 05:45:54 02/11/00 16:14:58	2/11/00 05:45:53 2/11/00 17:37:53	2/11/00 17:37:53 2/11/00 17:39:57	10:29
Stream 02	210105042	02/11/00 05:45:54 02/11/00 17:26:13	2/11/00 17:39:57 2/11/00 17:41:43	2/11/00 17:41:43 2/11/00 17:43:45	11:40
Stream 03	210105043	02/11/00 05:45:54 02/11/00 17:36:14	2/11/00 17:43:45 2/11/00 17:45:31	2/11/00 17:45:31 2/11/00 17:47:34	11:50
Stream 04	210105044	02/11/00 05:45:54 02/11/00 17:05:51	2/11/00 17:47:35 2/11/00 17:49:19	2/11/00 17:49:19 2/11/00 17:51:22	11:19
Stream 05	210105045	02/11/00 05:45:54 02/11/00 16:21:24	2/11/00 17:51:22 2/11/00 17:53:07	2/11/00 17:53:07 2/11/00 17:55:11	10:35

HEWLETT® PACKARD

HP NetServer LXr 8500 Microsoft SQL Server 2000

TPC-H Rev 1.2

Report Date: 18-Aug-2000

TPC-H Timing Intervals (in seconds)

Duration of stream execution:

Stream ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Stream 00	882.2	18.4	274.8	302.7	399.4	42.0	302.8	213.0	1289.5	327.9	187.5	360.5
Stream 01	2922.0	490.4	1211.3	1336.0	1743.0	68.3	1231.8	1271.1	6241.4	1357.3	1115.5	2351.6
Stream 02	2527.2	50.4	926.5	1455.1	899.9	253.9	1385.9	1307.4	6331.6	2191.4	1092.5	2327.7
Stream 03	2275.6	511.8	285.0	4747.2	1623.3	665.8	1689.7	312.3	7453.9	1893.5	1414.2	396.8
Stream 04	2757.4	67.7	543.7	2461.8	1927.5	653.4	569.8	848.8	7579.3	1194.5	488.7	1419.1
Stream 05	2301.4	442.1	1201.5	1726.9	1675.2	567.5	1004.9	960.9	7744.3	1695.8	626.9	1751.6
Minimum	2275.6	50.4	285.0	1336.0	899.9	68.3	569.8	312.3	6241.4	1194.5	488.7	396.8
Average	2556.7	312.5	833.6	2345.4	1573.8	441.8	1176.4	940.1	7070.1	1666.5	947.6	1649.4
Maximum	2922.0	511.8	1211.3	4747.2	1927.5	665.8	1689.7	1307.4	7744.3	2191.4	1414.2	2351.6

Stream ID	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream 00	507.0	36.7	63.9	236.7	64.6	1044.3	74.3	118.7	1680.0	81.1	87.3	114.9
Stream 01	2793.6	312.2	270.3	651.2	151.6	6302.8	145.5	323.2	5330.6	123.8	42720.0	123.6
Stream 02	2860.0	1300.8	372.3	3685.7	482.1	5094.0	126.3	972.8	6273.6	102.4	105.5	122.4
Stream 03	728.4	141.1	1630.9	326.4	251.5	6218.4	90.3	847.5	9009.5	106.9	105.9	123.1
Stream 04	1195.8	1597.0	192.8	761.3	479.0	5672.9	274.9	297.0	9709.2	103.8	104.9	122.9
Stream 05	1293.9	112.4	1634.4	899.9	296.4	6012.2	320.0	600.0	5164.3	98.1	105.2	123.3
Minimum	728.4	112.4	192.8	326.4	151.6	5094.0	90.3	297.0	5164.3	98.1	104.9	122.4
Average	1774.3	692.7	820.1	1264.9	332.1	5860.1	191.4	608.1	7097.4	107.0	8628.3	123.1
Maximum	2860.0	1597.0	1634.4	3685.7	482.1	6302.8	320.0	972.8	9709.2	123.8	42720.0	123.6

TPC Benchmark H Overview

The TPC Benchmark ™ H (TPC-H) is a decision support benchmark. It consists of a suite of business oriented ad-hoc queries and concurrent updates. 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(e.g., via a point and click GUI interface);
* 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.

Hewlett-Packard Company does not warrant or represent that a user can or will achieve performance similar to the benchmark results contained in this report. No warranty of system performance or price/performance is expressed or implied by this report.

1 General Items

1.1 Test Sponsor

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

Hewlett-Packard Company is the test sponsor of this TPC Benchmark H.

1.2 Parameter Settings

Settings must be provided for all customer-tunable parameters and options which 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.*

Comment 1: In the event that some parameters and options are set multiple times, it must be easily discernible by an interested reader when the parameter or option was modified and what new value it received each time.

Comment 2: This requirement can be satisfied by providing a full list of all parameters and options, as long as all those which have been modified from their default values have been clearly identified and these parameters and options are only set once.

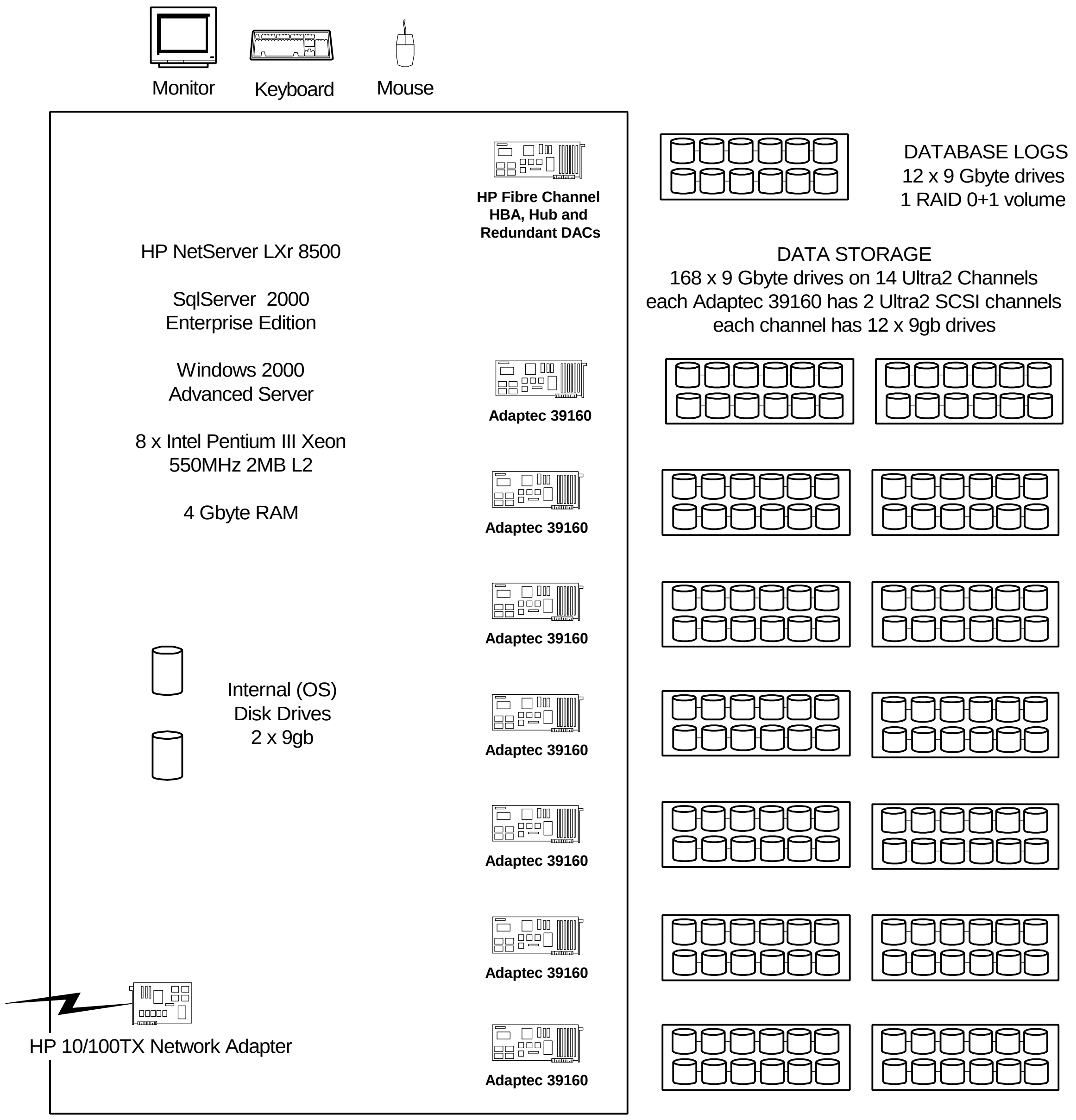
Appendix A contains the Windows NT Server and SqlServer parameters used in this benchmark.

1.3 Configuration Diagrams

Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:

- *Number and type of processors;*
- *Size of allocated memory, and any specific mapping/partitioning of memory unique to the test;*
- *Number and type of disk units (and controllers, if applicable);*
- *Number of channels or bus connections to disk units, including their protocol type;*
- *Number of LAN (e.g. Ethernet) connections, including routers, workstations, terminals, etc., that were physically used in the test or are incorporated into the pricing structure;*
- *Type and run-time execution location of software components (e.g. DBMS, query processing tools/languages, middleware components, software drivers, etc.).*

The server System Under Test (SUT), an HP NetServer LXr 8500 is depicted on the next page:


Monitor
Keyboard
Mouse
HP Fibre Channel HBA, Hub and Redundant DACs
DATABASE LOGS
12 x 9 Gbyte drives
1 RAID 0+1 volume
HP NetServer LXr 8500
SqlServer 2000 Enterprise Edition
Windows 2000 Advanced Server
8 x Intel Pentium III Xeon 550MHz 2MB L2
4 Gbyte RAM
DATA STORAGE
168 x 9 Gbyte drives on 14 Ultra2 Channels
each Adaptec 39160 has 2 Ultra2 SCSI channels
each channel has 12 x 9gb drives
Adaptec 39160
Adaptec 39160
Adaptec 39160
Adaptec 39160
Adaptec 39160
Adaptec 39160
Adaptec 39160
Internal (OS) Disk Drives 2 x 9gb
HP 10/100TX Network Adapter

The HP NetServer LXr 8500 system consisted of:

* 8 Intel Pentium III Xeon 550Mhz processors, each with 2Mbyte L2 cache
* 4 Gbyte RAM
* 7 Adaptec 39160 SCSI (2 port) Cards + 1 HP Fiber Channel Disk Array Pair
* 1 HP 10/100 TX Network Interface Card
* 180 HP Hot-Swap 9 Gbyte Disk Drives
* 2 HP Hot-Swap 9 Gbyte Disk Drive
* 15 HP RS/12 Storage Enclosures

2 Clause 1 Logical Database Design Related Items

2.1 Database Table Definitions

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

Appendix B describes the scripts that define and create the tables and indices for the TPC-H database.

2.2 Physical Organization of Database

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.

Appendix B contains the database and table creation statements. Clustered indexes were used on the LINEITEM (L_SHIPDATE) and ORDERS(O_ORDERDATE) tables. Default column ordering was used.

2.3 Horizontal Partitioning

Horizontal partitioning of tables is allowed. Groups of rows from a table may be assigned to different files, disks, or areas.… 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.

2.4 Replication

While there are some restrictions placed upon physical replication of objects in the test and qualification databases (see Clause 1.5.6), any such replication must be disclosed.

No replication of the base tables 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 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 2.1.8.1 of DBGEN and QGEN were used for this TPC-H benchmark.

3.3 Substitution Parameters Generation

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.

QGEN version 2.1.8.1 was used to generate the substitution parameters.

3.4 Query Text and Output Data from 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 definition 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 D contains the actual query text and query output. The following allowed minor query modifications were used in this implementation:

- 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. “YY”.
- In Q2, Q3, Q10, Q18 and Q21, the “top” function is used to restrict the number of output rows.

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 E 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 transaction were run with isolation level 1.

3.7 Source Code of Refresh Functions

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

Appendix G contains source code for the refresh functions.

4 Clause 3 Database System Properties Related Items

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.

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 ORDER, LINEITEM, and HISTORY tables.

1. The total price from the ORDER 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 ORDER 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 ORDER, LINEITEM, and HISTORY tables.

1. The total price from the ORDER 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 ORDER 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 ORDER 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 ORDER and LINEITEM 4.2.1

1. The consistency of the ORDER and LINEITEM tables was verified based on a sample of O_ORDERKEYs.
2. 100 ACID Transactions were submitted from each of 2 execution streams.
3. The consistency of the ORDER 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 some 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. The ACID Query blocked and did not see any uncommitted changes made by the ACID Transaction.
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. The ACID transaction 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. The ACID transaction 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 on Different Tables

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

1. An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. T1 was suspended prior to COMMIT.
2. Another ACID transaction, T2 was started using random values for PS_PARTKEY and PS_SUPPKEY.
3. ACID Transaction T2 completed.

4. ACID transaction T1 completed and the appropriate rows in the ORDER, LINEITEM, and HISTORY tables have been changed.

4.4.6 Updates not Indefinitely Delayed by Reads on Same Table

Demonstrates 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. An ACID transaction, T1, was started, executing Q1 against the qualification database. The substitution parameter was chosen from the interval [0..2159] so that the query ran for a sufficient length of time.
2. Before T1 completed, an ACID transaction, T2, was started using randomly selected values of O_KEY, L_KEY and DELTA.
3. T2 completed before T1 completed. Verified that the appropriate rows in ORDER, LINEITEM and HISTORY tables have been changed.

4.5 Durability

The tested system 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 3.5.2.

4.5.1 Failure of a Durable Medium

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

The database logs were stored on an HP Fibre RAID 0+1 volume made up of 12 physical drives mirrored by a redundant pair of disk array controllers. The tables for the database were stored on RAID 0 disks. The backup was done twice, producing one set of backup files on a one set of RAID0 disks, then another set of backup files on a different set of disks.

1. The database was backed up twice, do different sets of disk drives.
2. Six streams of ACID transactions were started.
3. While the test was running the primary disk array controller for the log volume was removed. The secondary disk array controller automatically assumed the primary disk array controller function.
4. While the test was running one physical drive of the RAID 1+0 log volume was removed.
5. After it was determined that the test would still run with the loss of a log disk, one physical drive of a RAID 0 data volume was removed.
6. The six streams of ACID transactions failed and recorded their number of committed transactions in success files.
7. The logfile disk array controller, the logfile disk and the data/backup file disk were replaced. The RAID 0+1 logfile volume and the RAID0 data volume were rebuilt.
8. The data files were restored to their state prior to the ACID transaction streams. The failed physical drive affected only one of the two backup copies, so a database restore was done using the unaffected copy.
9. The database ran through its roll forward recovery.
10. The counts in the success files and the HISTORY table count were compared. The counts matched.

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.

1. Six streams of ACID transactions were started.
2. While the streams of ACID transactions were running the system was powered off.
3. When power was restored the system rebooted and the database was restarted.
4. The database went through a recovery period.
5. The success file and the HISTORY table counts were compared, and they matched.

4.5.3 Memory Failure

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

See the previous section.

5 Clause 4 Scaling and Database Population Related Items

5.1 The Cardinality of Tables

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

Table	Cardinality
ORDER	150000000
LINEITEM	600037902
CUSTOMER	15000000
PART	20000000
SUPPLIER	1000000
PARTSUPP	80000000
NATION	25
REGION	5

5.2 Distribution of Tables and Logs Across Media

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

Disk drives for the database tables, temporary space, indexes, flatfiles and backup files were controlled by individual SCSI disk drives configured as seven NT logical disk volumes. Each NT logical disk drive is configured as twenty-four disk drives, using NT dynamic striped volumes.

The database tables, temporary space, indexes and flatfiles were evenly spread across the seven data volumes. The database backup files were stored on the same physical data volumes as the database. One set of backup files were placed on the odd numbered data volumes, while another set of backup files were placed on the even numbered data volumes.

The database logs were placed on a RAID 1+0 (HP Fibre channel terminology for RAID 1 + RAID 0) volume made up of 12 physical drives. The log drives were separate from the database drives.

The operating system, SqlServer binaries and all benchmark execution software were installed on two 9Gbyte internal drives.

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 + RAID 0 was used for log disks. RAID 0 was used for all the other database disks.

5.5 Modifications to 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 TPC supplied DBGEN version 2.1.8.1 was used to generate the database population for this benchmark. No modifications were made.

5.6 Database Load Time

The database load time for the test database (see clause 4.3) must be disclosed.

See the executive summary at the beginning of this report.

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:

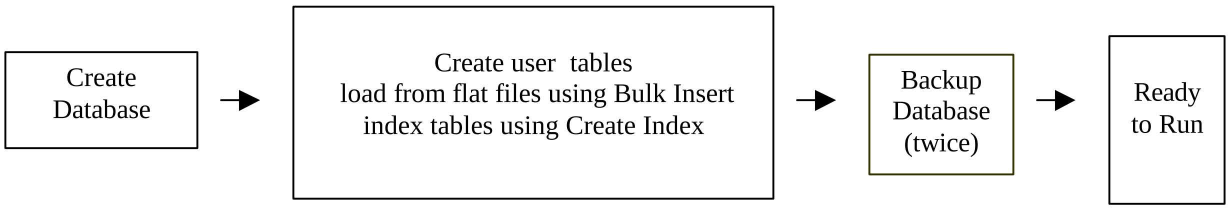
Type of Disk	HP Hot-Swap 9 Gbyte	HP Hot-Swap 9 Gbyte	Grand Total
Numer of disks	180	2	
Size (GB)	8.678	8.678	
Total GB	1562.04	17.356	1579
Scale Factor			100
Storage ratio			15.79

5.8 Database Load Mechanism Details and Illustration

The details of the database load must be disclosed, including a block diagram illustrating the overall process. Disclosure of the load procedure includes all steps, scripts, input and configuration file required to completely reproduce the test and qualification databases.

DBGEN was used to create flat files which were then loaded into the tables the SqlServer "bulk insert" command. Indexes were created. Next, a database backup was done twice, with each backup going to different disk drives.

The insert of rows into the database was accomplished by running eight concurrent threads, each of which performs a "bulk insert" operation that loads one eighth of each of the LINEITEM, ORDERS, PART, PARTSUPP, SUPPLIER and CUSTOMER tables. The NATION and REGION tables were loaded serially, each by a single thread. After each table load, indexes were created, first the clustered index, if defined, followed by any non-clustered indexes, if defined.


Create Database
Create user tables
load from flat files using Bulk Insert
index tables using Create Index
Backup Database (twice)
Ready to Run

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 and disk structure to create and load the data with adjustments for the size difference.

6 Clause 5 Performance Metrics and Execution-Rules Related Items

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

- Auditor requested queries were run against the database to verify correctness of the database load
- The database server was re-started

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 Function

The timing intervals for each query of the measured set and for both update functions must be reported for the power test.

The timing intervals for the each query and both update functions are given in the Numerical Quantities Summary earlier in this document.

6.4 Number of Streams for the Throughput Test

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

Five streams were used for the Throughput Test.

6.5 Start and End Date/Times for Each Query Stream

The start time and finish time for each query execution stream must be reported for the throughput test.

The throughput test start time and finish time for each stream are given in the Numerical Quantities Summary earlier in this document.

6.6 Total Elapsed Time for 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 given in the Numerical Quantities Summary earlier in this document.

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

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

The start and finish time for each update function in the update stream are given in the Numerical Quantities Summary earlier in this document.

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

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

The timing intervals for each query and each update function are given in the Numerical Quantities Summary earlier in this document.

6.9 Performance Metrics

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

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

6.10 The Performance Metric and Numerical Quantities from Both Runs

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

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

	QppH @ 100 Gbyte	QthH @ 100 Gbyte	QphH @ 100 Gbyte
Run1	1830.5	919.0	1297.0
Run2	1842.7	905.0	1291.4
% Difference	0.7%	1.5%	0.4%

6.11 System Activity Between Tests

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

The database server was restarted between runs.

7 SUT and Driver Implementation Related Items

7.1 Driver

A detailed textual description of how the driver performs its functions, how its various components interact and any product functionalities or environmental setting on which it relies must be provided. All related source code, scripts and configuration files must be disclosed. The information provided should be sufficient for an independent reconstruction of the driver.

Two scripts were used. The first one was used to create and load the database, while the second was used to run the Power and Throughput tests. These scripts are in Appendix F. A C program, semaphore.c, was used for coordination of parallel processes.

7.2 Implementation-Specific Layer (ISL)

If an implementation specific layer is used, then a detailed description of how it performs its functions, how its various components interact and any product functionalities or environmental setting on which it relies must be provided. All related source code, scripts and configuration files must be disclosed. The information provided should be sufficient for an independent reconstruction of the implementation specific layer.

A command script was used to control and track the execution of queries. The scripts are contained in Appendix F. Qgen was used to generate the query streams, along with the appropriate substitution values.

The following steps are performed, to accomplish the Power and Throughput Runs:

1. Power Run
   - Execute sixteen concurrent RF1 processes, each of which will apply a segment of an update set generated by dbgen. Each process submits multiple transactions, where a transaction spans a set of orders and their associated line items.
   - Execute the Stream0 queries, in the prescribed order.
   - Execute sixteen concurrent RF2 processes, each of which will apply a segment of an update set generated by dbgen. Each thread submits multiple transactions, where a transaction spans a set of orders and their associated line items.
2. Throughput Run
   - Execute five concurrent query streams. Each stream executes queries in the prescribed order for the appropriate Stream Id (1-5). Upon completion of each stream, a semaphore is set to indication completion.
   - Execute five consecutive RF1/RF2 transactions, against ascending Update sets produced by dbgen. The first RF1 waits on a semaphore prior to beginning its insert operations.

Each step is timed by the script. The timing information is stored in the database for later analysis. The inputs and outputs of steps are stored in text files for later analysis.

7.3 Profile-Directed Optimization

If profile-directed optimization as described in Clause 5.2.x [5.2.9 and 5.2.10] is used, such use must be disclosed. In particular, the procedure and any scripts used to perform the optimization must be disclosed.

Profile-directed optimization subject to the requirements of *5.2.9* and *5.2.10* was not used.

8 Pricing Related Items

8.1 Hardware and Software Used in the Priced System

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 reported.

A detailed list of hardware and software used in the priced system is included in the pricing sheet in the executive summary. All prices are currently effective.

8.2 Total Five Year Price

The total 5-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.

A detailed pricing sheet of all the hardware and software used in this configuration and the 5-year maintenance costs, demonstrating the computation of the total 5-year price of the configuration, is included in the executive summary at the beginning of this document.

8.3 Availability Date

The committed delivery date for general availability (availability date) of products used in the priced calculations must be reported. When the priced system includes products with different availability dates, the single availability date reported on the first page of the executive summary must be the date by which all components are committed to being available. The full disclosure report must report availability dates individually for at least each of the categories for which a pricing subtotal must be provided (see Clause 7.3.1.3). All availability dates, whether for individual components or for the SUT as a whole, must be disclosed to a precision of 1 day, but the precise format is left to the test sponsor.

Availability dates are provided in the executive summary at the beginning of this report.

9 Audit Related Items

9.1 Auditor's Report

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.

This implementation of the TPC Benchmark H was audited by Francios Raab for InfoSizing Inc Further information regarding the audit process may be obtained from:

InfoSizing Inc
1373 North Franklin Street
CO Springs, CO 80903
Phone: 719-473-7555
Fax: 719-473-7554

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

Benchmark Sponsor: **Larry Kemp**
Hewlett-Packard Company
14335 NE 24th, Suite B-201
Bellevue, WA 98007

February 14, 2000

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

Platform:	**HP NetServer LXr 8500**
Database Manager:	**Microsoft SQL Server 2000**
Operating System:	**Microsoft Windows 2000**

The results were:

CPU (Speed)	Memory	Disks	**QphH@100GB**
HP NetServer LXr 8500			
8 x Pentium III Xeon (550 MHz)	2MB L2-Cache/cpu 4 GB Main	180 x 9 GB ext. 2 x 9 GB int.	**1,291.4**

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

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

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

- 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 and no query variant
- 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,

[illegible]

François Raab
President

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

Appendix A Windows/2000 and SqlServer Parameter Settings

System Information Report

```
System Information report written at: 02/14/2000
09:41:31 AM
[System Summary]

Item    Value
OS Name Microsoft Windows 2000 Advanced Server
Version 5.0.2195  Build 2195
OS Manufacturer       Microsoft Corporation
System Name    HPE2
System Manufacturer   HP
System Model   HP NetServer LXr 8500
System Type    X86-based PC
Processor      x86 Family 6 Model 7 Stepping 3
GenuineIntel ~550 Mhz
Processor      x86 Family 6 Model 7 Stepping 3
GenuineIntel ~550 Mhz
Processor      x86 Family 6 Model 7 Stepping 3
GenuineIntel ~550 Mhz
Processor      x86 Family 6 Model 7 Stepping 3
GenuineIntel ~550 Mhz
Processor      x86 Family 6 Model 7 Stepping 3
GenuineIntel ~550 Mhz
Processor      x86 Family 6 Model 7 Stepping 3
GenuineIntel ~550 Mhz
Processor      x86 Family 6 Model 7 Stepping 3
GenuineIntel ~550 Mhz
Processor      x86 Family 6 Model 7 Stepping 3
GenuineIntel ~550 Mhz
BIOS Version   OCPRF100- PhoenixBIOS 4.0 Release 6.0
Windows Directory     C:\WINNT
System Directory      C:\WINNT\System32
Boot Device    \Device\Harddisk0\Partition1
Locale  United States
User Name      HPE2\Administrator
Time Zone      Pacific Standard Time
Total Physical Memory 3,931,648 KB
Available Physical Memory     3,618,580 KB
Total Virtual Memory  14,086,196 KB
Available Virtual Memory      13,569,640 KB
Page File Space       10,154,548 KB
Page File      C:\pagefile.sys
Page File      D:\pagefile.sys
Page File      E:\pagefile.sys
Page File      F:\pagefile.sys
```

Registry Settings

```
Key Name:
SYSTEM\CurrentControlSet\Control\Session Manager\I/O
System
Class Name:        <NO CLASS>
Last Write Time:   2/8/2000 - 2:43 AM
Value 0
  Name:            CountOperations
  Type:            REG_DWORD
  Data:            0

Key Name:
SYSTEM\CurrentControlSet\Control\Session
Manager\Memory Management
Class Name:        <NO CLASS>
Last Write Time:   2/8/2000 - 2:42 AM
Value 0
  Name:            ClearPageFileAtShutdown
  Type:            REG_DWORD
  Data:            0

Value 1
  Name:            DisablePagingExecutive
  Type:            REG_DWORD
  Data:            0

Value 2
  Name:            IoPageLockLimit
  Type:            REG_DWORD
  Data:            0

Value 3
  Name:            LargeSystemCache
  Type:            REG_DWORD
  Data:            0

Value 4
  Name:            NonPagedPoolQuota
  Type:            REG_DWORD
  Data:            0

Value 5
  Name:            NonPagedPoolSize
  Type:            REG_DWORD
  Data:            0

Value 6
  Name:            PagedPoolQuota
  Type:            REG_DWORD
  Data:            0

Value 7
  Name:            PagedPoolSize
  Type:            REG_DWORD
  Data:            0

Value 8
  Name:            PagingFiles
  Type:            REG_MULTI_SZ
  Data:            C:\pagefile.sys 2046 4092
                   D:\pagefile.sys 512 1024
                   E:\pagefile.sys 2650 2650
                   F:\pagefile.sys 1024 2048


Value 9
  Name:            PhysicalAddressExtension
  Type:            REG_DWORD
  Data:            0

Value 10
  Name:            SecondLevelDataCache
  Type:            REG_DWORD
  Data:            0

Value 11
  Name:            SystemPages
  Type:            REG_DWORD
  Data:            0x100000
```

SqlServer Parameter Settings

```
 name                                run_value
----------------------------------- ----------
affinity mask                              255
allow updates                                1
cost threshold for parallelism               0
cursor threshold                            -1
default full-text language                1033
default language                             0
extended memory size (MB)                    0
fill factor (%)                              0
index create memory (KB)                     0
```

```
 language in cache                     3
 lightweight pooling                   1
 locks                                 0
 max degree of parallelism            0
 max server memory (MB)       2147483647
 max text repl size (B)             2048
 max worker threads                  255
 media retention                       0
 min memory per query (KB)           512
 min server memory (MB)                0
 nested triggers                       1
 network packet size (B)           32768
 open objects                          0
 priority boost                        0
 query governor cost limit             0
 query wait (s)               2147483647
 recovery interval (min)           32767
 remote access                         1
 remote login timeout (s)              5
 remote proc trans                     0
 remote query timeout (s)             0
 resource timeout (s)                 10
 scan for startup procs                0
 set working set size                  0
 show advanced options                 1
 spin counter                      10000
 time slice (ms)                     100
 two digit year cutoff              2049
 user connections                    512
 user options                          0
```

```
The SqlServer software was installed using the
collation name Latin1_General_BIN.
```

Appendix B Database, Table, Index Creation and Backup Scripts

CREATEDATABASE.SQL

```
--  CreateDatabase
--  Uses FileGroups

Create Database tpch100g on
Primary (name=tpch100g,
filename='d:\tpch\tpch100g.mdf',size=10mb),
FileGroup tpch100g

(name=tpch100g3,filename='d:\dev\tpch100g3\',size=29
990mb),

(name=tpch100g4,filename='d:\dev\tpch100g4\',size=29
990mb),

(name=tpch100g5,filename='d:\dev\tpch100g5\',size=29
990mb),

(name=tpch100g6,filename='d:\dev\tpch100g6\',size=29
990mb),

(name=tpch100g7,filename='d:\dev\tpch100g7\',size=29
990mb),

(name=tpch100g8,filename='d:\dev\tpch100g8\',size=29
990mb),

(name=tpch100g9,filename='d:\dev\tpch100g9\',size=29
990mb)
Log on
       (name=tpch100gLog,filename='d:\dev\tpch100gL
og\',size=9998mb)

Alter Database tpch100g Add FileGroup LoadFg

Alter Database tpch100g Add File
        (name=LoadFg3,
filename='d:\dev\loadfg3\',size=17130mb),
        (name=LoadFg4,
filename='d:\dev\loadfg4\',size=17130mb),
        (name=LoadFg5,
filename='d:\dev\loadfg5\',size=17130mb),
        (name=LoadFg6,
filename='d:\dev\loadfg6\',size=17130mb),
        (name=LoadFg7,
filename='d:\dev\loadfg7\',size=17130mb),
        (name=LoadFg8,
filename='d:\dev\loadfg8\',size=17130mb),
        (name=LoadFg9,
filename='d:\dev\loadfg9\',size=17130mb)
to FileGroup LoadFg
```

CREATETABLES.SQL

```
--  CreateTables
--  Uses filegroups

create table PART
       (P_PARTKEY     int            not null,
        P_NAME        varchar(55)    not null,
        P_MFGR        char(25)       not null,
        P_BRAND       char(10)       not null,
        P_TYPE        varchar(25)    not null,
        P_SIZE        int            not null,
        P_CONTAINER   char(10)       not null,
        P_RETAILPRICE money          not null,
        P_COMMENT     varchar(23)    not null)
    on LoadFg

create table SUPPLIER
       (S_SUPPKEY     int            not null,
        S_NAME        char(25)       not null,
        S_ADDRESS     varchar(40)    not null,
        S_NATIONKEY   int            not null,
        S_PHONE       char(15)       not null,
        S_ACCTBAL     money          not null,
        S_COMMENT     varchar(101)   not null)
    on LoadFg

create table PARTSUPP
       (PS_PARTKEY    int            not null,
        PS_SUPPKEY    int            not null,
        PS_AVAILQTY   int            not null,
        PS_SUPPLYCOST money          not null,
        PS_COMMENT    varchar(199)   not null)
    on LoadFg

create table CUSTOMER
       (C_CUSTKEY     int            not null,
        C_NAME        varchar(25)    not null,
        C_ADDRESS     varchar(40)    not null,
        C_NATIONKEY   int            not null,
        C_PHONE       char(15)       not null,
        C_ACCTBAL     money          not null,
        C_MKTSEGMENT  char(10)       not null,
        C_COMMENT     varchar(117)   not null)
    on LoadFg

create table ORDERS
       (O_ORDERKEY    int            not null,
        O_CUSTKEY     int            not null,
        O_ORDERSTATUS char(1)        not null,
        O_TOTALPRICE  money          not null,
        O_ORDERDATE   datetime       not null,
        O_ORDERPRIORITY char(15)     not null,
        O_CLERK       char(15)       not null,
        O_SHIPPRIORITY int           not null,
        O_COMMENT     varchar(79)    not null)
    on LoadFg

create table LINEITEM
       (L_ORDERKEY    int            not null,
        L_PARTKEY     int            not null,
        L_SUPPKEY     int            not null,
        L_LINENUMBER  int            not null,
        L_QUANTITY    money          not null,
        L_EXTENDEDPRICE money        not null,
        L_DISCOUNT    money          not null,
        L_TAX         money          not null,
        L_RETURNFLAG  char(1)        not null,
        L_LINESTATUS  char(1)        not null,
        L_SHIPDATE    datetime       not null,
        L_COMMITDATE  datetime       not null,
        L_RECEIPTDATE datetime       not null,
        L_SHIPINSTRUCT char(25)      not null,
        L_SHIPMODE    char(10)       not null,
        L_COMMENT     varchar(44)    not null)
    on LoadFg

create table NATION
       (N_NATIONKEY   int            not null,
        N_NAME        char(25)       not null,
        N_REGIONKEY   int            not null,
        N_COMMENT     varchar(152)   not null)
    on LoadFg
```

```
create table REGION
       (R_REGIONKEY    int            not null,
        R_NAME         char(25)       not null,
        R_COMMENT      varchar(152)  not null)
    on LoadFg
```

CreateLINEITEMIndexes.sql

```
create clustered index L_SHIPDATE_CLUIDX
    on      LINEITEM(L_SHIPDATE)
    with    fillfactor=95
    on      tpch100g

create index    L_PARTKEY_SUPPKEY_IDX
    on  LINEITEM (L_PARTKEY,L_SUPPKEY)
    with    fillfactor=95
    on      tpch100g

create index    L_ORDERKEY_IDX
    on  LINEITEM (L_ORDERKEY)
    with    fillfactor=95
    on      tpch100g
```

CreateORDERSIndexes.sql

```
create clustered index O_ORDERDATE_CLUIDX
    on ORDERS (O_ORDERDATE)
    with fillfactor=95
    on tpch100g

 create index O_CUSTKEY_IDX
    on ORDERS (O_CUSTKEY)
    with fillfactor=95
    on tpch100g

 create unique index O_KEY_IDX
    on ORDERS(O_ORDERKEY)
    with fillfactor=95
    on tpch100g
```

CreatePARTIndexes.sql

```
create unique clustered index P_KEY_CLUIDX
    on PART(P_PARTKEY)
    on tpch100g
```

CreateSUPPLIERIndexes.sql

```
  create unique clustered index S_SUPPKEY_CLUIDX
    on SUPPLIER (S_SUPPKEY)
    on tpch100g
  create index S_NATION_KEYIDX
    on SUPPLIER (S_NATIONKEY)
    on tpch100g
```

CreatePARTSUPPIndexes.sql

```
create unique clustered index PS_KEY_CLUIDX
    on PARTSUPP(PS_PARTKEY,PS_SUPPKEY)
    on tpch100g
 create index PS_SUPPKEY_IDX
    on PARTSUPP (PS_SUPPKEY)
    on tpch100g
```

CreateCUSTOMERIndexes.sql

```
create unique clustered index C_KEY_CLUIDX
    on CUSTOMER(C_CUSTKEY)
    on tpch100g
  create index C_NATION_KEYIDX
    on CUSTOMER (C_NATIONKEY)
    on tpch100g
```

CreateNATIONIndexes.sql

```
create unique clustered index N_KEY_CLUIDX
    on NATION(N_NATIONKEY)
    on tpch100g
 create index N_REGIONKEY_IDX
    on NATION (N_REGIONKEY)
    on tpch100g
```

CreateREGIONIndexes.sql

```
create unique clustered index R_KEY_CLUIDX
    on REGION(R_REGIONKEY)
    on tpch100g
```

Backup.sql

```
backup database tpch100g to
disk='d:\dev\filesys4\tpch100gSetA.bak',
disk='d:\dev\filesys6\tpch100gSetA.bak',
disk='d:\dev\filesys8\tpch100gSetA.bak'
with init,stats=10

backup database tpch100g to
disk='d:\dev\filesys5\tpch100gSetB.bak',
disk='d:\dev\filesys7\tpch100gSetB.bak',
disk='d:\dev\filesys9\tpch100gSetB.bak'
with init,stats=10
```

Appendix C Disk and Volume Partitioning Specifications

	Drive Type	Windows/2000 Device Name	Size in MB
39160 SCSI 2xRS/12 24x9gb	NT Dynamic Stripe	d:\dev\tpch100g3	31560
	NT Dynamic Stripe	d:\dev\temp3	21480
	NT Dynamic Stripe	d:\dev\loadfg3	17280
	NT Dynamic Stripe	d:\dev\filesys3	136080
39160 SCSI 2xRS/12 24x9gb	NT Dynamic Stripe	d:\dev\tpch100g4	31560
	NT Dynamic Stripe	d:\dev\temp4	21480
	NT Dynamic Stripe	d:\dev\loadfg4	17280
	NT Dynamic Stripe	d:\dev\filesys4	136080
39160 SCSI 2xRS/12 24x9gb	NT Dynamic Stripe	d:\dev\tpch100g5	31560
	NT Dynamic Stripe	d:\dev\temp5	21480
	NT Dynamic Stripe	d:\dev\loadfg5	17280
	NT Dynamic Stripe	d:\dev\filesys5	136080
39160 SCSI 2xRS/12 24x9gb	NT Dynamic Stripe	d:\dev\tpch100g6	31560
	NT Dynamic Stripe	d:\dev\temp6	21480
	NT Dynamic Stripe	d:\dev\loadfg6	17280
	NT Dynamic Stripe	d:\dev\filesys6	136080
39160 SCSI 2xRS/12 24x9gb	NT Dynamic Stripe	d:\dev\tpch100g7	31560
	NT Dynamic Stripe	d:\dev\temp7	21480
	NT Dynamic Stripe	d:\dev\loadfg7	17280
	NT Dynamic Stripe	d:\dev\filesys7	136080
39160 SCSI 2xRS/12 24x9gb	NT Dynamic Stripe	d:\dev\tpch100g8	31560
	NT Dynamic Stripe	d:\dev\temp8	21480
	NT Dynamic Stripe	d:\dev\loadfg8	17280
	NT Dynamic Stripe	d:\dev\filesys8	136080
39160 SCSI 2xRS/12 24x9gb	NT Dynamic Stripe	d:\dev\tpch100g9	31560
	NT Dynamic Stripe	d:\dev\temp9	21480
	NT Dynamic Stripe	d:\dev\loadfg9	17280
	NT Dynamic Stripe	d:\dev\filesys9	136080
HP FCArray 1xRS/12 12x9gb	RAID0+1	d:\dev\tpch100gLog	10000
	RAID0+1	d:\dev\tempLog	16000
	RAID0+1	d:\dev\updateFiles	23000

Appendix D Validation Query Text and Output

```
Qualification Queries and Answers
-- using default substitutions

/* tpch 1.sql */
print 'BEGIN Q01'

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(dd, -90, '1998/12/01')
GROUP BY
L_RETURNFLAG
,L_LINESTATUS
ORDER BY
L_RETURNFLAG
,L_LINESTATUS
go
1> 2> 3> 4> 5> 6> BEGIN Q01
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20> 21> 22> 23>  L_RETURNFLAG
L_LINESTATUS SUM_QTY                SUM_BASE_PRICE
       SUM_DISC_PRICE         SUM_CHARGE
AVG_QTY
       AVG_PRICE              AVG_DISC
COUNT_ORDER
 ------------ ------------ ---------------------
---------------------
       --------------------- --------------------
-- ---------------------
       --------------------- --------------------
-- -----------
 A            F                    $37734107.0000
$56586554400.7300
            $53758257134.8700
$55909065224.7041               $25.5220
                  $38273.1297
$.0499     1478493
 N            F                      $991417.0000
$1487504710.3800
             $1413082168.0541
$1469649223.2395              $25.5164
                  $38284.4677
$.0500       38854
 N            O                    $74476040.0000
$111701729697.7400
           $106118230307.6056
$110367043876.2372                $25.5022
                  $38249.1179
$.0499     2920374
 R            F                    $37719753.0000
$56568041380.9000
            $53741292684.6040
$55889619121.7027               $25.5057
                  $38250.8546
$.0500     1478870

(4 rows affected)
1>
-- using default substitutions
```

```
/* tpch 2.sql */
print 'BEGIN Q2'

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
go
1> 2> 3> 4> 5> 6> BEGIN Q2
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20> 21> 22> 23> 24> 25> 26> 27>
28> 29> 30> 31> 32> 33> 34> 35> 36> 37> 38> 39>
40> 41> 42> 43> 44>  S_ACCTBAL              S_NAME
N_NAME
       P_PARTKEY   P_MFGR
       S_ADDRESS
S_PHONE
       S_COMMENT

 --------------------- ------------------------
------------------------
       ----------- ------------------------
       ---------------------------------------- --
-------------
       ------------------------------------------
----------------------------
       ------------------------------
            $9938.5300 Supplier#000005359
UNITED KINGDOM
            185358 Manufacturer#4
       QKuHYh,vZGiwu2FWEJoLDx04
33-429-790-6131
       blithely silent pinto beans are furiously.
slyly final deposits acros

            $9937.8400 Supplier#000005969
ROMANIA
            108438 Manufacturer#1
       ANDENSOSmk,miq23Xfb5RWt6dvUcvt6Qa
29-520-692-3537
       carefully slow deposits use furiously.
slyly ironic platelets above the
'-----                -----'
'-----lines surpressed-----'
'-----                -----'
       furiously dogged pinto beans cajole. bold,
express notornis until the s
       lyly pending
            $7852.4500 Supplier#000005864
RUSSIA
              8363 Manufacturer#4
       WCNfBPZeSXh3h,c
32-454-883-3821
       blithely regular deposits

            $7850.6600 Supplier#000001518
UNITED KINGDOM
             86501 Manufacturer#1
       ONda3YJiHKJOC
33-730-383-3892
       furiously final accounts wake carefully
idle requests. even dolphins wa
       ke acc
            $7843.5200 Supplier#000006683
FRANCE
             11680 Manufacturer#4
       2Z0JGkiv01Y00oCFwUGfviIbhzCdy
16-464-517-8943
       carefully bold accounts doub


(100 rows affected)
1>
-- using default substitutions

/* tpch 3.sql */
print 'BEGIN Q3'

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
go
1> 2> 3> 4> 5> 6> BEGIN Q3
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20> 21> 22> 23> 24>  L_ORDERKEY
REVENUE               O_ORDERDATE
O_SHIPPRIORITY
 ----------- -------------------- -------------
--------- -------------
     2456423           $406181.0111 1995-03-05
00:00:00.000              0
     3459808           $405838.6989 1995-03-04
00:00:00.000              0
      492164           $390324.0610 1995-02-19
00:00:00.000              0
     1188320           $384537.9359 1995-03-09
00:00:00.000              0
     2435712           $378673.0558 1995-02-26
00:00:00.000              0
     4878020           $378376.7952 1995-03-12
00:00:00.000              0
     5521732           $375153.9215 1995-03-13
00:00:00.000              0
     2628192           $373133.3094 1995-02-22
00:00:00.000              0
      993600           $371407.4595 1995-03-05
00:00:00.000              0
     2300070           $367371.1452 1995-03-13
00:00:00.000              0

(10 rows affected)
1>
-- using default substitutions

/* tpch 4.sql */
print 'BEGIN Q4'

SELECT
O_ORDERPRIORITY
,COUNT(*) AS ORDER_COUNT
FROM
ORDERS
WHERE O_ORDERDATE >= '1993-07-01'
AND O_ORDERDATE < DATEADD (mm, 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
go
1> 2> 3> 4> 5> 6> BEGIN Q4
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20>  O_ORDERPRIORITY ORDER_COUNT
 --------------- -----------
 1-URGENT              10594
 2-HIGH                10476
 3-MEDIUM              10410
 4-NOT SPECIFIED       10556
 5-LOW                 10487

(5 rows affected)
1>
-- using default substitutions

/* tpch 5.sql */
print 'BEGIN Q5'

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(YY, 1, '1994-01-01')
GROUP BY
N_NAME
ORDER BY
REVENUE DESC
go
1> 2> 3> 4> 5> 6> BEGIN Q5
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20> 21> 22> 23> 24> 25> 26>
N_NAME                    REVENUE
 ------------------------- ---------------------
 INDONESIA                          $55502041.1697
 VIETNAM                            $55295086.9967
 CHINA                              $53724494.2566
 INDIA                              $52035512.0002
 JAPAN                              $45410175.6954

(5 rows affected)
1>
-- using default substitutions

/* tpch 6.sql */
print 'BEGIN Q6'

SELECT
SUM(L_EXTENDEDPRICE*L_DISCOUNT) AS REVENUE
FROM
LINEITEM
WHERE
L_SHIPDATE >= '1994-01-01'
AND L_SHIPDATE < dateadd (yy, 1, '1994-01-01')
AND L_DISCOUNT BETWEEN .06 - 0.01 AND .06 + 0.01
AND L_QUANTITY < 24
go
1> 2> 3> 4> 5> 6> BEGIN Q6
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11>  REVENUE
 ---------------------
        $123141078.2283

(1 row affected)
1>
-- using default substitutions

/* tpch 7.sql */
print 'BEGIN Q7'

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(YY,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
go
1> 2> 3> 4> 5> 6> BEGIN Q7
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20> 21> 22> 23> 24> 25> 26> 27>
28> 29> 30> 31> 32> 33> 34> 35> 36> 37> 38> 39>
40> 41> 42>  SUPP_NATION               CUST_NATION
L_YEAR
       REVENUE
 ------------------------- ----------------------
-- -----------
       ---------------------
 FRANCE                    GERMANY
1995
               $54639732.7336
 FRANCE                    GERMANY
1996
               $54633083.3076
 GERMANY                   FRANCE
1995
               $52531746.6697
 GERMANY                   FRANCE
1996
               $52520549.0224

(4 rows affected)
1>
-- using default substitutions

/* tpch 8.sql */
print 'BEGIN Q8'

SELECT
O_YEAR
,SUM(CASE WHEN NATION = 'BRAZIL'
THEN VOLUME
ELSE 0
END) / SUM(VOLUME) AS MKT_SHARE
FROM
(
SELECT
DATEPART(YY,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
go
1> 2> 3> 4> 5> 6> BEGIN Q8
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20> 21> 22> 23> 24> 25> 26> 27>
28> 29> 30> 31> 32> 33> 34> 35> 36> 37> 38> 39>
O_YEAR      MKT_SHARE
 ----------- ---------------------
        1995                $.0344
        1996                $.0414

(2 rows affected)
1>
-- using default substitutions

/* tpch 9.sql */
print 'BEGIN Q9'

SELECT
NATION
,O_YEAR
,SUM(AMOUNT) AS SUM_PROFIT
FROM
(
SELECT
N_NAME AS NATION
,DATEPART(YY, 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
go
1> 2> 3> 4> 5> 6> BEGIN Q9
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20> 21> 22> 23> 24> 25> 26> 27>
28> 29> 30> 31> 32> 33> 34> 35>  NATION
O_YEAR      SUM_PROFIT
 ------------------------- ----------- -----------
-----------
 ALGERIA                          1998
$31342867.2345
 ALGERIA                          1997
$57138193.0233
 ALGERIA                          1996
$56140140.1330
 ALGERIA                          1995
$53051469.6534
 ALGERIA                          1994
$53867582.1286
 ALGERIA                          1993
$54942718.1324
 ALGERIA                          1992
$54628034.7127
 ARGENTINA                        1998
$30211185.7081
 ARGENTINA                        1997
$50805741.7523
 ARGENTINA                        1996
$51923746.5755
 ARGENTINA                        1995
$49298625.7666
 ARGENTINA                        1994
$50835610.1095
 ARGENTINA                        1993
$51646079.1775
 ARGENTINA                        1992
$50410314.9948
 BRAZIL                           1998
$27217924.3832
 BRAZIL                           1997
$48378669.1989
 BRAZIL                           1996
$50482870.3572
'-----                -----'
'-----lines surpressed-----'
'-----                -----'
 UNITED KINGDOM                   1994
$48086499.7115
 UNITED KINGDOM                   1993
$49166827.2235
 UNITED KINGDOM                   1992
$49349122.0825
 UNITED STATES                    1998
$25126238.9461
 UNITED STATES                    1997
$50077306.4186
 UNITED STATES                    1996
$48048649.4703
 UNITED STATES                    1995
$48809032.4226
 UNITED STATES                    1994
$49296747.1827
 UNITED STATES                    1993
$48029946.8014
 UNITED STATES                    1992
$48671944.4983
 VIETNAM                          1998
$30442736.0594
 VIETNAM                          1997
$50309179.7942
 VIETNAM                          1996
$50488161.4100
 VIETNAM                          1995
$49658284.6125
 VIETNAM                          1994
$50596057.2607
 VIETNAM                          1993
$50953919.1519
 VIETNAM                          1992
$49613838.3151

(175 rows affected)
1>
-- using default substitutions

/* tpch 10.sql */
print 'BEGIN Q10'

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(MM, 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
go
1> 2> 3> 4> 5> 6> BEGIN Q10
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20> 21> 22> 23> 24> 25> 26> 27>
28> 29> 30> 31> 32> 33>  C_CUSTKEY   C_NAME
REVENUE
       C_ACCTBAL              N_NAME
       C_ADDRESS
C_PHONE
       C_COMMENT

 ----------- ------------------------ -----------
-----------
       --------------------- -------------------
-----
       ---------------------------------------- --
-------------
       -----------------------------------------
----------------------------
       -----------------------------------------
---
       57040 Customer#000057040
$734235.2455
                    $632.8700 JAPAN
       Eioyzjf4pp
22-895-641-3466
       requests sleep blithely about the furiously
i

      143347 Customer#000143347
$721002.6948
                   $2557.4700 EGYPT
       1aReFYv,Kw4
14-742-935-3718
       fluffily bold excuses haggle finally after
the u

       60838 Customer#000060838
$679127.3077
                   $2454.7700 BRAZIL
       64EaJ5vMAHWJlBOxJklpNc2RJiWE
12-913-494-9813
       furiously even pinto beans integrate under
the ruthless foxes; ironic,
       even dolphins across the slyl
      101998 Customer#000101998
$637029.5667
                   $3790.8900 UNITED KINGDOM
       01c9CILnNtfOQYmZj
33-593-865-6378
       accounts doze blithely! enticing, final
deposits sleep blithely special
        accounts. slyly express accounts pla
      125341 Customer#000125341
$633508.0860
                   $4983.5100 GERMANY
       S29ODD6bceU8QSuuEJznkNaK
17-582-695-5962
       quickly express requests wake quickly
blithely
       25501 Customer#000025501
$620269.7849
                   $7725.0400 ETHIOPIA
         W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ
15-874-808-6793
       quickly special requests sleep evenly among
the special deposits. speci
       al deposi
      115831 Customer#000115831
$596423.8672
                   $5098.1000 FRANCE
       rFeBbEEyk dl ne7zV5fDrmiq1oK09wV7pxqCgIc
16-715-386-3788
       carefully bold excuses sleep alongside of
the thinly idle

       84223 Customer#000084223
$594998.0239
                    $528.6500 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
$585603.3918
                   $5583.0200 IRAN
       vXCxoCsU0Bad5JQI ,oobkZ
20-834-292-4707
       express requests sublate blithely regular
requests. regular, even ideas
        solve.
       39922 Customer#000039922
$584878.1134
                   $7321.1100 GERMANY
       Zgy4s50l2GKN4pLDPBU8m342gIw6R
17-147-757-8036
       even pinto beans haggle. slyly bold
accounts inte

        6226 Customer#000006226
$576783.7606
                   $2230.0900 UNITED KINGDOM
       8gPu8,NPGkfyQQ0hcIYUGPIBWc,ybP5g,
33-657-701-3391
       quickly final requests against the regular
instructions wake blithely f
       inal instructions. pa
         922 Customer#000000922
$576767.5333
                   $3869.2500 GERMANY
       Az9RFaut7NkPnc5zSD2PwHgVwr4jRzq
17-945-916-9648
       boldly final requests cajole blith

      147946 Customer#000147946
$576455.1320
                   $2030.1300 ALGERIA
       iANyZHjqhyy7Ajah0pTrYyhJ
10-886-956-3143
       furiously even accounts are blithely above
the furiousl

      115640 Customer#000115640
$569341.1933
                   $6436.1000 ARGENTINA
       Vtgfia9qI 7EpHgecU1X
11-411-543-4901
       final instructions are slyly according to
the

       73606 Customer#000073606
$568656.8578
                   $1785.6700 JAPAN
       xuR0Tro5yChDfOCrjkd2ol
22-437-653-6966
```

```
       furiously bold orbits about the furiously
busy requests wake across the
        furiously quiet theodolites. d
      110246 Customer#000110246
$566842.9815
                   $7763.3500 VIETNAM
       7KzflgX MDOq7sOkI
31-943-426-9837
       dolphins sleep blithely among the slyly
final

      142549 Customer#000142549
$563537.2368
                   $5085.9900 INDONESIA
       ChqEoK43OysjdHbtKCp6dKqjNyvvi9
19-955-562-2398
       regular, unusual dependencies boost slyly;
ironic attainments nag fluff
       ily into the unusual packages?
      146149 Customer#000146149
$557254.9865
                   $1791.5500 ROMANIA
       s87fvzFQpU
29-744-164-6487
       silent, unusual requests detect quickly
slyly regul

       52528 Customer#000052528
$556397.3509
                    $551.7900 ARGENTINA
       NFztyTOR10UOJ
11-208-192-3205
       unusual requests detect. slyly dogged
theodolites use slyly. deposit

       23431 Customer#000023431
$554269.5360
                   $3381.8600 ROMANIA
       HgiV0phqhaIa9aydNoIlb
29-915-458-2654
       instructions nag quickly. furiously bold
accounts cajol


(20 rows affected)
1>
-- using default substitutions

/* tpch 11.sql */
print 'BEGIN Q11'

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
go
1> 2> 3> 4> 5> 6> BEGIN Q11
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20> 21> 22> 23> 24> 25> 26> 27>
28> 29>  PS_PARTKEY  VALUE
 ----------- ---------------------
      129760        $17538456.8600
      166726        $16503353.9200
      191287        $16474801.9700
      161758        $16101755.5400
       34452        $15983844.7200
      139035        $15907078.3400
        9403        $15451755.6200
      154358        $15212937.8800
       38823        $15064802.8600
       85606        $15053957.1500
       33354        $14408297.4000
      154747        $14407580.6800
       82865        $14235489.7800
       76094        $14094247.0400
         222        $13937777.7400
      121271        $13908336.0000
       55221        $13716120.4700
'-----                -----'
'-----lines surpressed-----'
'-----                -----'
      194299         $7898421.2400
      105235         $7897829.9400
       77207         $7897752.7200
       96712         $7897575.2700
       10157         $7897046.2500
      171154         $7896814.5000
       79373         $7896186.0000
      113808         $7893353.8800
       27901         $7892952.0000
      128820         $7892882.7200
       25891         $7890511.2000
      122819         $7888881.0200
      154731         $7888301.3300
      101674         $7879324.6000
       51968         $7879102.2100
       72073         $7877736.1100
        5182         $7874521.7300

(1048 rows affected)
1>
-- using default substitutions

/* tpch 12.sql */
print 'BEGIN Q12'

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(YY, 1, '1994-01-01')
GROUP BY
```

```
L_SHIPMODE
ORDER BY
L_SHIPMODE
go
1> 2> 3> 4> 5> 6> BEGIN Q12
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20> 21> 22> 23> 24> 25> 26> 27>
28> 29> 30>  L_SHIPMODE HIGH_LINE_COUNT
LOW_LINE_COUNT
 ---------- --------------- --------------
 MAIL                  6202           9324
 SHIP                  6200           9262

(2 rows affected)
1>
-- using default substitutions

/* tpch 13.sql */
print 'BEGIN Q13'

SELECT
C_COUNT
,COUNT(*) AS CUSTDIST
FROM
(
SELECT
C_CUSTKEY
,COUNT(O_ORDERKEY)
FROM
CUSTOMER LEFT OUTER JOIN ORDERS ON
C_CUSTKEY = O_CUSTKEY
AND O_COMMENT NOT LIKE '%special%requests%'
GROUP BY
C_CUSTKEY
) AS C_ORDERS (C_CUSTKEY, C_COUNT)
GROUP BY
C_COUNT
ORDER BY
CUSTDIST DESC
,C_COUNT DESC
go
1> 2> 3> 4> 5> 6> BEGIN Q13
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20> 21> 22>  C_COUNT     CUSTDIST
 ----------- -----------
           0       50004
           9        6641
          10        6566
          11        6058
           8        5949
          12        5553
          13        4989
          19        4748
           7        4707
          18        4625
          15        4552
          17        4530
          14        4484
          20        4461
          16        4323
          21        4217
          22        3730
           6        3334
          23        3129
          24        2622
          25        2079
           5        1972
          26        1593
          27        1185
           4        1033
          28         869
          29         559
           3         398
          30         373
          31         235
           2         144
          32         128
          33          71
          34          48
          35          33
           1          23
          36          17
          37           7
          40           4
          38           4
          39           2
          41           1
Warning: Null value eliminated from aggregate.

(42 rows affected)
1>
-- using default substitutions

/* tpch 14.sql */
print 'BEGIN Q14'

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(MM, 1, '1995-09-01')
go
1> 2> 3> 4> 5> 6> BEGIN Q14
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14>
PROMO_REVENUE
 ----------------------------------------
                       16.380778626395540

(1 row affected)
1>
-- using default substitutions

/* tpch 15.sql */
print 'BEGIN Q15'

if exists (select * from sysindexes where name =
'REVENUE0')
  drop view REVENUE0
go

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(MM, 3, '1996-01-01')
GROUP BY
L_SUPPKEY
go

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

DROP VIEW REVENUE0
go
1> 2> 3> 4> 5> 6> BEGIN Q15
1> 2> 3> 4> 1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12>
13> 1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14>
15> 16> 17> 18> 19> 20> 21> 22> 23>  S_SUPPKEY
S_NAME                    S_ADDRESS
       S_PHONE         TOTAL_REVENUE
 ----------- ------------------------ -----------
----------------------------
       --------------- ---------------------
        8449 Supplier#000008449
Wp34zim9qYFbVctdW
       20-469-856-8873          $1772627.2087

(1 row affected)
1>
-- using default substitutions

/* tpch 16.sql */
print 'BEGIN Q16'

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 '%Customer%Complaints%'
)
GROUP BY
P_BRAND
,P_TYPE
,P_SIZE
ORDER BY
SUPPLIER_CNT DESC
,P_BRAND
,P_TYPE
,P_SIZE
go
1> 2> 3> 4> 5> 6> BEGIN Q16
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20> 21> 22> 23> 24> 25> 26> 27>
28> 29>  P_BRAND    P_TYPE
P_SIZE      SUPPLIER_CNT
 ---------- ------------------------ -----------
------------
 Brand#41   MEDIUM BRUSHED TIN                  3
28
 Brand#54   STANDARD BRUSHED COPPER            14
27
 Brand#11   STANDARD BRUSHED TIN               23
24
 Brand#11   STANDARD BURNISHED BRASS           36
24
 Brand#15   MEDIUM ANODIZED NICKEL              3
24
 Brand#15   SMALL ANODIZED BRASS                 45
24
 Brand#15   SMALL BURNISHED NICKEL              19
24
 Brand#21   MEDIUM ANODIZED COPPER               3
24
 Brand#22   SMALL BRUSHED NICKEL                 3
24
 Brand#22   SMALL BURNISHED BRASS               19
24
 Brand#25   MEDIUM BURNISHED COPPER             36
24
 Brand#31   PROMO POLISHED COPPER               36
24
 Brand#33   LARGE POLISHED TIN                  23
24
 Brand#33   PROMO POLISHED STEEL                14
24
 Brand#35   PROMO BRUSHED NICKEL                14
24
 Brand#41   ECONOMY BRUSHED STEEL                9
24
 Brand#41   ECONOMY POLISHED TIN                19
24
'-----                -----'
'-----lines surpressed-----'
'-----                -----'
 Brand#55   STANDARD POLISHED TIN               19
4
 Brand#55   STANDARD POLISHED TIN               36
4
 Brand#11   SMALL BRUSHED TIN                   19
3
 Brand#15   LARGE PLATED NICKEL                 45
3
 Brand#15   LARGE POLISHED NICKEL                9
3
 Brand#21   PROMO BURNISHED STEEL               45
3
 Brand#22   STANDARD PLATED STEEL               23
3
 Brand#25   LARGE PLATED STEEL                  19
3
 Brand#32   STANDARD ANODIZED COPPER            23
3
 Brand#33   SMALL ANODIZED BRASS                 9
3
 Brand#35   MEDIUM ANODIZED TIN                 19
3
 Brand#51   SMALL PLATED BRASS                  23
3
 Brand#52   MEDIUM BRUSHED BRASS                45
3
 Brand#53   MEDIUM BRUSHED TIN                  45
3
 Brand#54   ECONOMY POLISHED BRASS               9
3
 Brand#55   PROMO PLATED BRASS                  19
3
 Brand#55   STANDARD PLATED TIN                 49
3

(18314 rows affected)
1>
-- using default substitutions

/* tpch 17.sql */
print 'BEGIN Q17'

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
)
go
1> 2> 3> 4> 5> 6> BEGIN Q17
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18>  AVG_YEARLY
 -------------------------
            348406.0542857

(1 row affected)
1>
-- using default substitutions

/* tpch 18.sql */
print 'BEGIN Q18'

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
go
1> 2> 3> 4> 5> 6> BEGIN Q18
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20> 21> 22> 23> 24> 25> 26> 27>
28> 29> 30> 31> 32> 33> 34> 35>  C_NAME
C_CUSTKEY   O_ORDERKEY  O_ORDERDATE
       O_TOTALPRICE
 ------------------------- ----------- -----------
-----------------------
       --------------------- --------------------
--
 Customer#000128120            128120     4722021
1994-04-07 00:00:00.000
                 $544089.0900
$323.0000
 Customer#000144617            144617     3043270
1997-02-12 00:00:00.000
                 $530604.4400
$317.0000
 Customer#000013940             13940     2232932
1997-04-13 00:00:00.000
                 $522720.6100
$304.0000
 Customer#000066790             66790     2199712
1996-09-30 00:00:00.000
                 $515531.8200
$327.0000
 Customer#000046435             46435     4745607
1997-07-03 00:00:00.000
                 $508047.9900
$309.0000
 Customer#000015272             15272     3883783
1993-07-28 00:00:00.000
                 $500241.3300
$302.0000
 Customer#000146608            146608     3342468
1994-06-12 00:00:00.000
                 $499794.5800
$303.0000
 Customer#000096103             96103     5984582
1992-03-16 00:00:00.000
                 $494398.7900
$312.0000
 Customer#000024341             24341     1474818
1992-11-15 00:00:00.000
                 $491348.2600
$302.0000
 Customer#000137446            137446     5489475
1997-05-23 00:00:00.000
                 $487763.2500
$311.0000
 Customer#000107590            107590     4267751
1994-11-04 00:00:00.000
                 $485141.3800
$301.0000
 Customer#000050008             50008     2366755
1996-12-09 00:00:00.000
                 $483891.2600
$302.0000
 Customer#000015619             15619     3767271
1996-08-07 00:00:00.000
                 $480083.9600
$318.0000
 Customer#000077260             77260     1436544
1992-09-12 00:00:00.000
                 $479499.4300
$307.0000
 Customer#000109379            109379     5746311
1996-10-10 00:00:00.000
                 $478064.1100
$302.0000
 Customer#000054602             54602     5832321
1997-02-09 00:00:00.000
                 $471220.0800
$307.0000
 Customer#000105995            105995     2096705
1994-07-03 00:00:00.000
                 $469692.5800
$307.0000
 Customer#000148885            148885     2942469
1992-05-31 00:00:00.000
                 $469630.4400
$313.0000
 Customer#000114586            114586      551136
1993-05-19 00:00:00.000
                 $469605.5900
$308.0000
 Customer#000105260            105260     5296167
1996-09-06 00:00:00.000
                 $469360.5700
$303.0000
 Customer#000147197            147197     1263015
1997-02-02 00:00:00.000
                 $467149.6700
$320.0000
 Customer#000064483             64483     2745894
1996-07-04 00:00:00.000
```

```
                 $466991.3500
$304.0000
 Customer#000136573            136573     2761378
1996-05-31 00:00:00.000
                 $461282.7300
$301.0000
 Customer#000016384             16384      502886
1994-04-12 00:00:00.000
                 $458378.9200
$312.0000
 Customer#000117919            117919     2869152
1996-06-20 00:00:00.000
                 $456815.9200
$317.0000
 Customer#000012251             12251      735366
1993-11-24 00:00:00.000
                 $455107.2600
$309.0000
 Customer#000120098            120098     1971680
1995-06-14 00:00:00.000
                 $453451.2300
$308.0000
 Customer#000066098             66098     5007490
1992-08-07 00:00:00.000
                 $453436.1600
$304.0000
 Customer#000117076            117076     4290656
1997-02-05 00:00:00.000
                 $449545.8500
$301.0000
 Customer#000129379            129379     4720454
1997-06-07 00:00:00.000
                 $448665.7900
$303.0000
 Customer#000126865            126865     4702759
1994-11-07 00:00:00.000
                 $447606.6500
$320.0000
 Customer#000088876             88876      983201
1993-12-30 00:00:00.000
                 $446717.4600
$304.0000
 Customer#000036619             36619     4806726
1995-01-17 00:00:00.000
                 $446704.0900
$328.0000
 Customer#000141823            141823     2806245
1996-12-29 00:00:00.000
                 $446269.1200
$310.0000
 Customer#000053029             53029     2662214
1993-08-13 00:00:00.000
                 $446144.4900
$302.0000
 Customer#000018188             18188     3037414
1995-01-25 00:00:00.000
                 $443807.2200
$308.0000
 Customer#000066533             66533       29158
1995-10-21 00:00:00.000
                 $443576.5000
$305.0000
 Customer#000037729             37729     4134341
1995-06-29 00:00:00.000
                 $441082.9700
$309.0000
 Customer#000003566              3566     2329187
1998-01-04 00:00:00.000
                 $439803.3600
$304.0000
 Customer#000045538             45538     4527553
1994-05-22 00:00:00.000
                 $436275.3100
$305.0000
 Customer#000081581             81581     4739650
1995-11-04 00:00:00.000
                 $435405.9000
$305.0000
 Customer#000119989            119989     1544643
1997-09-20 00:00:00.000
                 $434568.2500
$320.0000
 Customer#000003680              3680     3861123
1998-07-03 00:00:00.000
                 $433525.9700
$301.0000
 Customer#000113131            113131      967334
1995-12-15 00:00:00.000
                 $432957.7500
$301.0000
 Customer#000141098            141098      565574
1995-09-24 00:00:00.000
                 $430986.6900
$301.0000
 Customer#000093392             93392     5200102
1997-01-22 00:00:00.000
                 $425487.5100
$304.0000
 Customer#000015631             15631     1845057
1994-05-12 00:00:00.000
                 $419879.5900
$302.0000
 Customer#000112987            112987     4439686
1996-09-17 00:00:00.000
                 $418161.4900
$305.0000
 Customer#000012599             12599     4259524
1998-02-12 00:00:00.000
                 $415200.6100
$304.0000
 Customer#000105410            105410     4478371
1996-03-05 00:00:00.000
                 $412754.5100
$302.0000
 Customer#000149842            149842     5156581
1994-05-30 00:00:00.000
                 $411329.3500
$302.0000
 Customer#000010129             10129     5849444
1994-03-21 00:00:00.000
                 $409129.8500
$309.0000
 Customer#000069904             69904     1742403
1996-10-19 00:00:00.000
                 $408513.0000
$305.0000
 Customer#000017746             17746        6882
1997-04-09 00:00:00.000
                 $408446.9300
$303.0000
 Customer#000013072             13072     1481925
1998-03-15 00:00:00.000
                 $399195.4700
$301.0000
 Customer#000082441             82441      857959
1994-02-07 00:00:00.000
                 $382579.7400
$305.0000
 Customer#000088703             88703     2995076
1994-01-30 00:00:00.000
                 $363812.1200
$302.0000

(57 rows affected)
1>
-- using default substitutions

/* tpch 19.sql */
print 'BEGIN Q19'

SELECT
SUM(L_EXTENDEDPRICE* (1 - L_DISCOUNT)) AS REVENUE
FROM
```

```
LINEITEM
,PART
WHERE
(
P_PARTKEY = L_PARTKEY
AND P_BRAND = 'Brand#12'
AND P_CONTAINER IN ('SM CASE', 'SM BOX', 'SM
PACK', 'SM PKG')
AND L_QUANTITY >= 1 AND L_QUANTITY <= 1 + 10
AND P_SIZE BETWEEN 1 AND 5
AND L_SHIPMODE IN ('AIR', 'AIR REG')
AND L_SHIPINSTRUCT = 'DELIVER IN PERSON'
)
OR
(
P_PARTKEY = L_PARTKEY
AND P_BRAND = 'Brand#23'
AND P_CONTAINER IN ('MED BAG', 'MED BOX', 'MED
PKG', 'MED PACK')
AND L_QUANTITY >= 10 AND L_QUANTITY <= 10 + 10
AND P_SIZE BETWEEN 1 AND 10
AND L_SHIPMODE IN ('AIR', 'AIR REG')
AND L_SHIPINSTRUCT = 'DELIVER IN PERSON'
)
OR
(
P_PARTKEY = L_PARTKEY
AND P_BRAND = 'Brand#34'
AND P_CONTAINER IN ('LG CASE', 'LG BOX', 'LG
PACK', 'LG PKG')
AND L_QUANTITY >= 20 AND L_QUANTITY <= 20 + 10
AND P_SIZE BETWEEN 1 AND 15
AND L_SHIPMODE IN ('AIR', 'AIR REG')
AND L_SHIPINSTRUCT = 'DELIVER IN PERSON'
)
go
1> 2> 3> 4> 5> 6> BEGIN Q19
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20> 21> 22> 23> 24> 25> 26> 27>
28> 29> 30> 31> 32> 33> 34> 35> 36> 37>  REVENUE
 ----------------------
          $3083843.0578

(1 row affected)
1>
-- using default substitutions

/* tpch 20.sql */
print 'BEGIN Q20'

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(YY,1,'1994-01-01')
)
)
AND S_NATIONKEY = N_NATIONKEY
AND N_NAME = 'CANADA'
ORDER BY
S_NAME
go
1> 2> 3> 4> 5> 6> BEGIN Q20
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20> 21> 22> 23> 24> 25> 26> 27>
28> 29> 30> 31> 32> 33> 34> 35> 36> 37> 38> 39>
40> 41> 42>  S_NAME                   S_ADDRESS
 ------------------------- ----------------------
-----------------
 Supplier#000000020
iybAE,RmTymrZVYaFZva2SH,j
 Supplier#000000091
YV45D7TkfdQanOOZ7q9QxkyGUapU1oOWU6q3
 Supplier#000000197
YC2Acon6kjY3zj3Fbxs2k4Vdf7X0cd2F
 Supplier#000000226        83qOdU2EYRdPQAQhEtn
GRZEd
 Supplier#000000285
Br7e1nnt1yxrw6ImgpJ7YdhFDjuBf
 Supplier#000000378        FfbhyCxWvcPrO8ltp9
 Supplier#000000402
i9Sw4DoyMhzhKXCH9By,AYSgmD
 Supplier#000000530        0qwCMwobKY
OcmLyfRXlagA8ukENJv,
 Supplier#000000688        D
fw5ocppmZpYBBIPI718hCihLDZ5KhKX
 Supplier#000000710        f19YPvOyb
QoYwjKC,oPycpGfieBAcwKJo
 Supplier#000000736
l6i2nMwVuovfKnuVgaSGK2rDy65DlAFLegiL7
 Supplier#000000761
zlSLelQUj2XrvTTFnv7WAcYZGvvMTx882d4
 Supplier#000000884        bmhEShejaS
 Supplier#000000887        urEaTejH5POADP2ARrf
 Supplier#000000935        ij98czM
2KzWe7dDTOxB8sq0UfCdvrX
 Supplier#000000975        ,AC
e,tBpNwKb5xMUzeohxlRn, hdZJo73gFQF8y
 Supplier#000001263        rQWr6nf8ZhB2TAiIDIvo5Io
'-----                -----'
'-----lines surpressed-----'
'-----                -----'
 Supplier#000009252        F7cZaPUHwh1
ZKyj3xmAVWC1XdP ue1p5m,i
 Supplier#000009278        RqYTzgxj93CLX
0mcYfCENOefD
 Supplier#000009327        uoqMdf7e7Gj9dbQ53
 Supplier#000009430        igRqmneFt
 Supplier#000009567
r4Wfx4c3xsEAjcGj71HHZByornl D9vrztXlv4
 Supplier#000009601
51m637bO,Rw5DnHWFUvLacRx9
 Supplier#000009709
rRnCbHYgDgl9PZYnyWKVYSUW0vKg
 Supplier#000009753        wLhVEcRmd7PkJF4FBnGK7Z
 Supplier#000009796        z,y4Idmr15DOvPUqYG
 Supplier#000009799         4wNjXGa4OKWl
 Supplier#000009811        E3iuyq7UnZxU7oPZIe2Gu6
 Supplier#000009812
APFRMy3lCbgFga53n5t9DxzFPQPgnjrGt32
 Supplier#000009862        rJzweWeN58
 Supplier#000009868        ROjGgx5gvtkmnUUoeyy7v
 Supplier#000009869
ucLqxzrpBTRMewGSM29t0rNTM30g1Tu3Xgg3mKag
 Supplier#000009899        7XdpAHrzr1t,UQFZE
 Supplier#000009974
7wJ,J5DKcxSU4Kp1cQLpbcAvB5AsvKT

(204 rows affected)
1>
-- using default substitutions

/* tpch 21.sql */
print 'BEGIN Q21'

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
go
1> 2> 3> 4> 5> 6> BEGIN Q21
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20> 21> 22> 23> 24> 25> 26> 27>
28> 29> 30> 31> 32> 33> 34> 35> 36> 37> 38> 39>
40> 41> 42> 43>  S_NAME                    NUMWAIT
 ------------------------- -----------
 Supplier#000002829                20
 Supplier#000005808                18
 Supplier#000000262                17
 Supplier#000000496                17
 Supplier#000002160                17
 Supplier#000002301                17
 Supplier#000002540                17
 Supplier#000003063                17
 Supplier#000005178                17
 Supplier#000008331                17
 Supplier#000002005                16
 Supplier#000002095                16
 Supplier#000005799                16
 Supplier#000005842                16
 Supplier#000006450                16
 Supplier#000006939                16
 Supplier#000009200                16
 Supplier#000009727                16
 Supplier#000000486                15
 Supplier#000000565                15
 Supplier#000001046                15
 Supplier#000001047                15
 Supplier#000001161                15
 Supplier#000001336                15
 Supplier#000001435                15
 Supplier#000003075                15
 Supplier#000003335                15
 Supplier#000005649                15
 Supplier#000006027                15
 Supplier#000006795                15
 Supplier#000006800                15
 Supplier#000006824                15
 Supplier#000007131                15
 Supplier#000007382                15
 Supplier#000008913                15
 Supplier#000009787                15
 Supplier#000000633                14
 Supplier#000001960                14
 Supplier#000002323                14
 Supplier#000002490                14
 Supplier#000002993                14
 Supplier#000003101                14
 Supplier#000004489                14
 Supplier#000005435                14
 Supplier#000005583                14
 Supplier#000005774                14
 Supplier#000007579                14
 Supplier#000008180                14
 Supplier#000008695                14
 Supplier#000009224                14
 Supplier#000000357                13
 Supplier#000000436                13
 Supplier#000000610                13
 Supplier#000000788                13
 Supplier#000000889                13
 Supplier#000001062                13
 Supplier#000001498                13
 Supplier#000002056                13
 Supplier#000002312                13
 Supplier#000002344                13
 Supplier#000002596                13
 Supplier#000002615                13
 Supplier#000002978                13
 Supplier#000003048                13
 Supplier#000003234                13
 Supplier#000003727                13
 Supplier#000003806                13
 Supplier#000004472                13
 Supplier#000005236                13
 Supplier#000005906                13
 Supplier#000006241                13
 Supplier#000006326                13
 Supplier#000006384                13
 Supplier#000006394                13
 Supplier#000006624                13
 Supplier#000006629                13
 Supplier#000006682                13
 Supplier#000006737                13
 Supplier#000006825                13
 Supplier#000007021                13
 Supplier#000007417                13
 Supplier#000007497                13
 Supplier#000007602                13
 Supplier#000008134                13
 Supplier#000008234                13
 Supplier#000009435                13
 Supplier#000009436                13
 Supplier#000009564                13
 Supplier#000009896                13
 Supplier#000000379                12
 Supplier#000000673                12
 Supplier#000000762                12
 Supplier#000000811                12
 Supplier#000000821                12
 Supplier#000001337                12
 Supplier#000001916                12
 Supplier#000001925                 12
 Supplier#000002039                12
 Supplier#000002357                 12
 Supplier#000002483                 12

(100 rows affected)
1>
-- using default substitutions

/* tpch 22.sql */
print 'BEGIN Q22'

SELECT
CNTRYCODE
,COUNT(*) AS NUMCUST
,SUM(C_ACCTBAL) AS TOTACCTBAL
FROM
(
SELECT
SUBSTRING(C_PHONE,1,2) AS CNTRYCODE
,C_ACCTBAL
FROM
CUSTOMER
WHERE
SUBSTRING(C_PHONE,1,2) IN
('13', '31', '23', '29', '30', '18', '17')
AND C_ACCTBAL > (
SELECT
AVG(C_ACCTBAL)
FROM
CUSTOMER
WHERE
C_ACCTBAL > 0.00
AND SUBSTRING(C_PHONE,1,2) IN
('13', '31', '23', '29', '30', '18', '17')
)
AND NOT EXISTS (
SELECT
*
FROM
ORDERS
WHERE
O_CUSTKEY = C_CUSTKEY
)
) AS CUSTSALE
GROUP BY
CNTRYCODE
ORDER BY
CNTRYCODE
go
1> 2> 3> 4> 5> 6> BEGIN Q22
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14> 15>
16> 17> 18> 19> 20> 21> 22> 23> 24> 25> 26> 27>
28> 29> 30> 31> 32> 33> 34> 35> 36> 37> 38> 39>
CNTRYCODE NUMCUST     TOTACCTBAL
 --------- ----------- ---------------------
 13                888          $6737713.9900
 17                861          $6460573.7200
 18                964          $7236687.4000
 23                892          $6701457.9500
 29                948          $7158866.6300
 30                909          $6808436.1300
 31                922          $6806670.1800

(7 rows affected)
1>
```

Appendix E Seed and Query Substitution Parameters

```
Substitution Parameters for Stream0

-- using 210105040 as a seed to the RNG
Q1 74
Q2 23 BRASS EUROPE EUROPE
Q3 BUILDING 1995-03-23 1995-03-23
Q4 1996-08-01 1996-08-01
Q5 EUROPE 1997-01-01 1997-01-01
Q6 1997-01-01 1997-01-01 0.05 0.05
Q7 JAPAN JORDAN JORDAN JAPAN
Q8 JORDAN MIDDLE EAST SMALL POLISHED STEEL
Q9 firebrick
Q10 1993-09-01 1993-09-01
Q11 MOROCCO 0.0000010000 MOROCCO
Q12 AIR  MAIL 1997-01-01 1997-01-01
Q13 unusual deposits
Q14 1997-12-01 1997-12-01
Q15  1994-07-01 1994-07-01
Q16 Brand#14 LARGE BRUSHED 13 3 7 5 47 1 44 46
Q17 Brand#44 WRAP DRUM
Q18 314
Q19 Brand#52 9 9 Brand#21 10 10 Brand#52 28 28
Q20 green 1994-01-01 1994-01-01 BRAZIL
Q21 ETHIOPIA
Q22 27 14 19 32 29 17 22 27 14 19 32 29 17 22

Substitution Parameters for Stream1

-- using 210105041 as a seed to the RNG
Q1 82
Q2 10 TIN AMERICA AMERICA
Q3 HOUSEHOLD 1995-03-09 1995-03-09
Q4 1994-05-01 1994-05-01
Q5 MIDDLE EAST 1993-01-01 1993-01-01
Q6 1993-01-01 1993-01-01 0.03 0.03
Q7 EGYPT ETHIOPIA ETHIOPIA EGYPT
Q8 EGYPT MIDDLE EAST SMALL BURNISHED STEEL
Q9 cyan
Q10 1994-07-01 1994-07-01
Q11 CANADA 0.0000010000 CANADA
Q12 REG AIR  MAIL 1993-01-01 1993-01-01
Q13 unusual packages
Q14 1993-03-01 1993-03-01
Q15  1997-02-01 1997-02-01
Q16 Brand#54 STANDARD ANODIZED 2 39 41 32 43 11 33 7
Q17 Brand#41 SM BAG
Q18 312
Q19 Brand#14 5 5 Brand#54 11 11 Brand#52 24 24
Q20 rosy 1997-01-01 1997-01-01 PERU
Q21 RUSSIA
Q22 27 31 15 30 29 19 21 27 31 15 30 29 19 21

Substitution Parameters for Stream2

-- using 210105042 as a seed to the RNG
Q1 90
Q2 48 COPPER EUROPE EUROPE
Q3 BUILDING 1995-03-25 1995-03-25
Q4 1996-11-01 1996-11-01
Q5 AFRICA 1993-01-01 1993-01-01
Q6 1993-01-01 1993-01-01 0.08 0.08
Q7 VIETNAM CANADA CANADA VIETNAM
Q8 VIETNAM ASIA STANDARD BRUSHED STEEL
Q9 chiffon
Q10 1993-04-01 1993-04-01
Q11 MOZAMBIQUE 0.0000010000 MOZAMBIQUE
Q12 SHIP  FOB 1993-01-01 1993-01-01
Q13 unusual packages
Q14 1993-06-01 1993-06-01
Q15  1994-11-01 1994-11-01
Q16 Brand#34 MEDIUM PLATED 6 29 28 5 40 14 47 18
Q17 Brand#43 SM PACK
Q18 313
Q19 Brand#11 10 10 Brand#32 12 12 Brand#51 20 20
Q20 cornsilk 1995-01-01 1995-01-01 GERMANY
Q21 KENYA
Q22 14 12 30 29 15 17 22 14 12 30 29 15 17 22

Substitution Parameters for Stream3

-- using 210105043 as a seed to the RNG
Q1 98
Q2 36 STEEL AMERICA AMERICA
Q3 HOUSEHOLD 1995-03-11 1995-03-11
Q4 1994-08-01 1994-08-01
Q5 AMERICA 1993-01-01 1993-01-01
Q6 1993-01-01 1993-01-01 0.06 0.06
Q7 JORDAN BRAZIL BRAZIL JORDAN
Q8 JORDAN MIDDLE EAST STANDARD PLATED COPPER
Q9 blue
Q10 1994-01-01 1994-01-01
Q11 EGYPT 0.0000010000 EGYPT
Q12 FOB  REG AIR 1997-01-01 1997-01-01
Q13 express packages
Q14 1993-09-01 1993-09-01
Q15  1997-06-01 1997-06-01
Q16 Brand#24 ECONOMY POLISHED 15 32 13 43 36 24 47
27
Q17 Brand#44 SM DRUM
Q18 315
Q19 Brand#14 5 5 Brand#25 13 13 Brand#45 27 27
Q20 navy 1994-01-01 1994-01-01 VIETNAM
Q21 FRANCE
Q22 20 28 10 29 19 14 11 20 28 10 29 19 14 11

Substitution Parameters for Stream4

-- using 210105044 as a seed to the RNG
Q1 106
Q2 24 BRASS MIDDLE EAST MIDDLE EAST
Q3 AUTOMOBILE 1995-03-27 1995-03-27
Q4 1997-03-01 1997-03-01
Q5 ASIA 1993-01-01 1993-01-01
Q6 1993-01-01 1993-01-01 0.03 0.03
Q7 ETHIOPIA ALGERIA ALGERIA ETHIOPIA
Q8 ETHIOPIA AFRICA STANDARD ANODIZED COPPER
Q9 aquamarine
Q10 1994-10-01 1994-10-01
Q11 PERU 0.0000010000 PERU
Q12 MAIL  FOB 1993-01-01 1993-01-01
Q13 express packages
Q14 1994-01-01 1994-01-01
Q15  1995-02-01 1995-02-01
Q16 Brand#54 SMALL ANODIZED 12 16 47 23 33 22 18 44
Q17 Brand#41 LG BAG
Q18 312
Q19 Brand#21 10 10 Brand#53 14 14 Brand#44 23 23
Q20 azure 1997-01-01 1997-01-01 IRAQ
Q21 UNITED KINGDOM
Q22 21 26 14 24 16 13 20 21 26 14 24 16 13 20

Substitution Parameters for Stream5

-- using 210105045 as a seed to the RNG
Q1 114
Q2 12 NICKEL AMERICA AMERICA
Q3 HOUSEHOLD 1995-03-13 1995-03-13
Q4 1994-12-01 1994-12-01
Q5 EUROPE 1994-01-01 1994-01-01
Q6 1994-01-01 1994-01-01 0.09 0.09
Q7 RUSSIA UNITED STATES UNITED STATES RUSSIA
Q8 RUSSIA EUROPE PROMO POLISHED COPPER
Q9 violet
Q10 1993-07-01 1993-07-01
Q11 ETHIOPIA 0.0000010000 ETHIOPIA
Q12 TRUCK  FOB 1994-01-01 1994-01-01
Q13 express packages
Q14 1994-04-01 1994-04-01
Q15  1997-09-01 1997-09-01
```

```
Q16 Brand#34 LARGE BURNISHED 8 10 34 28 33 2 26 25
Q17 Brand#43 LG PACK
Q18 314
Q19 Brand#23 6 6 Brand#41 15 15 Brand#34 20 20
Q20 lavender 1996-01-01 1996-01-01 ARGENTINA
Q21 MOROCCO
Q22 30 17 16 23 28 11 10 30 17 16 23 28 11 10
```

Appendix F Implementation Specific Layer and Source Code

SETUP.CMD – USED TO CREATE AND LOAD THE DATABASE

```
echo off

rem
rem Modify the following parameters for your configuration
rem
rem Make certain that DBGEN_PARALLELISM is GT 4 for this version

set DB=tpch100g
set HOMEDRIVE=d:
set HOMEDIR=\ScriptedTPCH\Setup
set OUTPUTDRIVE=d:
set OUTPUTDIR=\ScriptedTPCH\Output
set SCALEFACTOR=100
set DBGEN_PARALLELISM=8
set FLATFILEDRIVE=d:
set FLATFILEDIR=\dev\FileSys3
set RF1_PARALLELISM=16
set RF2_PARALLELISM=16
set UPDATEDRIVE=d:
set UPDATEDIR=\dev\FileSys3
set UPDATE_SETS=12

set DoDBGEN=TRUE
set DoDBCREATE=TRUE
set DoBULKINSERT=TRUE
set DoCLEANUP=TRUE
set DoBACKUP=TRUE

%HOMEDRIVE%
cd %HOMEDIR%

if '%1' == 'DBGEN' goto :DBGEN
if '%1' == 'BULKINSERTn' goto :BULKINSERTn

echo Checking for existence of HOMEDIR and OUTPUTDIR
if NOT EXIST %HOMEDRIVE%%HOMEDIR% goto :ERROR_EXIT
if NOT EXIST %OUTPUTDRIVE%%OUTPUTDIR% goto :ERROR_EXIT
if NOT EXIST %UPDATEDRIVE%%UPDATEDIR% goto :ERROR_EXIT

echo Finding next output directory in %OUTPUTDRIVE%%OUTPUTDIR%
set OUTPUTNUMBER=1
:OUTPUTLOOP
  if NOT EXIST %OUTPUTDRIVE%%OUTPUTDIR%\%OUTPUTNUMBER% goto
:OUTPUTLOOPEND
  set /a OUTPUTNUMBER=%OUTPUTNUMBER%+1
  goto :OUTPUTLOOP
:OUTPUTLOOPEND
set OUTPUTPATH=%OUTPUTDRIVE%%OUTPUTDIR%\%OUTPUTNUMBER%
echo Output will be found at %OUTPUTPATH%
mkdir %OUTPUTPATH%

if NOT '%DoDBGEN%' == 'TRUE' goto :DBCREATE

rem
rem DBGEN invokes dbgen.exe in parallel
rem

echo Starting DBGEN of FlatFiles
copy dists.dss %FLATFILEDRIVE%\%FLATFILEDIR%
for /l %%i in (1,1,4) do start cmd /C Setup DBGEN %%i
semaphore -wait DBGEN -count 4
for /l %%i in (5,1,%DBGEN_PARALLELISM%) do start cmd /C Setup DBGEN %%i
set /a DBGENS=%DBGEN_PARALLELISM%-4
semaphore -wait DBGEN -count %DBGENS%
echo Starting DBGEN of Update Files
%UPDATEDRIVE%
cd %UPDATEDIR%
%HOMEDRIVE%%HOMEDIR%\dbgen -U %UPDATE_SETS% -s
%SCALEFACTOR% -qf -C %UPDATE_SETS% -i %RF1_PARALLELISM% -d
%RF2_PARALLELISM% 2>%OUTPUTPATH%\dbgen_Update.out
%HOMEDRIVE%
cd %HOMEDIR%
goto :DBCREATE

:DBGEN
%FLATFILEDRIVE%
cd %FLATFILEDIR%
%HOMEDRIVE%%HOMEDIR%\dbgen -qfF -s%SCALEFACTOR% -
C%DBGEN_PARALLELISM% -S%2 2>%OUTPUTPATH%\dbgen_%2%.out
%HOMEDRIVE%
cd %HOMEDIR%
semaphore -release DBGEN
goto :EOF

:DBCREATE
if NOT '%DoDBCREATE%' == 'TRUE' goto :BULKINSERT
rem
rem DBCREATE invokes the file %DBNAME%\CreateDatabase.sql
rem
echo Starting database creation
osql -Usa -P -Q"if exists (select * from sysdatabases where name='%DB%')drop
database %DB%" -o %OUTPUTPATH%\DropDatabase.out -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -i %DB%\CreateDatabase.sql -o %OUTPUTPATH%\CreateDatabase.out
-b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -Q"sp_dboption %DB%,'trunc',TRUE" -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -Q"sp_dboption %DB%,'select',TRUE" -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -Q"sp_dboption %DB%,'torn',FALSE" -b
if ERRORLEVEL 1 goto :ERROR_EXIT

:BULKINSERT
if NOT '%DoBULKINSERT%' == 'TRUE' goto :CREATEINDEXES
rem
rem BULKINSERT starts a process per dbgen segment
rem

echo Dropping and Re-Creating Tables
for %%i in (LINEITEM ORDERS CUSTOMER PART PARTSUPP SUPPLIER NATION
REGION) do osql -Usa -P -d%DB% -Q"drop table %%i" -o
%OUTPUTPATH%\Drop_Table_%%i.out
osql -Usa -P -d%DB% -i%DB%\CreateTables.sql -o
%OUTPUTPATH%\CreateTables.out -b
if ERRORLEVEL 1 goto :ERROR_EXIT

echo Starting bulk inserts
osql -Usa -P -d%DB% -Q"if exists (select * from sysindexes where name =
'LOADTIMES') drop table LOADTIMES" -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -Q"create table LOADTIMES(STEP char(35),TIMESTAMP
datetime)" -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -Q"insert into LOADTIMES values ('LOAD begin',getdate())" -b
if ERRORLEVEL 1 goto :ERROR_EXIT
for %%i in (LINEITEM ORDERS CUSTOMER PART PARTSUPP SUPPLIER) do call
:BULKINSERTi %%i
echo for NATION and REGION
osql -Usa -P -d%DB% -Q"insert into LOADTIMES values ('Nation/Region insert
begin',getdate())" -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -Q"bulk insert NATION from
'%FLATFILEDRIVE%%FLATFILEDIR%\Nation.tbl' with (FieldTerminator = '|',
RowTerminator ='|\n',tablock)" -o %OUTPUTPATH%\BulkInsert_Nation.out -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -i%DB%\CreateNATIONIndexes.sql -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -Q"bulk insert REGION from
'%FLATFILEDRIVE%%FLATFILEDIR%\Region.tbl' with (FieldTerminator = '|',
RowTerminator ='|\n',tablock)" -o %OUTPUTPATH%\BulkInsert_Region.out -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -i%DB%\CreateREGIONIndexes.sql -b
```

```
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -Q"insert into LOADTIMES values ('Nation/Region insert
end',getdate())" -b
if ERRORLEVEL 1 goto :ERROR_EXIT
goto :CLEANUP

:BULKINSERTi
echo Starting bulk inserts for %1
osql -Usa -P -d%DB% -Q"insert into LOADTIMES values ('%1 bulk insert
begin',getdate())"
for /l %%j in (1,1,%DBGEN_PARALLELISM%) do start cmd /C Setup BULKINSERTn
%1 %%j
semaphore -wait %1 -count %DBGEN_PARALLELISM%
echo Starting create indexes for %1
osql -Usa -P -d%DB% -Q"insert into LOADTIMES values ('%1 create index
begin',getdate())" -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -i %DB%\Create%1Indexes.sql -o
%OUTPUTPATH%\Create%1Indexes.out -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -Q"insert into LOADTIMES values ('%1 end',getdate())" -b
if ERRORLEVEL 1 goto :ERROR_EXIT
goto :EOF

:BULKINSERTn
osql -Usa -P -d%DB% -Q"bulk insert %2 from
'%FLATFILEDRIVE%%FLATFILEDIR%\%2.tbl.%3' with (FieldTerminator = '|',
RowTerminator ='|\n',tablock)" -o %OUTPUTPATH%\BulkInsert_%2_%3.out -b
semaphore -release %2
goto :EOF


:CLEANUP
if NOT '%DoCLEANUP%' == 'TRUE' goto :BACKUP
rem
rem CLEANUP sets statistics and lock options
rem

echo Setting Cleanup Options

osql -Usa -P -d%DB% -Q"insert into LOADTIMES values ('Cleanup start',getdate())" -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -Q"sp_createstats" -o %OUTPUTPATH%\CreateStats.out -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -Q"sp_dboption '%DB%','auto create statistics','OFF'" -o
%OUTPUTPATH%\AutoCreateStats.out
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -Q"sp_dboption '%DB%','auto update statistics','OFF'" -o
%OUTPUTPATH%\AutoUpdateStats.out
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -Q"sp_indexoption 'LINEITEM', 'disallowpagelocks', 'TRUE'" -o
%OUTPUTPATH%\DisAllowPageLocksLINEITEM.out -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -Q"sp_indexoption 'ORDERS', 'disallowpagelocks', 'TRUE'" -o
%OUTPUTPATH%\DisAllowPageLocksOrders.out -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -i CreateRF1Proc.sql -o %OUTPUTPATH%\CreateRF1Proc.out
-b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -i CreateRF2Proc.sql -o %OUTPUTPATH%\CreateRF2Proc.out
-b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -Q"exec sp_tableoption 'NATION','pintable',1" -o
%OUTPUTPATH%\pinNATION.out -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -Q"exec sp_tableoption 'REGION','pintable',1" -o
%OUTPUTPATH%\pinREGION.out -b
if ERRORLEVEL 1 goto :ERROR_EXIT
rem
rem osql -Usa -P -d%DB% -Q"exec sp_tableoption 'SUPPLIER','pintable',1" -o
%OUTPUTPATH%\pinSUPPLIER.out -b
rem if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -Q"insert into LOADTIMES values ('Cleanup end',getdate())" -b
if ERRORLEVEL 1 goto :ERROR_EXIT

:BACKUP
rem
rem BACKUP is the final step, using a script
rem

if NOT '%DoBACKUP%' == 'TRUE' goto :DONE

echo Starting Backup
osql -Usa -P -d%DB% -Q"insert into LOADTIMES values ('Backup start',getdate())" -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -i %DB%\Backup.sql -o %OUTPUTPATH%\Backup.out -b
if ERRORLEVEL 1 goto :ERROR_EXIT
osql -Usa -P -d%DB% -Q"insert into LOADTIMES values ('Backup end',getdate())" -b
if ERRORLEVEL 1 goto :ERROR_EXIT

:DONE

osql -Usa -P -d%DB% -Q"insert into LOADTIMES values ('LOAD end',getdate())"
osql -Usa -P -d%DB% -Q"select TIMESTAMP from LOADTIMES where STEP='LOAD
end'" -o %OUTPUTPATH%\LoadEND.out
echo Last step is to run Francois' dbtables-ms.sql script
osql -Usa -P -d%DB% -i dbtables-ms.sql -o %OUTPUTPATH%\dbtables-ms.out
echo Done!  Check for output in %OUTPUTPATH%
goto :EOF

:ERROR_EXIT
echo Setup aborted due to errors
echo Check output in %OUTPUTPATH%
exit /B
```

RUN.CMD – USED FOR POWER AND THROUGHPUT TESTS

```
echo off

rem
rem Modify the following parameters for your configuration
rem

set DB=tpch100g
set HOMEDRIVE=d:
set HOMEDIR=\ScriptedTpch\Run
set OUTPUTDRIVE=d:
set OUTPUTDIR=\ScriptedTpch\Output
set UPDATEDRIVE=d:
set UPDATEDIR=\dev\filesys3
set UPDATESET=7
set UPDATESEGMENTS=16
set SEED=210105040
set SCALEFACTOR=100

set DoRESTORE=FALSE
set DoSETUP=TRUE
set DoPOWER=TRUE
set DoPOWERRefresh=TRUE
set DoPOWERStream0=TRUE
set DoTHROUGHPUT=TRUE

%HOMEDRIVE%
cd %HOMEDIR%

if '%1' == 'RF1' goto RF1
if '%1' == 'RF2' goto RF2
if '%1' == 'THROUGHPUT_STREAM' goto :THROUGHPUT_STREAM

echo Checking for existence of HOMEDIR and OUTPUTDIR
if NOT EXIST %HOMEDRIVE%%HOMEDIR% goto :ERROR_EXIT
if NOT EXIST %OUTPUTDRIVE%%OUTPUTDIR% goto :ERROR_EXIT

set OUTPUTNUMBER=1
:OUTPUTLOOP
  if NOT EXIST %OUTPUTDRIVE%%OUTPUTDIR%\%OUTPUTNUMBER% goto
:OUTPUTLOOPEND
  set /a OUTPUTNUMBER=%OUTPUTNUMBER%+1
  goto :OUTPUTLOOP
:OUTPUTLOOPEND
set OUTPUTPATH=%OUTPUTDRIVE%%OUTPUTDIR%\%OUTPUTNUMBER%
```

```
echo Output will be found at %OUTPUTPATH%
mkdir %OUTPUTPATH%

if NOT '%DoRESTORE%' == 'TRUE' goto :SETUP

rem
rem Use the Restore.sql script in SETUP
rem

echo Starting Restore
osql -Usa -P -i ..\Setup\%DB%\Restore.sql -b
if ERRORLEVEL 1 goto :ERROR_EXIT

:SETUP
if NOT '%DoSETUP%' == 'TRUE' goto :POWER

rem
rem Create the Power and five Throughput Streams
rem

echo QGening the Power and Throughput Streams
pushd templates
for /l %%i in (0,1,5) do %HOMEDIR%\qgen -s %SCALEFACTOR% -r %SEED% -
p%%i > %HOMEDIR%\Stream%%i.sql
popd

rem
rem Create the TIMES table
rem
osql -Usa -P -d%DB% -Q"if exists (select * from sysindexes where name = 'TIMES')
drop table TIMES"
osql -Usa -P -d%DB% -Q"create table TIMES(QUERY char(5),STREAM int,START
datetime)"

rem
rem Begin POWER run
rem


rem
rem Execute the RF1 Transaction set in parallel
rem

:POWER
if NOT '%DoPOWER%' == 'TRUE' goto :THROUGHPUT
echo Beginning Power Run
if NOT '%DoPOWERRefresh%' == 'TRUE' goto :STREAM0

echo Running the RF1s
set /a UPDATE_SEGMENT=%1+%UPDATESET%
osql -Usa -P -d%DB% -Q"insert into TIMES values ('RF1',0,getdate())"
for /l %%i in (1,1,%UPDATESEGMENTS%) do start /abovenormal cmd /C Run RF1
%UPDATE_SEGMENT% %%i
semaphore -wait RF1 -count %UPDATESEGMENTS%

rem
rem Execute the PowerRun Queries
rem

:STREAM0
if NOT '%DoPOWERStream0%' == 'TRUE' goto :STREAM0_DONE
echo Running Stream0
osql -E -d%DB% -iStream0.sql -o %OUTPUTPATH%\Stream0.out -b
if ERRORLEVEL 1 goto :ERROR_EXIT
:STREAM0_DONE

rem
rem Execute the RF2 Transaction set in Parallel
rem

if NOT '%DoPOWERRefresh%' == 'TRUE' goto :POWERDONE
echo Running the RF2s
osql -Usa -P -d%DB% -Q"insert into TIMES values ('RF2',0,getdate())"
for /l %%i in (1,1,%UPDATESEGMENTS%) do start /abovenormal cmd /C Run RF2
%UPDATE_SEGMENT% %%i
semaphore -wait RF2 -count %UPDATESEGMENTS%

:POWERDONE
osql -Usa -P -d%DB% -Q"insert into TIMES values ('QXX',0,getdate())"

rem
rem Execute the THROUGHPUT Run
rem

:THROUGHPUT
if NOT '%DoTHROUGHPUT%' == 'TRUE' goto :DONE
echo Running the Throughput Streams

for /l %%i in (1,1,5) do start /abovenormal cmd /C Run THROUGHPUT_STREAM %%i
osql -Usa -P -d%DB% -Q"insert into TIMES values ('RF1',1,getdate())"
semaphore -wait THROUGHPUT_QUERIES -count 5

for /l %%j in (1,1,5) do call :THROUGHPUT_REFRESH %%j
goto :DONE

:THROUGHPUT_STREAM
osql -E -d%DB% -iStream%2.sql -o %OUTPUTPATH%\Stream%2.out
osql -Usa -P -d%DB% -Q"insert into TIMES values ('QXX',%2,getdate())"
semaphore -release THROUGHPUT_QUERIES
goto :EOF

:THROUGHPUT_REFRESH
if %1 neq 1 osql -Usa -P -d%DB% -Q"insert into TIMES values ('RF1',%1,getdate())"
set /a UPDATE_SEGMENT=%1+%UPDATESET%
for /l %%i in (1,1,%UPDATESEGMENTS%) do start /abovenormal cmd /C Run RF1
%UPDATE_SEGMENT% %%i
semaphore -wait RF1 -count %UPDATESEGMENTS%
osql -Usa -P -d%DB% -Q"insert into TIMES values ('RF2',%1,getdate())"
for /l %%i in (1,1,%UPDATESEGMENTS%) do start /abovenormal cmd /C Run RF2
%UPDATE_SEGMENT% %%i
semaphore -wait RF2 -count %UPDATESEGMENTS%
osql -Usa -P -d%DB% -Q"insert into TIMES values ('RFX',%1,getdate())"
goto :EOF

rem
rem Final Step -- Write out completion to Log
rem

:DONE
osql -Usa -P -d%DB% -iReport.sql -o%OUTPUTPATH%\Report.out
echo Done! Output can be found at %OUTPUTPATH%
goto :EOF

rem
rem Subroutine for Executing RF1s
rem called from RF1 and THROUGHPUT_RF1
rem

:RF1
osql -Usa -P -l 120 -d%DB% -Q"exec RF1
'%UPDATEDRIVE%%UPDATEDIR%',%2,%3" -o
%OUTPUTPATH%\RF1_%2_%3.out
semaphore -release RF1
goto :EOF

rem
rem Subroutine for Executing RF2s
rem called from RF2 and THROUGHPUT_RF2
rem

:RF2
osql -Usa -P -l 120 -d%DB% -Q"exec RF2
'%UPDATEDRIVE%%UPDATEDIR%',%2,%3" -o
%OUTPUTPATH%\RF2_%2_%3.out
semaphore -release RF2
goto :EOF

:ERROR_EXIT
echo Run aborted due to error
echo Check output in %OUTPUTPATH%
exit /B
```

SEMAPHORE.CPP – USED BY THE SETUP AND RUN SCRIPTS

```
#define _WIN32_WINNT            0x0400

#include <windows.h>
#include <string.h>
#include <iostream.h>
#include <stdlib.h>
#include <stdio.h>
#include <assert.h>

void main(int argc, char **argv)
{

        typedef enum { eUnknown, eWait, eSignal, eRelease, eWaitList,
eWaitGroup } OPERATION;

        OPERATION           eOP = eUnknown;

        int                 iCount;
        int                 i;

        HANDLE              hSemaphore;
        HANDLE              *pHandles;
        SYSTEMTIME          Time;

        if (argc < 3)
                goto usage;

        if (_stricmp(argv[1], "-wait") == 0)
                eOP = eWait;
        else if (_stricmp(argv[1], "-signal") == 0)
                eOP = eSignal;
        else if (_stricmp(argv[1], "-release") == 0)
                eOP = eRelease;
        else if (_stricmp(argv[1], "-waitlist") == 0)
                eOP = eWaitList;
        else if (_stricmp(argv[1], "-waitgroup") == 0)
                eOP = eWaitGroup;
        else goto usage;


        if ((eOP == eWait) || (eOP == eRelease))
        {
                // argv[2] is the semaphore name
                // if -count option specified, then there must be exactly 5 args
                if ((argc == 5) && (_stricmp(argv[3], "-count") == 0))
                {
                        iCount = atoi(argv[4]);
                        if (iCount < 1)
                                goto usage;
                }
                        // check that
                else if (argc != 3)
                        goto usage;
                else
                        iCount = 1;
        }
        else if (eOP == eWaitGroup)
        {
                if ((argc != 5) || (_stricmp(argv[3], "-count") != 0))
                        goto usage;
                iCount = atoi(argv[4]);
                if (iCount < 1)
                        goto usage;
        }
        else
                // eWaitList or eSignal
                iCount = argc - 2;

        if (eOP == eWait)
        {
                printf( "semaphore name  = %s\n", argv[2] );
                printf( "semaphore count = %d\n", iCount );
                hSemaphore = CreateSemaphore( NULL, 0, 2000000000,
argv[2] );
                if (hSemaphore == NULL)
                {
                        DWORD dwError = GetLastError();
                        cout << "*ERROR*  CreateSemaphore returned "
<< dwError << endl;
                        exit(EXIT_FAILURE);
                }
                for (i=0; i<iCount; i++)
                {
                        WaitForSingleObject( hSemaphore, INFINITE );
                        GetLocalTime( &Time );
                        printf( "%4.4d-%2.2d-%2.2d %2.2d:%2.2d:%2.2d
- released \n",
                                Time.wYear, Time.wMonth,
Time.wDay, Time.wHour, Time.wMinute, Time.wSecond );
                }
                CloseHandle( hSemaphore );
        }
        else if ((eOP == eWaitGroup) || (eOP == eWaitList))
        {
                char **szEventNames;
                szEventNames = new char*[iCount];
                char szTmp[128];

                printf( "event-list =" );
                for (i=0; i<iCount; i++)
                {
                        if (eOP == eWaitGroup)
                        {
                                wsprintf( szTmp, "%s.%d", argv[2],
i+1 );
                                szEventNames[i] = new
char[strlen(szTmp)+1];
                                strcpy( szEventNames[i], szTmp );
                        }
                        else
                        {
                                szEventNames[i] = new
char[strlen(argv[i+2])+1];
                                strcpy( szEventNames[i], argv[i+2] );
                        }

                        printf( " %s", szEventNames[i] );
                }
                printf( "\n" );

                pHandles = new HANDLE[iCount-1];
                for (i=0; i<iCount; i++)
                {
                        pHandles[i] = CreateEvent( NULL, TRUE /*
manual reset */, FALSE /* initially non-signaled */, szEventNames[i] );
                        if (pHandles[i] == NULL)
                        {
                                DWORD dwError = GetLastError();
                                cout << "*ERROR*  CreateEvent
returned " << dwError << endl;
                                exit(EXIT_FAILURE);
                        }
                }
                for (i=iCount; i>0;i--)
                {
                        int idx = WaitForMultipleObjects( i, pHandles,
FALSE /* wait for all */, INFINITE ) - WAIT_OBJECT_0;
                        GetLocalTime( &Time );
                        printf( "%4.4d-%2.2d-%2.2d %2.2d:%2.2d:%2.2d
- signaled: %s \n",
                                Time.wYear, Time.wMonth,
Time.wDay, Time.wHour, Time.wMinute, Time.wSecond, szEventNames[idx]  );

                        HANDLE hTmp = pHandles[idx];
                        pHandles[idx] = pHandles[i-1];
                        pHandles[i-1] = hTmp;

                        char* szTmp = szEventNames[idx];
                        szEventNames[idx] = szEventNames[i-1];
                        szEventNames[i-1] = szTmp;
```

```
				}
				for (i=0; i<iCount; i++)
						CloseHandle( pHandles[i] );
		}
		else if (eOP == eRelease)
		{
				hSemaphore = OpenSemaphore(
SEMAPHORE_MODIFY_STATE, FALSE, argv[2] );
				if (hSemaphore == NULL)
				{
						DWORD dwError = GetLastError();
						cout << "*ERROR*  OpenSemaphore returned "
<< dwError << endl;
						exit(EXIT_FAILURE);
				}
				if (!ReleaseSemaphore( hSemaphore, iCount, NULL ))
				{
						DWORD dwError = GetLastError();
						cout << "*ERROR*  ReleaseSemaphore returned
" << dwError << endl;
						exit(EXIT_FAILURE);
				}
				CloseHandle( hSemaphore );
		}
		else if (eOP == eSignal)
		{
				for (i=0; i<iCount; i++)
				{
						HANDLE hHandle = OpenEvent(
EVENT_MODIFY_STATE, FALSE, argv[i+2] );
						if (hHandle == NULL)
						{
								DWORD dwError = GetLastError();
								cout << "*ERROR*  OpenEvent
returned " << dwError << endl;
								exit(EXIT_FAILURE);
						}
						SetEvent( hHandle );
						CloseHandle( hHandle );
				}
		}

		exit(EXIT_SUCCESS);

  // syntax was bad; show usage and quit
usage:
		printf(
				"Semaphore Utility - Ver. 1.2 - 26-Jul-99 \n"
				"Copyright (C) Microsoft Corp 1999.  All rights reserved.\n\n"
				"usage: \n"
				"  semaphore { -wait | -release } <semaphore-name> [ -count
<count> ] \n"
				"  semaphore { -waitlist | -signal } <event-list> \n"
				"  semaphore -waitgroup <event-prefix> -count <count>\n"
				"\n"
				"   <semaphore-name> == alpha-numeric identifier \n"
				"   <count> == integer > 0; default value = 1 \n"
				"   <event-list> == { <event-name> ... } \n"
				"   <event-name> == alpha-numeric identifier \n"
				"   <event-prefix> == alpha-numeric identifier \n"
				"\n"
				"There are two modes to choose from: a semaphore or a list
of events. \n"
				"\n"
				"Semaphore mode: \n"
				"A semaphore is a single identifier with an associated count.
Each time \n"
				"the semaphore is released, the count is decremented by one
(or the amount \n"
				"specified).  When the count reaches zero, the waiter
completes. If there \n"
				"are multiple waiters on the same semaphore, each release
releases only \n"
				"the number of waiters specified in count.\n"
				"\n"
				"List of Events: \n"
				"A list of events (alpha-numeric tags) is specified for the
waiter.  The \n"
				"waiter doesn't complete until all of the events have been
signaled.  A \n"
				"given event may be signaled more than once.  There are two
ways to define \n"
				"the list of events, either explicitly (-waitlist) by naming all of
them or \n"
				"implicitly (-waitgroup) with a prefix and a count.  Using the -
waitgroup \n"
				"option, you provide an alpha-numeric tag which is used as
the prefix for a \n"
				"group of events.  The event names are generated by
concatenating the prefix \n"
				"with \".<n>\", where <n> is 1 to the specified count. \n"
				);

		exit(EXIT_FAILURE);
}
```

Appendix G Refresh Function Source Code

RF1 Stored Procedure

```
-- CreateRF1Proc.sql

if exists (select name from sysobjects where name =
'RF1')
       drop procedure RF1
GO

--
--  Create a stored RefreshInsert procedure which
will catch the deadlock
--  victim abort and restart the insert transaction.
--
CREATE PROCEDURE RF1
@flatfiledir CHAR(40), @updateset INTEGER, @segment
INTEGER
AS
BEGIN

DECLARE @min_orderkey INTEGER
DECLARE @max_orderkey INTEGER
DECLARE @range INTEGER
DECLARE @max_set INTEGER
DECLARE @SQLstring NVARCHAR(255)
DECLARE @insert_sets INTEGER

set @insert_sets=100

--
--  Create the insert tables
--
create table #NEWORDERS (
        O_ORDERKEY int not null,
        O_CUSTKEY int not null,
        O_ORDERSTATUS char(1)  not null,
        O_TOTALPRICE  money  not null,
        O_ORDERDATE datetime not null,
        O_ORDERPRIORITY char(15)  not null,
        O_CLERK  char(15) not null,
        O_SHIPPRIORITY  int not null,
        O_COMMENT varchar(79) not null
)
create table #NEWLINEITEM(
       L_ORDERKEY  int  not null,
       L_PARTKEY int not null,
       L_SUPPKEY int not null,
       L_LINENUMBER int not null,
       L_QUANTITY money not null,
       L_EXTENDEDPRICE money not null,
       L_DISCOUNT money not null,
       L_TAX money not null,
       L_RETURNFLAG char(1) not null,
       L_LINESTATUS char(1) not null,
       L_SHIPDATE datetime not null,
       L_COMMITDATE datetime not null,
       L_RECEIPTDATE datetime not null,
       L_SHIPINSTRUCT char(25)  not null,
       L_SHIPMODE char(10 ) not null,
       L_COMMENT varchar(44) not null
)

create unique clustered index NEWORDERS on
#NEWORDERS (O_ORDERKEY)
create clustered index NEWLINEITEM on #NEWLINEITEM
(L_ORDERKEY)

--
--  Generate an SQL statement inserting the current
updateset value into
--   the command.  Next execute the statement to
bulk load the new lineitem
--  insert values.
--
SET @SQLstring="bulk insert #NEWLINEITEM from '"
              + RTRIM(convert(CHAR,@flatfiledir))
+"\Lineitem.tbl.u"
              + RTRIM(Convert(char,@updateset)) +
"."
              + RTRIM(convert(char,@segment))
              + "' with
(FieldTerminator='|',RowTerminator='|\n',order(L_ORD
ERKEY),codepage='RAW')"
EXEC sp_executesql @SQLstring

--
--  Generate an SQL statement inserting the current
updateset value into
--  the command.  Next execute the statement to bulk
load the new order
--  insert values.
--
SET @SQLstring="bulk insert #NEWORDERS from '"
              + RTRIM(convert(CHAR,@flatfiledir))
+"\Orders.tbl.u"
              + RTRIM(Convert(char,@updateset)) +
"."
              + RTRIM(convert(char,@segment))
              + "' with
(FieldTerminator='|',RowTerminator='|\n',order(O_ORD
ERKEY),CODEPAGE='RAW')"

EXEC sp_executesql @SQLstring

--
--  Obtain minimum and maximum order key and compute
the range of each
--  set to be inserted into the ORDERS and LINEITEM
tables.
--
SELECT
@min_orderkey=MIN(O_ORDERKEY),@max_orderkey=MAX(O_OR
DERKEY) FROM #NEWORDERS
SET @range = (@max_orderkey - @min_orderkey) /
@insert_sets

--
--  This handles the case when the max-
min/insert_sets is less that 1
--
IF @range = 0
-- BEGIN
       SET @range = (@max_orderkey - @min_orderkey)
/ 1
-- END

--
--  Loop through the order keys only inserting a
sets into the
--  ORDERS and LINTEITEM tables
--
SET @max_set = @min_orderkey - 1
WHILE @max_set < @max_orderkey
BEGIN
  --
  -- Set the range from min_orderkey to max_set
  --
  SET @max_set = @min_orderkey + @range
  if @max_set > @max_orderkey
     SET @max_set = @max_orderkey + 1

  --
  --  Insert into ORDERS and LINEITEM tables
  --
  INSERT_TRANS:
  begin transaction
      insert into ORDERS SELECT * FROM #NEWORDERS
```

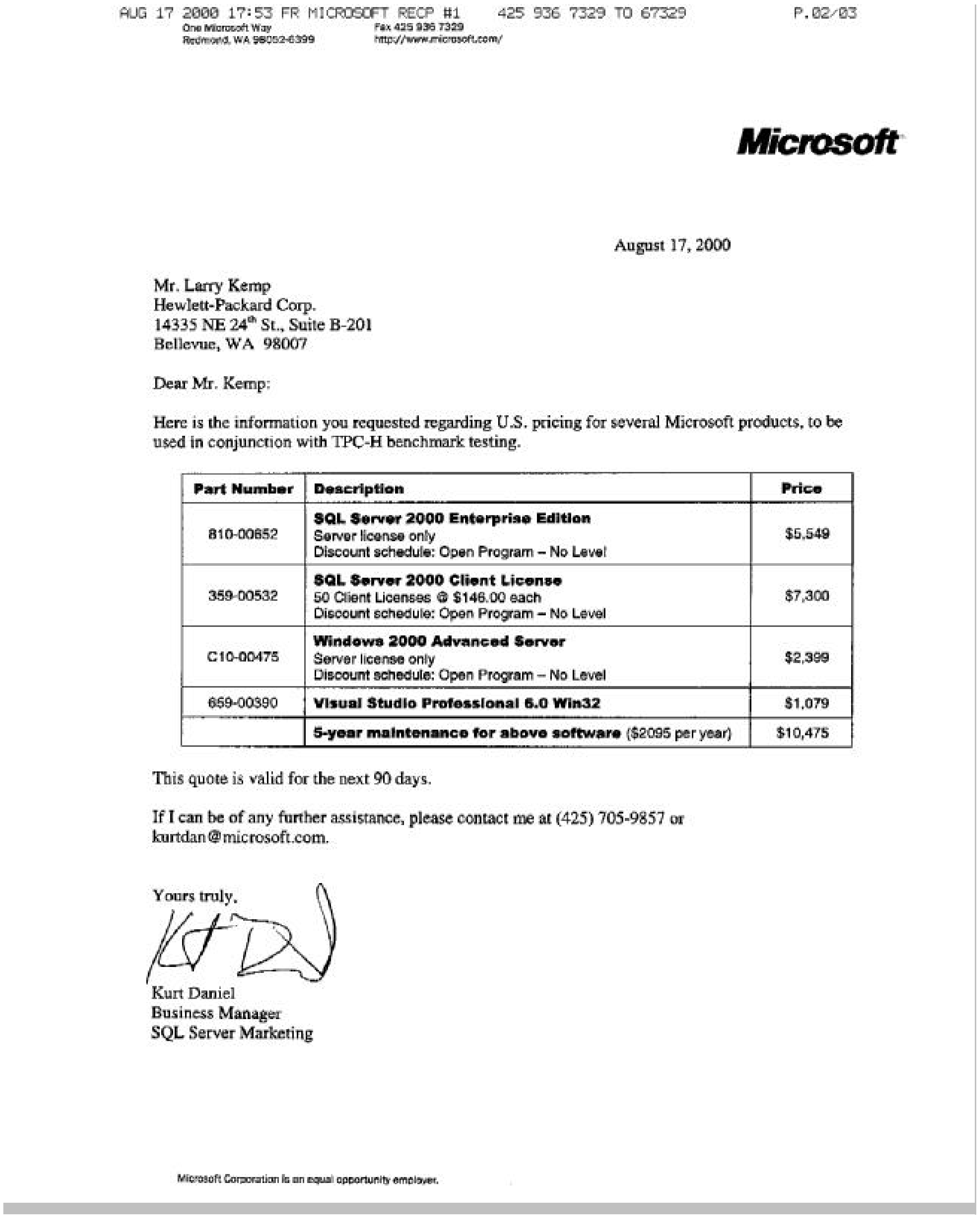
```
             WHERE O_ORDERKEY >= @min_orderkey AND
O_ORDERKEY < @max_set
      insert into LINEITEM SELECT * FROM
#NEWLINEITEM
             WHERE L_ORDERKEY >= @min_orderkey AND
L_ORDERKEY < @max_set
  commit transaction

  --
  --  If deadlock victim abort then restart the
transaction
  --
  if (@@error = 1205)
    BEGIN
      print 'Insert deadlock - restarting RF1
transaction'
      rollback transaction
      GOTO INSERT_TRANS
    END

  --
  --  Move min_orderkey to start of next insert set
  --
  SET @min_orderkey = @max_set
END

END

GO
```

RF2 Stored Procedure

```
-- CreateRF2Proc.sql

if exists (select name from sysobjects where name =
'RF2')
       drop procedure RF2
GO

--
--  Create a stored Refresh Delete procedure which
will catch the deadlock
--  victim abort and restart the delete transaction.
--
CREATE PROCEDURE RF2
@flatfiledir CHAR(40), @updateset INTEGER, @segment
INTEGER
AS
BEGIN

DECLARE @min_orderkey INTEGER
DECLARE @max_orderkey INTEGER
DECLARE @range INTEGER
DECLARE @max_set INTEGER
DECLARE @SQLstring NVARCHAR(255)
DECLARE @delete_sets INTEGER

set @delete_sets=100

create table #OLDORDERS  (O_ORDERKEY int)
create unique clustered index OLDORDERS
  on #OLDORDERS (O_ORDERKEY)
  with sorted_data

--
--  Generate an SQL statement inserting the current
updateset value into
--  the command.  Next execute the statement to bulk
load the old order
--  delete values
--
SET @SQLstring="bulk insert #OLDORDERS from '"
              + RTRIM(convert(CHAR,@flatfiledir))
+"\Delete.u"
              + RTRIM(Convert(char,@updateset)) +
"."
              + RTRIM(convert(char,@segment))
              + "' with
(order(O_ORDERKEY),codepage='RAW')"
EXEC sp_executesql @SQLstring

--
--  Obtain minimum and maximum order key and compute
the
--  range of each delete set
--
SELECT
@min_orderkey=MIN(O_ORDERKEY),@max_orderkey=MAX(O_OR
DERKEY) FROM #OLDORDERS
SET @range = (@max_orderkey - @min_orderkey) /
@delete_sets

--
--  This handles the case when the max-
min/delete_sets is less that 1
--
IF @range = 0
-- BEGIN
       SET @range = (@max_orderkey - @min_orderkey)
/ 1
-- END

--
--  Loop through the order keys only deleting sets
from orders
--  and lineitem tables
--
SET @max_set = @min_orderkey - 1
WHILE @max_set < @max_orderkey
BEGIN
  --
  -- Set the range from min_orderkey to max_set
  --
  SET @max_set = @min_orderkey + @range
  if @max_set > @max_orderkey
     SET @max_set = @max_orderkey + 1

  --
  --  Delete from ORDERS and LINEITEM table
  --
  DELETE_TRANS:
  begin transaction
    delete from ORDERS where O_ORDERKEY in
           (select * from #OLDORDERS WHERE
O_ORDERKEY >= @min_orderkey AND O_ORDERKEY <
@max_set)
    delete from LINEITEM where L_ORDERKEY in
           (select * from #OLDORDERS WHERE
O_ORDERKEY >= @min_orderkey AND O_ORDERKEY <
@max_set)

  commit transaction

  --
  --  If deadlock victim abort then restart the
transaction
  --
  if (@@error = 1205)
    BEGIN
      print 'Delete deadlock - restarting RF2
transaction'
      rollback transaction
      GOTO DELETE_TRANS
    END

  --
  --  Move min_orderkey to start of next delete set
  --
  SET @min_orderkey = @max_set

  END
END
```

```
GO
```

Appendix H Price Quotes



AUG 17 2000 17:53 FR MICROSOFT RECP #1 425 936 7329 TO 67329 P.02/03

One Microsoft Way
Redmond, WA 98052-6399

Fax 425 936 7329
http://www.microsoft.com/

Microsoft

August 17, 2000

Mr. Larry Kemp
Hewlett-Packard Corp.
14335 NE 24th St., Suite B-201
Bellevue, WA 98007

Dear Mr. Kemp:

Here is the information you requested regarding U.S. pricing for several Microsoft products, to be used in conjunction with TPC-H benchmark testing.

Part Number	Description	Price
810-00652	**SQL Server 2000 Enterprise Edition** Server license only Discount schedule: Open Program – No Level	$5,549
359-00532	**SQL Server 2000 Client License** 50 Client Licenses @ $146.00 each Discount schedule: Open Program – No Level	$7,300
C10-00475	**Windows 2000 Advanced Server** Server license only Discount schedule: Open Program – No Level	$2,399
659-00390	**Visual Studio Professional 6.0 Win32**	$1,079
	5-year maintenance for above software ($2095 per year)	$10,475

This quote is valid for the next 90 days.

If I can be of any further assistance, please contact me at (425) 705-9857 or kurtdan@microsoft.com.

Yours truly,

[illegible]

Kurt Daniel
Business Manager
SQL Server Marketing

Microsoft Corporation is an equal opportunity employer.

Software House International

Pricing Proposal

SHI Account Executive: Matthew Martin
Telephone: (800) 766-6357 ext. 106 Fax: (408) 232-2585

August 15, 2000 - Hewlett-Packard NSD TPC-H @ 100GB

Description	Part #	Qty	Price	Extended
HP NetServer LXr 8500 One Pentium III Xeon 550MHz 2MB L2 cache 256MB RAM, etc.	**D8543AV**	**1**	**$16,290**	**$16,290**
Intel Pentium III Xeon 550MHz 2Mbyte L2 processor upgrade	**D8531A**	**7**	**$5,590**	**$39,130**
HP NetServer LXr8500 Memory Carrier Card	**D7071A**	**1**	**$680**	**$680**
256MB Dimm for LXr 8500	**D9325A**	**15**	**$739**	**$11,085**
Adaptec SCSI Card 39160	**18223000**	**7**	**$319**	**$2,233**
HP Fiber Host Bus Adapter	**D8602A**	**1**	**$1,349**	**$1,349**
HP NetServer 10/100TX PCI LAN Adapter	**D5013A**	**1**	**$82**	**$82**
HP 9.1 GB 10K HotSwap Wide Ultra2 SCSI Disk	**D6107A**	**2**	**$430**	**$860**
HP 17in Display	**D2828A**	**1**	**$185**	**$185**
HP NetServer mini-DIN keyboard and mouse	**D4950B/C3751B**	**1**	**$79**	**$79**
HP Rack System/E33 (33 EIA units usable space)	**J1501A**	**2**	**$1,680**	**$3,360**
	Server Hardware Subtotal			**$75,333**
HP NetServer Rack Storage/12FC	**D5991A**	**1**	**6,159**	**$6,159**
HP Fibre Channel Controller	**D5990A**	**1**	**4,450**	**$4,450**
HP Fibre Channel Hub	**D6976A**	**1**	**3,130**	**$3,130**
HP NetServer Rack Storage/12	**D5989B**	**14**	**$1,890**	**$26,460**
HP 9GB, 10krpm Hot-swap disk module	**D6107A**	**180**	**$430**	**$77,400**
HP SCSI Cable 2.5m UDHTS 68/HDTS 68	**D6020A**	**14**	**$97**	**$1,358**
APC UPS	**588293**	**1**	**$1,725**	**$1,725**
			Storage Subtotal	**$120,682**

Quote Good for Ninety Days

Hewlett-Packard Company
3000 Hanover Street

August 19, 2000

Mr. Larry Gray
Re: HP NetServer LXr 8500

Hewlett-Packard is pleased to submit this formal quote to provide five years of HP SupportPack Hardware Maintenance Service for your HP NetServer LXr 8500 and concurrently purchased mass storage subsystem.

HP's support service provides these benefits for your business:

- **HP-trained service representatives**
- **Multiple coverage options from date of purchase**
- **Multiple options for hardware repair response times**
- **Technical assistance for installation, product configuration and setup, problem solving and normal operation on your HP product**
- **Five years of pre-paid support, purchased direct from HP**

A.1 Terms & Conditions

The following terms and conditions must be met for the SupportPack to be valid:

Appendix B: Required configuration

- HP NetServer LXr8500 with eight 550MHz 2MB L2 cache processors **(HP p/n D8543AV)**
- **One Fiber Channel RS/12FC (HP p/n D5991A) disk subsystem, populated with 12 9 GB disk drives (HP p/n D6107A)**
- **fourteen** HP RS/12 SCSI rack storage disk enclosures **(HP p/n D5989B), populated with 168 9 GB disk drives (HP p/n D6107A).**

B.1.1 Support level

This support provides HP's best possible response time during coverage hours of 8 am to 9 pm, Monday through Friday, except HP holidays. An HP Authorized Representative will arrive on-site and begin hardware maintenance service within 4 hours of the call receipt between 8 am and 5 pm local time. The 4 hour on-site response is available to sites within 100 miles of a major metropolitan areas. See chart below.

Distance from	Response Time 4-

Customer-designated Site to primary HP Support Office	hour Support
0-50 miles	**4 hours**
51-100 miles	**4 hours**
101-200 miles	**8 hours**
201-300 miles	*****
Over 300 miles	*****

This maintenance agreement is an upgrade to the three year warranty for your new system providing the response shown above with full HP parts replacement for the complete five year term.

This proposal does not include: consumables, user maintenance, non-HP Devices or, any product previously repaired by an unauthorized technician or user.

Your total cost for 5 years of hardware support is $29,500. This is for U.S. customers only. Payment is due upon purchase of the SupportPacks for the above products (a discount has been applied for advance payment, and must be purchased direct from HP at the time of hardware purchase).

The terms of this quotation are good for 90 days from today's date.

Approved by:
Hewlett-Packard North America Marketing Manager