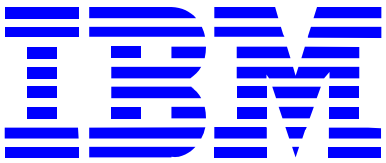


TPC Benchmark™ H Full Disclosure Report

IBM Power 595 Model 9119-FHA

Using

Sybase IQ Single Application Server Edition v.15.1 ESD #1.2

First Edition

November 24, 2009

The following terms used in this publication are trademarks of their respective companies as follows:

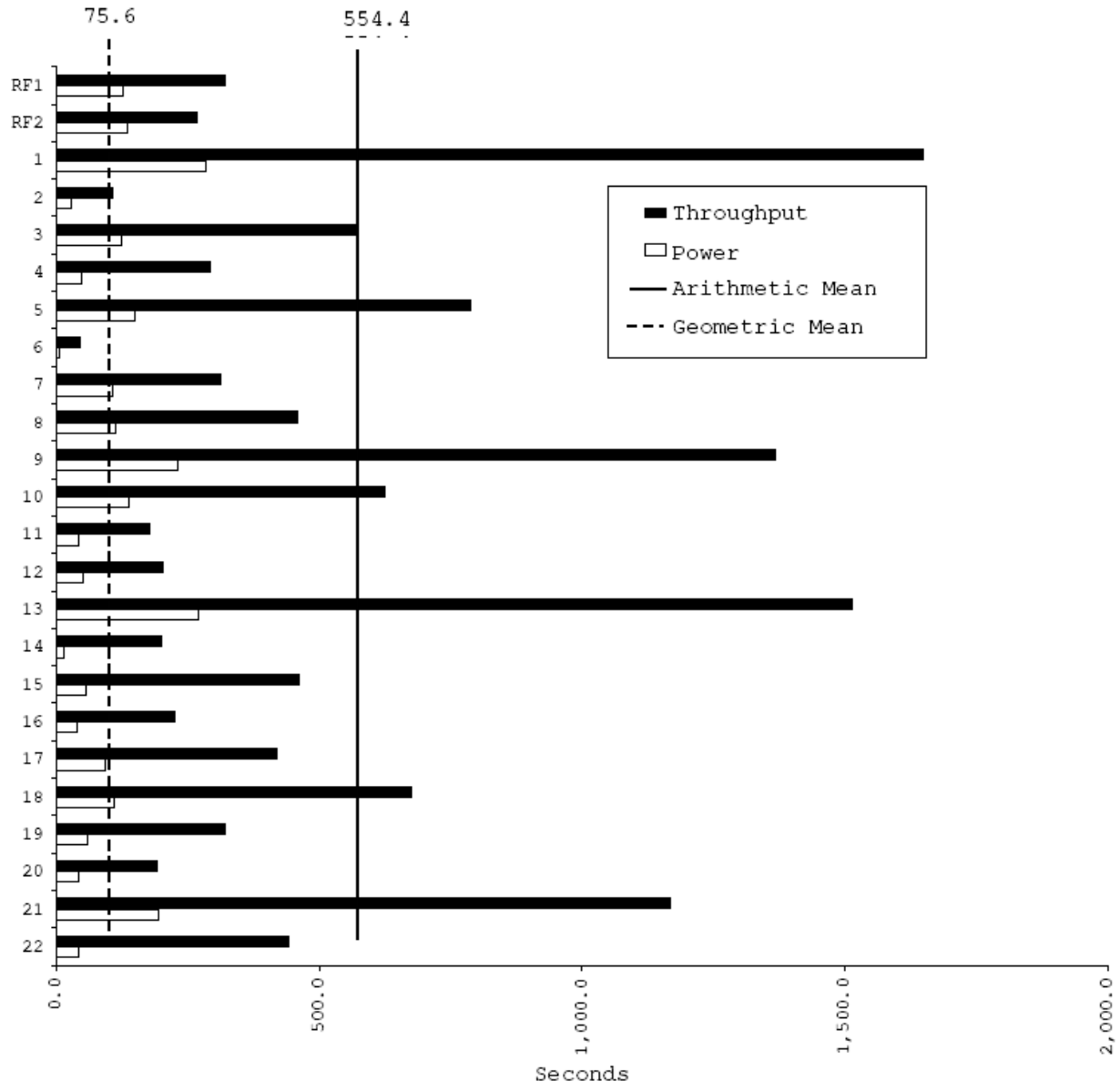
Trademark of the Transaction Processing Performance Council:
TPC Benchmark
TPC-H
QppH
QthH
QphH

Trademark of International Business Machines Corporation:
IBM
pSeries
POWER6
TotalStorage
AIX
AIX VERSION 6.1

Trademark of Sybase Inc:
Sybase IQ

First Edition November 24, 2009

	IBM Power 595 Model 9119-FHA			TPC-H Rev. 2.8.0 Report Date: November 24, 2009
Total System Cost	Composite Query per Hour Rating			Price/Performance
3,224,000 USD	156,537.3 QphH@3000GB			20.60 USD Price/QphH@3000GB
Database Size	Database Manager	Operating System	Other Software	Availability Date
3000GB	Sybase IQ Single Application Server Edition v.15.1 ESD #1.2	AIX V6.1	None	Now



Database Load Time: 10:02:25	Load included backup: N	Total Data Storage/Database Size: 6.58
RAID (base tables): Y	RAID (Base Tables and Auxiliary Data Structures): Y	RAID (All): Y

System Configurations

1 IBM Power 595 Model 9119-FHA

Processors	64 5.0GHz POWER6 Cores with 32MB L3 Cache per chip
Memory	512GB
Disk Controllers	24 x 4Gigabit Fibre Channel PCI-E Adapters & 12 IBM TotalStorage DS4800 dual controllers
Disk Drives	288 x73GB 15K 4Gb FC drives; 5 x 146.8GB SAS drives
Total Disk Storage	20,226.27 GB (GB is defined as 1024 * 1024 * 1024 bytes)

IBM Power 595
Model 9119-FHA

TPC-H Rev. 2.8.0

Report Date:
November 24, 2009

Description	Part No.	Source	Unit Price	Qty	Ext Price	Maint Price
Server Hardware						
Server 1:9119 Model 595	9119-FHA	1	91,000	1	91,000	63,192
GX Dual-port 12x HCA	1816	1	4,000	6	24,000	
2.5M Enhanced 12X Cable (to connect to a gx card in	1831	1	650	12	7,800	
146.8 GB Ultra320 15K RPM	3279	1	1,299	5	6,495	
0/8 - core POWER6 5.0GHz CoD, 0-core Active	4695	1	67,700	8	541,600	99,840
Single Processor Activation for FC 4695 (5.0 GHz)	4755	1	33,900	64	2,169,600	430,080
256GB Bundle DDR2 Activations	5681	1	83,200	2	166,400	
0/8GB DDR2 Memory (4x2GB) DIMMS - 667 MHz	5694	1	650	64	41,600	
2-PORT BASETX ETHT.PCI-X A	5767	1	699	1	699	
4.7 GB IDE Slimline DVD-RAM Drive	5757	1	660	1	660	
4 Gigabit Dual-port Fibre Channel PCI-E Adapter	5774	1	3,308	24	79,392	
PCI-X DDR Dual - x4 SAS Adapter	5912	1	1,092	1	1,092	
12xI/O Drawer PCI-E, SFF disk	5803	1	32,000	3	96,000	26,712
Bulk Power Regulator	6333	1	4,200	8	33,600	
Bulk Power Distribution Assembly	6334	1	2,500	4	10,000	
Slimline Doors (F&R), H CEC AND I/O Expansion Rac	6869	1	8,000	1	8,000	
UPIC Cable Group, BPD1 to I/O Drawer at A01	6941	1	500	1	500	
UPIC Cable Group, BPD1 to I/O Drawer A05	6942	1	500	1	500	
UPIC Cable Group, BPD1 to I/O Drawer A09	6943	1	500	1	500	
3rd & 4th AMDs + UPIC Y-Cable Group (BPC to Fans)	6952	1	5,000	1	5,000	
UPIC Y-Cable Group BPD1 to 1st Processor Node	6961	1	650	1	650	
UPIC Y-Cable Group BPD1 to 2nd Processor Node	6962	1	650	1	650	
UPIC Y-Cable Group BPD1 to 3rd Processor Node	6963	1	650	1	650	
UPIC Y-Cable Group BPD1 to 4th Processor Node	6964	1	650	1	650	
UPIC Y-Cable Group BPD1 to 5th Processor Node	6965	1	650	1	650	
UPIC Y-Cable Group BPD1 to 6th Processor Node	6966	1	650	1	650	
UPIC Y-Cable Group BPD1 to 7th Processor Node	6967	1	650	1	650	
UPIC Y-Cable Group BPD1 to 8th Processor Node	6968	1	650	1	650	
Ethernet Cable, 15M, Hardware Management	7802	1	34	2	68	
Line Cord, 6AWG/Type W, 14ft, IEC309 100A Plug	8686	1	2,500	4	10,000	
HMC 1:7042-C07 Desktop Hardw.Mgmt.Console	7042-C07	1	1,830	1	1,830	1,344
IBM T117 TFT 17 inch Color Display	3645	1	875	1	875	
Quiet Touch Keyboard - USB, Business Black,	5951	1	104	1	104	
Power Cord (6-foot), To Wall (125V, 15A), Plug Type #	6470	1	18	2	36	
Ethernet Cable, 6M, Hardware Management	7801	1	15	1	15	
Mouse - Business Black with Keyboard Attachment	8841	1	78	1	78	
PCI-e 1Gb Ethernet UTP 2Port	5767	1	699	1	699	
				Subtotal	**3,303,343**	**621,168**
Storage						
DS4800 Disk System Model 82 (4 GB Cache)	1815-82A	1	53,995	12	647,940	
DS4800 8-Storage Partitions	8870	1	10,000	12	120,000	
(22R4255) DS4800 AIX Host Kit	7711	1	7,000	12	84,000	
DS4000 EXP810 Enclosure	1812-81A	1	6,000	48	288,000	
Short Wave SFP	2410	1	998	96	95,808	
16-PAK 4 GBPS 73GB GB/15K	5433	1	14,816	18	266,688	
Fiber Cable 25m	5625	1	189	48	9,072	
Fiber Cable 1m	5601	1	79	96	7,584	
3 Year Warranty Service Upgrade 1812-81A 24x7x4		1	960	48	0	46,080
3 Year Warranty Service Upgrade 1815-82A 24x7x4		1	3,200	12	0	38,400
ServicePac for 3-Year 24x7x4 Support (Rack)	41L2760	1	300	5	0	1,500
IBM S2 42U Standard Rack	93074RX	1	1,489	5	7,445	
				Subtotal	**1,526,537**	**85,980**

IBM Power 595
Model 9119-FHA

TPC-H Rev. 2.8.0

Report Date:
November 24, 2009

Server Software						
AIX 6.1 (media only)	5692-A6P	1	50	1	50	
AIX Software per Processor	5765-G62	1	2,495	64	159,680	
AIX IBM Management Edition for AIX	5765-AME	1	490	64	31,360	
AIX Software Maintanance (3Y)	73-SM3-474	1	2,836	64		181,504
AIX Software Maintanance 24x7 Upgrade (3Y)	73-SM3-476	1	732	64		46,848
AIX IBM Mgmt Edition of AIX Software Maintanance	373-AME-989	1	196	64		12,544
HMC Software SUB (3Y)	5773-0570	1	236	1		236
HMC Software Support (3Y)	5773-0569	1	675	1		675
C for AIX user Lic+SW maint 12 MO	D5A1DLL	1	975	1	975	
C for AIX user annual SW maint renewal	E1A1FLL	1	195	2		390
				Subtotal	**192,065**	**242,197**
Third Party Hardware/Software						
SybaseIQ15 (64 processors)		2	2,595	64	166,080	
Sybase IQ15 (64 processors, 3 years)		2	1,713	64		109,632
				Subtotal	**166,080**	**109,632**
				Total	**5,188,025**	**1,058,977**
				Total IBM Discounts*		**-2,981,645**
				Sybase Discount (15%)	**-41,357**	

Three-Year Cost of Ownership	**3,224,000**
QphH@3000GB	**156,537**
$/QphH@3000GB	**20.60**

Notes:

For pricing details and contact information please see appendix E

Pricing Sources: 1) IBM 2) Sybase

*Discounts are based on US list prices for similar quantities & configurations including pre-payment for maintenance. The discount applies to the totality of all items with price source of "1".

Audited by: Francois Raab of Infosizing (www.sizing.com)

The system as configured for the test is available Now.

Prices used in TPC benchmarks reflect the actual prices a customer would pay for a one-time purchase of the stated components. Individually negotiated discounts are not permitted. Special prices base 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 the stated prices are not available according to these terms, please notify the TPC at pricing@tpc.org. Thank You.

Numerical Quantities Summary

Measurement Results

Database Scaling (SF/Size)	=	3000
Total Data Storage/Database Size	=	6.58
Start of Database Load	=	10/27/09 13:48:39
End of Database Load	=	10/27/09 23:51:04
Database Load Time	=	10:02:25
Query Streams for Throughput Test	=	9
TPC-H Power Metric (QppH@3000GB)	=	142,790.7
TPC-H Throughput Metric (QthH@3000GB)	=	171,607.4
Composite Query-per-Hour Rating (QphH@3000GB)	=	156,537.3
Total System Price over 3 years	=	3,224,000 USD
TPC-H Price/Performance Metric ($/QphH@3000GB)	=	20.60 USD

Measurement Intervals

Measurement Interval in Throughput Test (Ts) = 12,461 seconds

Duration of Stream Execution

		Date and Time Stamps			Duration
		Query Start	RF1 Start	RF2 Start	Queries
Stream ID	Seed Used	Query End	RF1 End	RF2 End	RFs
Stream 00	1027235104	10/28/2009 07:21:48.311	10/28/2009 07:19:38.526	10/28/2009 07:59:16.489	00:37:19.179
		10/28/2009 07:59:07.490	10/28/2009 07:21:45.390	10/28/2009 08:01:30.918	00:04:21.293
Stream 01	1027235105	10/28/2009 08:01:31.123	10/28/2009 08:02:31.664	10/28/2009 08:07:41.693	03:21:51.692
		10/28/2009 11:23:22.815	10/28/2009 08:07:41.079	10/28/2009 08:13:01.749	00:10:29.471
Stream 02	1027235106	10/28/2009 08:01:31.605	10/28/2009 08:13:03.055	10/28/2009 08:36:05.651	03:25:33.290
		10/28/2009 11:27:04.895	10/28/2009 08:18:34.126	10/28/2009 08:41:19.961	00:10:45.381
Stream 03	1027235107	10/28/2009 08:01:31.572	10/28/2009 08:41:20.872	10/28/2009 08:45:53.532	03:18:38.186
		10/28/2009 11:20:09.758	10/28/2009 08:45:52.902	10/28/2009 08:49:56.850	00:08:35.348
Stream 04	1027235108	10/28/2009 08:01:31.167	10/28/2009 09:01:22.017	10/28/2009 09:06:55.081	03:19:38.196
		10/28/2009 11:21:09.363	10/28/2009 09:06:54.570	10/28/2009 09:11:50.184	00:10:27.656
Stream 05	1027235109	10/28/2009 08:01:31.631	10/28/2009 09:11:50.999	10/28/2009 09:30:06.207	03:21:40.793
		10/28/2009 11:23:12.424	10/28/2009 09:17:57.633	10/28/2009 09:34:40.962	00:10:41.389
Stream 06	1027235110	10/28/2009 08:01:31.190	10/28/2009 09:34:42.171	10/28/2009 09:46:02.247	03:24:10.787
		10/28/2009 11:25:41.977	10/28/2009 09:40:05.915	10/28/2009 09:50:51.268	00:10:12.765
Stream 07	1027235111	10/28/2009 08:01:31.601	10/28/2009 09:50:51.813	10/28/2009 10:02:03.096	03:26:00.751
		10/28/2009 11:27:32.352	10/28/2009 09:56:25.916	10/28/2009 10:05:38.677	00:09:09.684
Stream 08	1027235112	10/28/2009 08:01:31.191	10/28/2009 10:11:07.498	10/28/2009 10:16:22.189	03:24:21.676
		10/28/2009 11:25:52.867	10/28/2009 10:16:21.588	10/28/2009 10:20:09.478	00:09:01.379
Stream 09	1027235113	10/28/2009 08:01:31.557	10/28/2009 10:37:35.733	10/28/2009 10:42:47.146	03:27:39.591
		10/28/2009 11:29:11.148	10/28/2009 10:42:46.485	10/28/2009 10:46:46.720	00:09:10.326

Timing Interval (in seconds)

Stream ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Stream 0	282.7	27.5	122.1	46.4	149.3	7.0	107.3	112.5	230.5	136.7	41.9	50.3
Stream 1	1,550.7	73.3	580.3	170.1	681.3	61.2	354.4	359.6	1,721.6	471.4	91.9	272.1
Stream 2	1,841.3	93.5	733.3	369.6	846.6	82.8	321.7	494.0	1,326.9	571.5	277.6	235.1
Stream 3	1,182.3	93.9	584.1	293.5	833.6	39.1	343.3	539.1	1,203.1	550.8	222.4	233.6
Stream 4	1,749.5	91.9	521.2	165.7	888.1	32.7	300.4	396.3	1,467.3	638.9	266.6	173.9
Stream 5	1,541.6	82.3	562.0	314.3	984.1	32.5	267.3	564.7	1,156.9	591.9	119.5	175.9
Stream 6	1,809.1	123.1	596.7	343.0	896.1	29.8	291.1	316.6	1,603.9	776.4	183.5	169.8
Stream 7	1,757.9	111.5	596.7	291.6	774.4	37.7	262.1	501.3	1,473.0	668.6	148.6	67.7
Stream 8	1,582.6	100.2	405.9	384.5	849.1	54.6	317.0	417.2	1,210.7	606.3	106.9	235.3
Stream 9	1,814.0	177.8	528.6	295.8	346.2	43.8	347.6	530.9	1,135.6	746.0	172.8	249.8
Minimum	1,182.3	73.3	405.9	165.7	346.2	29.8	262.1	316.6	1,135.6	471.4	91.9	67.7
Avgerage	1,647.7	105.3	567.6	292.0	788.8	46.0	311.7	457.7	1,366.6	624.6	176.6	201.5
Maximum	1,841.3	177.8	733.3	384.5	984.1	82.8	354.4	564.7	1,721.6	776.4	277.6	272.1

Stream ID	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream 0	269.9	14.3	55.7	40.3	94.1	111.0	60.3	42.5	193.5	43.5	126.9	134.4
Stream 1	1,472.5	251.3	496.7	247.0	287.7	725.4	268.8	237.5	1,422.0	314.9	309.4	320.1
Stream 2	1,458.9	189.1	487.4	249.9	354.6	711.0	342.4	219.7	628.4	498.2	331.1	314.3
Stream 3	1,479.8	178.5	560.1	271.7	243.3	904.0	183.8	165.1	1,377.1	436.1	272.0	243.3
Stream 4	1,424.1	192.3	374.3	267.4	280.3	581.6	198.3	214.8	1,208.8	543.8	332.6	295.1
Stream 5	1,560.1	386.9	474.0	192.3	238.8	676.3	218.8	228.3	1,366.0	366.4	366.6	274.8
Stream 6	1,308.9	154.7	462.8	222.8	488.6	587.8	215.0	174.4	882.5	614.3	323.7	289.0
Stream 7	1,695.4	104.3	204.9	194.1	249.5	760.6	866.4	183.7	1,154.4	256.2	334.1	215.6
Stream 8	1,556.1	218.2	584.7	325.1	775.7	520.7	361.9	153.2	1,185.5	310.2	314.1	227.3
Stream 9	1,661.6	113.3	495.6	42.1	841.8	617.4	226.8	148.1	1,287.4	636.7	310.8	239.6
Minimum	1,308.9	104.3	204.9	42.1	238.8	520.7	183.8	148.1	628.4	256.2	272.0	215.6
Avgerage	1,513.0	198.7	460.1	223.6	417.8	676.1	320.2	191.6	1,168.0	441.9	321.6	268.8
Maximum	1,695.4	386.9	584.7	325.1	841.8	904.0	866.4	237.5	1,422.0	636.7	366.6	320.1

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

Benchmark Sponsor: William Bostic
IBM Power Systems Performance
11501 Burnet Road
Austin, TX 78758

November 23, 2009

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

Platform: **IBM Power 595 Model 9119-FHA**

Database Manager: **Sybase IQ Single Application Server Edition v.15.1 ESD#1.2**

Operating System: **AIX V6.1**

The results were:

CPU (Speed)	Memory	Disks	QphH@3000GB
IBM Power 595 Model 9119-FHA			
32 x POWER6 Dual-Core (5.0GHz)	32 MB L3/chip 512 GB Main	288 x 73GB ext. 5 x 146 GB int.	**156,537.3**

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 3,000GB and populated accordingly
- The compliance of the database auxiliary data structures was verified
- The database load time was correctly measured and reported

- The required ACID properties were verified and met
- The query input variables were generated by QGEN
- The query text was produced using minor modifications 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 9 query streams
- The ratio between the longest and the shortest query was such that no query timings were adjusted
- The execution times for queries and refresh functions were correctly measured and reported
- The repeatability of the measured results was verified
- The required amount of database log was configured
- The system pricing was verified for major components and maintenance
- The major pages from the FDR were verified for accuracy

Additional Audit Notes:

None.

Respectfully Yours,

François Raab
President

Abstract

This report documents the full disclosure information required by the TPC Benchmark™ H Standard Specification Revision 2.8.0 dated September 11, 2008 for measurements on the IBM System Power 595 Model 9119-FHA.

The software used includes AIX V6.1 operating system with Sybase IQ Single Application Server Edition v.15.1 ESD #1.2.

Preface

TPC Benchmark™ H Standard Specification was developed by the Transaction Processing Performance Council (TPC). It was released on February 26, 1999, and most recently revised (Revision 2.8.0) on September 11, 2008. This is the full disclosure report for benchmark testing of the IBM Power 595 Model 9119-FHA according to the TPC Benchmark™ H Standard Specification.

TPC Benchmark™ H is a Decision Support benchmark. It is a suite of business oriented 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.

The TPC-H operations are modeled as follows:

- The database is continuously available 24 hours a day, 7 days a week, for ad-hoc queries from multiple end users and data modifications against all tables, except possibly during infrequent (e.g., once a month) maintenance sessions;
- The TPC-H database tracks, possibly with some delay, the state of the OLTP database through on-going refresh functions which batch together a number of modifications impacting some part of the decision support database;
- Due to the world-wide nature of the business data stored in the TPC-H database, the queries and the refresh functions may be executed against the database at any time, especially in relation to each other. In addition, this mix of queries and refresh functions is subject to specific ACIDity requirements, since queries and refresh functions may execute concurrently;

- To achieve the optimal compromise between performance and operational requirements, the database administrator can set, once and for all, the locking levels and the concurrent scheduling rules for queries and refresh functions.

The minimum database required to run the benchmark holds business data from 10,000 suppliers. It contains almost ten million rows representing a raw storage capacity of about 1 gigabyte. Compliant benchmark implementations may also use one of the larger permissible database populations (e.g., 100 gigabytes), as defined in Clause 4.1.3.

The performance metric reported by TPC-H is called the TPC-H Composite Query-per-Hour Performance Metric (QphH@Size), and reflects multiple aspects of the capability of the system to process queries. These aspects include the selected database size against which the queries are executed, the query processing power when queries are submitted by a single stream, and the query throughput when queries are submitted by multiple concurrent users. The TPC-H Price/Performance metric is expressed as $/QphH@Size. To be compliant with the TPC-H standard, all references to TPC-H results for a given configuration must include all required reporting components (see Clause 5.4.6). The TPC believes that comparisons of TPC-H results measured against different database sizes are misleading and discourages such comparisons.

The TPC-H database must be implemented using a commercially available database management system (DBMS) and the queries executed via an interface using dynamic SQL. The specification provides for variants of SQL, as implementers are not required to have implemented a specific SQL standard in full.

TPC-H uses terminology and metrics that are similar to other benchmarks, originated by the TPC and others. Such similarity in terminology does not in any way imply that TPC-H results are comparable to other benchmarks. The only benchmark results comparable to TPC-H are other TPC-H results compliant with the same revision.

Despite the fact that this benchmark offers a rich environment representative of many decision support systems, this benchmark does not reflect the entire range of decision support requirements. In addition, the extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-H approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments. Extrapolations to any other environment are not recommended.

Benchmark results are highly dependent upon workload, specific application requirements, and systems design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC-H should not be used as a substitute for a specific customer application benchmarking when critical capacity planning and/or product evaluation decisions are contemplated.

Benchmark sponsors are permitted several possible system designs, provided that they adhere to the model described in Clause 6. A full disclosure report (FDR) of the implementation details, as specified in Clause 8, must be made available along with the reported results.

1.0 General Items

1.1. Benchmark Sponsor

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

This benchmark was sponsored by International Business Machines Corporation.

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:

- *Data Base tuning options;*
- *Optimizer/Query execution options;*
- *Query Processing tool/language configuration parameters;*
- *Recovery/commit options;*
- *Consistency/locking options;*
- *Operating system and configuration parameters;*
- *Configuration parameters and options for any other software component incorporated into the pricing structure;*
- *Compiler optimization options.*

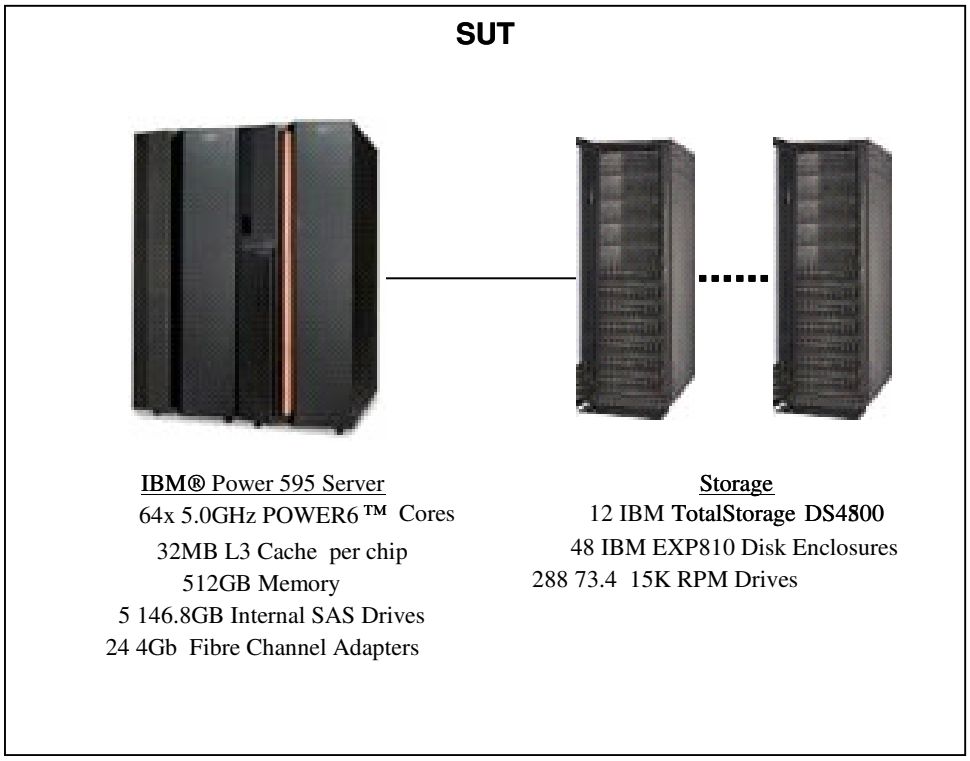
Appendix A "Tunable Parameters" contains a list of all SYBASE parameters and operating system parameters. Session initialization parameters can be set during or immediately after establishing the connection to the database within the tpcdbatch program documented in Appendix D "Driver Source Code". This result uses the default session initialization parameters established during preprocessing/binding of the tpcdbatch program.

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 and type of disk units (and controllers, if applicable)*
- *Number and type of disk units (and controllers, if applicable).*
- *Number of channels or bus connections to disk units, including the protocol type*
- *Number of LAN (e.g. Ethernet) connections, including routers, work stations, 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, middle-ware components, software drivers, etc.)

IBM Power 595 Model 9119-FHA Benchmark Configuration



The system was an IBM Power 595 Model 9119-FHA with

- 64 5.0GHz POWER6 Cores with 32MB L3 cache per chip
- 512 GB of memory
- 24 IBM 4Gb Dual-port Fibre Channel PCI-E adaptes
- 5 146.8GB 15K RPM Internal SAS disk drives
- 288 73.4GB 15K RPM external FC disk drives
- 48 EXP810 disk enclosures
- 12 IBM TotalStorage DS4800 dual controllers

The tested and priced configurations differ in disk capacity. For full details of the priced configuration, see the pricing spreadsheet in the Executive Summary.

2.0 Clause 2: Logical Database Design Related Items

Appendix B "Programs and Scripts" contains the programs and input files used to load the test database. The test and qualification databases use the same table definitions, indices and partitioning methods. Thus, the buildtpcd script documented in Appendix B was used for both the qualification and test databases except that different input files were used to define the tablespace devices and sizes.

2.1. 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 "Programs and Scripts" contains the table definitions and the programs to load the database.

2.2. Database Organization

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

Appendix B "Programs and Scripts" contains the DDL for the table and index definitions.

2.3. Horizontal Partitioning

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

Horizontal partitioning was used for lineitem and orders tables see Appendix B "Programs and Scripts".

2.4. Replication

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

No replication was used.

3.0 Clause 3: Queries and Refresh Functions

3.1. Query Language

The query language used to implement the queries must be identified (e.g.,"RALF/SQL-Plus").

SQL was the query language used.

3.2. Verification for the Random Number Generator

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

The supplied QGEN version 2.8.0 and DBGEN 2.8.0 were used.

3.3. Substitution Parameters

3.3.1. Method of 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. If QGEN is used, the version number, release number, modification number and patch level of QGEN must be disclosed.

The supplied QGEN version 2.8.0 was used to generate the substitution parameters.

3.4. Query Text

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

Appendix C.1 "Qualification Query Output" contains the output for each of the queries.

The functional query definitions and variants used in this disclosure use the following minor query modifications.

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

3.5. Query Substitution Parameters and Seeds

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

Appendix C2, "Query Substitution Parameters" contains the query substitution parameters used in the performance tests.

3.6. Isolation Level

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

The isolation level used to run the queries was repeatable read.

3.7. Refresh Functions

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

The refresh functions are part of the implementation specific layer/driver code included in Appendix B.

4.0 Clause 4: Database System Properties

The results of the ACID tests must be disclosed along with a description of how the ACID requirements were met. This includes disclosing the code written to implement the ACID Transaction and Query.

All ACID tests were conducted according to specification. The Atomicity, Isolation, Consistency and Durability tests were performed on the IBM Power 595 Model 9119-FHA. Appendix E. "Acid Transaction Source Code" contains the source code for the ACID transaction and query.

4.1. Atomicity Requirements

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.

4.1.1. Atomicity of 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.

The following steps were performed to verify the atomicity of completed transactions:

1. The total price from the ORDER table and the extended price from the LINEITEM table were retrieved for a random Orderkey. The number of records in the HISTORY table was also retrieved.
2. The ACID transaction T1 was executed for the Orderkey used in Step 1.
3. The ACID transaction committed.
4. The total price and the extended price were retrieved for the same orderkey used in step 1 and step 2. It was verified that: T1.EXTENDEDPRICE = OLD.EXTENDEDPRICE + ((T1.DELTA) * (OLD.EXTENDEDPRICE/OLD.QUANTITY)), T1.TOTALPRICE = OLD.TOTALPRICE + ((T1.EXTENDEDPRICE-OLD.EXTENDEDPRICE)*(1-DISCOUNT)*(1+TAX)), and that the number of records in the history table had increased by 1.

4.1.2. Atomicity of Aborted Transactions

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.

The following steps were performed to verify the atomicity of the aborted ACID transaction:

1. The total price from the ORDER table and the extended price from the LINEITEM table were retrieved for a random Orderkey. The number of records in the HISTORY table was also retrieved.
2. The ACID transaction was executed for the Orderkey used in step 1.
3. The transaction was rolled back.
4. The total price and the extended price were retrieved for the same orderkey used in step 1 and step 2. It was verified that the extended price and the total price were the same as in step 1. The

number of records in the HISTORY table was retrieved again and verified to be the same as in step1.

4.2. Consistency Requirements

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

4.2.1. Consistency Condition

A consistent state for the TPC-H database is defined to exist when:

O_TOTALPRICE = SUM(L_EXTENDEDPRICE(1-L_DISCOUNT)*(1+L_TAX)*

for each ORDER and LINEITEM defined by (O_ORDERKEY=L_ORDERKEY)

The following queries were executed before and after a measurement to show that the database was always in a consistent state both initially and after a measurement. Check the implementation if

```
SELECT sum ("truncate" ("truncate"( round(cast(l_extendedprice as numeric(26,16)),2) *
        (1 - round(cast(l_discount as numeric(26,16)),2)),2)
    * (1 + round(cast(l_tax as numeric(26,16)),2)) , 2))
    FROM lineitem WHERE l_orderkey = o_key;
```

is equal to o_total of order table.

4.2.2. Consistency Tests

Verify that the ORDER and LINEITEM tables are initially consistent as defined in Clause 4.3.2.1, based on a random sample of at least 10 distinct values of O_ORDERKEY.

The queries defined in 4.3.2, "Consistency Condition" were run after initial database build and prior to executing the ACID transaction. The queries showed that the database is in a consistent state.

After executing 22 streams of 100 ACID transactions each, the queries defined in 4.3.2, "Consistency Condition" were run again. The queries showed that the database was still in a consistent state.

4.3. Isolation Requirements

4.3.1. Isolation Test 1

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

The following steps were performed to satisfy the test of isolation for a read-only and a read-write committed transaction:

1. 1st session: Started an ACID transaction with a randomly selected O_KEY, L_KEY and DELTA. The transaction was delayed for 10 seconds just prior to the Commit.
2. 2nd session: Started an ACID query for the same O_KEY as in the ACID transaction. The query completed without blocking and did not see any of the uncommitted changes made by the ACID transaction.
3. 1st session: the ACID transaction resumed and successfully completed the Commit.

4.3.2. Isolation Test 2

This test demonstrates isolation for the read-write conflict of read-write transaction and read-only transaction when the read-write transaction is rolled back.

The following steps were performed to satisfy the test of isolation for read-only and a rolled back read-write transaction:

1. 1st session: Performed the ACID transaction for a random O_KEY, L_KEY and DELTA. The transaction was delayed for 10 seconds just prior to the Rollback.
2. 2nd session: Started an ACID query for the same O_KEY as in the ACID transaction. The query completed without blocking and did not see any of the uncommitted changes made by the ACID transaction.
3. 1st session: the ACID transaction resumed and successfully completed the Rollback.

4.3.3. Isolation Test 3

This test demonstrates isolation for the write-write conflict of two refresh transactions when the first transaction is committed.

The following steps were performed to verify isolation of two refresh transactions:

1. 1st session: Started an ACID transaction T1 for a randomly selected O_KEY, L_KEY and DELTA. The transaction was delayed for 30 seconds just prior to the COMMIT.
2. 2nd session: Started a second ACID transaction T2 for the same O_KEY, L_KEY, and for a randomly selected DELTA2. This transaction was forced to wait.
3. 1st session: The ACID transaction T1 was released and the Commit was executed, releasing the record. With the LINEITEM record now released, the ACID transaction T2 completed.
4. Verified that:

   T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE +
   (DELTA*(T1.L_EXTENDEDPRICE)/T1.L_QUANTITY)

4.3.4. Isolation Test 4

This test demonstrates isolation for write-write conflict of two ACID transactions when the first transaction is rolled back.

The following steps were performed to verify the isolation of two ACID transactions after the first one is rolled back:

1. 1st session: Started an ACID transaction T1 for a randomly selected O_KEY, L_KEY, and DELTA. The transaction was delayed for 30 seconds just prior to the rollback.
2. 2nd session: Started a second ACID transaction T2 for the same O_KEY, L_KEY used by the 1st session. This transaction was forced to wait.
3. 1st session: Rollback the ACID transaction T1. With the LINEITEM record now released, the ACID transaction T2 completed.
4. Verified that T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE

4.3.5. Isolation Test 5

This test demonstrates the ability of read and writes transactions affecting different database tables to make progress concurrently.

1. 1st session: Started an ACID transaction, T1, for a randomly selected O_KEY, L_KEY and DELTA. The ACID transaction was suspended prior to COMMIT.
2. 2nd session: Started a second ACID transaction, T2, which selected random values of PS_PARTKEY and PS_SUPPKEY and returned all columns of the PARTSUPP table for which PS_PARTKEY and PS_SUPPKEY were equal to the selected values.
3. T2 completed.
4. T1 was allowed to complete.
5. It was verified that the appropriate rows in the ORDERS, LINEITEM and HISTORY tables have been changed.

4.3.6. Isolation Test 6

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

1. 1st session: A transaction T1, which executeed TPC-H query 1 with a randomly selected DELTA was started.
2. 2nd session: Before T1 completed, an ACID transaction T2, with randomly selected values of O_KEY, L_KEY and DELTA, was started.
3. T2 completed and appropriate rows in the ORDERS, LINEITEM and HISTORY tables had been changed.
4. T1 completed executing query 1.

4.4. Durability Requirements

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

4.4.1. Permanent Failure of Single Durable Medium and Loss of System Power

These tests were combined and conducted on the qualification database. The following steps were performed:

1. The consistency test described in section 4.2.1 was verified.
2. The current count of the total number of records in the HISTORY table was determined giving hist1.
3. A test to run 200 ACID transactions on each of 10 execution streams was started such that each stream executes a different set of transactions.

4. One of the disks containing the SYBASE transaction log recovery data, database table data, and database index was removed from the enclosure after at least 100 ACID transactions had completed from each of the execution streams.
5. Because the disks were in RAID 1 configuration the applications continued running the ACID transactions.
6. A controller which was on the preferred path of database log data, database table data, and database index was removed, the preferred path automatically switched to another controller and the applications continued running the ACID transactions.
7. The system was shutdown by switching off the power for all system components.
8. The system was powered back on and rebooted, and the database was restarted.
9. Step 2 was performed giving hist2. It was verified that hist2 - hist1 was greater than or equal to the number of records in the success file.
10. Consistency condition described in 4.3.2 was verified.

4.4.2. Failure of Storage Controller, and Loss of System Power

The test was combined with 4.4.1.

5.0 Clause 5: Scaling and Database Population Related Items

5.1. 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.

The following table contains the TPC Benchmark™ H defined tables and the number of rows for each table as they existed upon build completion:

Table	Rows
Lineitem	18,000,048,306
Orders	4,500,000,000
Customer	450,000,000
Supplier	30,000,000
Part	600,000,000
Partsupp	2,400,000,000
Nation	25
Region	5

5.2. Distribution of Tables and Logs

The distribution of tables and logs across all media must be explicitly depicted for the tested and priced systems.

Sybase IQ was configured on an IBM Power 595 Model 9119-FHA server. The system had

- 24 IBM 4Gb Fibre Channel PCI-E adapters
- 288 73.4GB external disk drives
- 5 146.8GB internal drives

The IBM Power 595 Model 9119-FHA had 12 IBM TotalStorage DS4800 dual controllers. 12 RAID-1 arrays were configured in each DS4800. Each of the controllers was accessed through two FC adapters.

Permanent tables, their auxiliary data structures, database logs and temp dbspaces resided in tablespaces created on the RAID-1 arrays. See Appendix B "Programs and Scripts".

The Operating System resided on an internal disk. The database software resided on an internal disk.

5.3. Mapping of Database Partitions/Replications

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

The database was not replicated. Lineitem and orders are partitioned by year.

5.4. Implementation of RAID

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 used must be disclosed for each device.

RAID level 1 was used for database tables, indexes, and recovery logs.

5.5. DBGEN Modifications

The version number, release number, modification number, and patch level of DBGEN must be disclosed. 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 standard distribution of DBGEN version 2.8.0 was used.

5.6. Database Loading

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

The database load time was 10:02:25.

5.7. Data Storage Ratio

The data storage ratio must be disclosed. It is computed by dividing the total data storage of the priced configuration (expressed in GB) by the size chosen for the test database as defined in clause 4.1.3.1. The ratio must be reported to the nearest 1/100th, rounded up.

The calculation of the data storage ratio is shown in the following table:

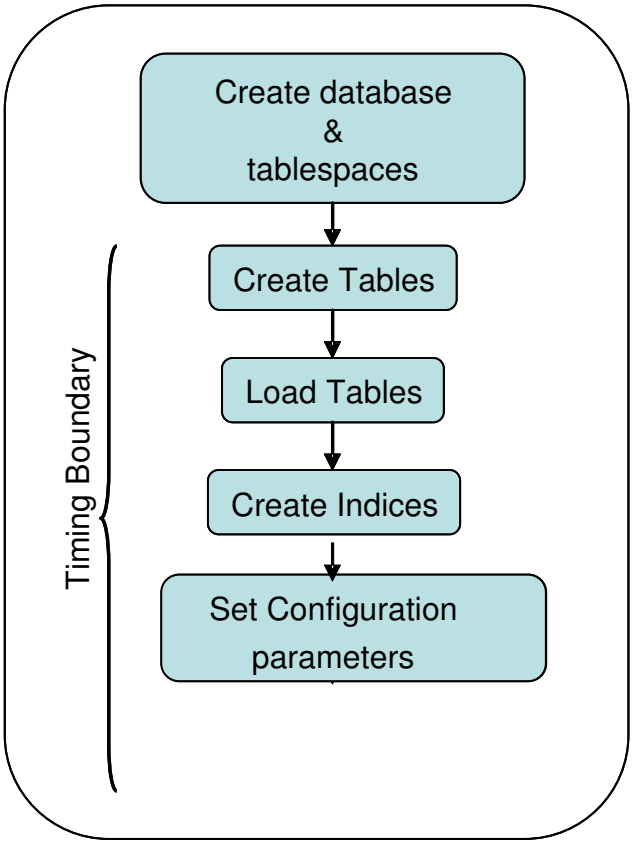
Disk Type	Number of Disks	Space per Disk	Sub-Total Disk Space	Database Size	Data Storage Ratio
146GB	5	136.23 GB	681.15GB		
4Gb FC 73.4GB	288	67.865 GB	19,545.12 GB		
Total			20,226.27 GB	3,000 GB	6.58

5.8. Details of Database Loading

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

Flat files for each of the tables were created using DBGEN. Appendix B "Programs and Scripts" contains the programs and input files used to load the database.

Database Load Procedure:



6.0 Clause 6: 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.

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. Start database server
2. RF1 Refresh Transaction
3. Stream 00 Execution
4. 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 refresh functions must be reported for the power test.

See Numerical Quantities Summary in the Executive Summary.

6.4. Number of Streams for the Throughput Test

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

See Numerical Quantities Summary in the Executive Summary.

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.

See Numerical Quantities Summary in the Executive Summary.

6.6. Total Elapsed Time for the Measurement Interval

The total elapsed time of the measurement interval (see Clause 5.3.6) must be reported for the throughput test.

See Numerical Quantities Summary in the Executive Summary.

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

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

See Numerical Quantities Summary in the Executive Summary

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

The timing intervals (see Clause 5.3.7) for each query of each stream and for each refresh function must be reported for the throughput test.

See Numerical Quantities Summary in the Executive Summary.

6.9. Performance Metrics

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

See Numerical Quantities Summary in the Executive Summary.

6.10. The Performance Metric and Numerical Quantities from Both Runs

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

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

	QppH@3000GB	QthH@3000GB	QphH@3000GB
Run 1	142,790.7	171,607.4	156,537.3
Run 2	142,600.2	172,479.4	156,829.8
% Difference	0.13%	0.51%	0.19%

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.

7.0 Clause 7: SUT and Driver Implementation

7.1. Driver

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

Appendix D "Driver Source Code" contains the source code used for the driver and all scripts used in connection with it.

The power test and throughput test are invoked by calling run_tpch with the scale factor, the scope, and the total number of query streams specified. For the power test the power stream 0 SQL script will be executed along with the refresh functions SQL scripts. Then the throughput test followed with the specified query streams to be run, along with the refresh function streams.

7.2. Implementation Specific Layer

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

The performance tests are performed using dbisqlc and iqisql. dbisqlc and iqisql are Sybase-provided utilities which allow SQL statements to be executed against a Sybase IQ database. Both dbisqlc and iqisql utilities are invoked from the command line on the SUT. They read input from files containing SQL statements and sends results to stdout. dbisqlc uses information in the .odbc.ini file to connect to the database while iqisql uses information in interfaces file for the same.

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

7.3. Profile-Directed Optimization

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

Profile-directed optimization was not used.

8.0 Clause 8: Pricing-Related Items

8.1. Hardware and Software Used

A detailed list of hardware and software used in the priced system must be reported. Each item must have a vendor part number, description, and release/revision level, and indicate General Availability (see Clause 7.2.2.1) either implicitly or explicitly(omitted Availability Dates default to the System Availability Date). If package pricing is used, contents of the package must be disclosed. Pricing source(s) and effective date(s) of price(s) must also be reported.

The detailed list of all hardware and software for the priced configuration is listed in the Executive Summary.

8.2. Three Year Cost of System Configuration

The total 3-year price of the entire configuration must be reported, including: hardware, software, hardware maintenance, and software support charges. Separate component pricing is required (see Clause 7.3.1. Pricing Spreadsheet.) Hardware maintenance and software support must be reported separately. The software support level must be disclosed separately from that of hardware, with separate pricing and discounts.

The price sheet for this disclosure is contained in the executive summary pages.

The pricing spreadsheet includes maintenance costs for 3 years. This service provides 7 days per week, 24 hours per day coverage.

Discounts are based on US list prices and for similar quantities and configurations. A discount of 49.95% has been applied to all IBM hardware, software, and services based on the total value and quantities of the components of the configuration, including full payment of all components and maintenance.

The prices listed for the IBM software products includes software support that provides the items identified in paragraph 7.1.5.6 of the TPC-H Benchmark Specification.

For assistance with any of these prices or their applicability to any customer's requirements, please contact one of the following individuals:

Dan Hebrank, IBM Sales & Distribution, Systems & Technology Sales

email: dhebrank@us.ibm.com phone 1-314-283-4674

Das Joydeep, Sybase Inc.

email: Joydeep.Das@sybase.com phone 1-925-236-5214

8.3. System Availability Date

The System Availability Date (see Clause 7.2.2.1) must be the single availability date reported on the first page of the executive summary. 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

7.3.1.4). All Availability Dates required to be reported must be disclosed to a precision of 1 day, but the precise format is left to the test sponsor.

The System Availability Date is Now.

9.0 Clause 9: Audit Items

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

The auditor's attestation letter is included at the front of this report.

Appendix - A Tunable Parameters

A.1 Sybase IQ Database Configuration

tpch.cfg

```
# tpch.cfg
-n  tpch_15_0_sybase
-x  tcpip{port=5788}

# The following parameters are also found in the configuration file
# $ASDIR/scripts/default.cfg.  Any parameters not specified below
# and not in the start up parameter list, will be added by start_asiq
# using default.cfg as a guide.

-c 64M
-gd all
-gm 25
-gc 5000
-gr 5000
-gp 4096
-tl 0
-iqmt 2000
-iqmc 190000
-iqtc 120000
-iqtss 240
-iqnumbercpus 100
-iqpartition 64
-iqgovern 10
```

options.sql

```
SET OPTION "PUBLIC".STRING_RTRUNCATION='Off';
SET OPTION "PUBLIC".Allow_Nulls_By_Default='Off';
SET OPTION "PUBLIC".Append_Load='On';
SET OPTION "PUBLIC".Force_No_Scroll_Cursors='On';
SET OPTION "PUBLIC".Garray_Fill_Factor_Percent=3;
SET OPTION "PUBLIC".Load_Memory_Mb=0;
SET OPTION "PUBLIC".Max_IQ_Threads_Per_Connection=500;
SET OPTION "PUBLIC".Minimize_Storage='On';
SET OPTION "PUBLIC".Notify_Modulus=10000000;
SET OPTION "PUBLIC".Row_Counts='On';
SET OPTION "PUBLIC".Sweeper_Threads_Percent=8;
SET OPTION "PUBLIC".Wash_Area_Buffers_Percent = '20';
SET OPTION "PUBLIC".Prefetch_Threads_Percent = 15;
SET OPTION "PUBLIC".Max_Hash_Rows=18000000;
SET OPTION PUBLIC.Default_Having_Selectivity_PPM = 40;
SET OPTION "PUBLIC".Hash_Thrashing_Percent=100;
SET OPTION "PUBLIC".QUERY_TEMP_SPACE_LIMIT=0;
SET OPTION "PUBLIC".subquery_flattening_preference=3;
SET OPTION "PUBLIC".Max_Query_Parallelism=100;
SET OPTION PUBLIC.FP_PREDICATE_WORKUNIT_PAGES=50;
SET OPTION PUBLIC.ROW_PREFETCH_SIZE=40;
```

.odbc.ini

```
[ODBC Data Source]
SybaseIQ=Sybase IQ Driver

[tpch_15_0]
Userid=DBA
Password=sql
EngineName=tpch_15_0_sybase
DatabaseName=tpch
DatabaseFile=tpch.db
CommLinks=tcpip(host=tpch2;port=5788)
AutoStop=no

[test_15_0]
Userid=DBA
Password=sql
EngineName=test_15_0_sybase
DatabaseName=iqdemo
DatabaseFile=iqdemo.db
CommLinks=tcpip(host=tpch2;port=5788)
AutoStop=no
```

A.2 AIX Parameters

```
chdev -l sys0 -a maxuproc=8000
ioo -r -o lvm_bufcnt=16
LDR_CNTRL=TEXTPSIZE=64K@DATAPSIZE=64K@STACKPSIZE=
64K
```

Appendix - B Programs and Scripts

B.1 create_database.sql

```
CREATE DATABASE '/testdb/tpch.db'
TRANSACTION LOG ON
COLLATION 'ISO_BINENG'
CASE RESPECT
PAGE SIZE 4096
BLANK PADDING ON
JAVA ON
JCONNECT ON
IQ PATH '/testdb/M001'
IQ PAGE SIZE 524288
TEMPORARY PATH '/testdb/T001';
```

B.2 create_dbspaces_15_0.sh

```
--
================================================
===============
-- Notation:
-- IQ MAIN DBSPACES will be named as iq[0-9]+
-- and
-- IQ TEMP DBSPACES will be named as iqtmp[0-9]+
--
-- This notation is required to help in reporting the main
-- and temp dbspaces in the kit.
--
================================================
===============
alter dbspace IQ_SYSTEM_MAIN ADD
FILE iq2 '/testdb/M002' ,
FILE iq3 '/testdb/M003',
FILE iq4 '/testdb/M004',
FILE iq5 '/testdb/M005',
FILE iq6 '/testdb/M006',
FILE iq7 '/testdb/M007',
FILE iq8 '/testdb/M008',
FILE iq9 '/testdb/M009',
FILE iq10 '/testdb/M010',
FILE iq11 '/testdb/M011',
FILE iq12 '/testdb/M012',
FILE iq13 '/testdb/M013',
FILE iq14 '/testdb/M014',
FILE iq15 '/testdb/M015',
FILE iq16 '/testdb/M016',
FILE iq17 '/testdb/M017',
FILE iq18 '/testdb/M018',
FILE iq19 '/testdb/M019',
FILE iq20 '/testdb/M020',
FILE iq21 '/testdb/M021',
FILE iq22 '/testdb/M022',
FILE iq23 '/testdb/M023',
FILE iq24 '/testdb/M024',
FILE iq25 '/testdb/M025',
FILE iq26 '/testdb/M026',
FILE iq27 '/testdb/M027',
FILE iq28 '/testdb/M028',
FILE iq29 '/testdb/M029',
FILE iq30 '/testdb/M030',
FILE iq31 '/testdb/M031',
FILE iq32 '/testdb/M032',
FILE iq33 '/testdb/M033',
FILE iq34 '/testdb/M034',
FILE iq35 '/testdb/M035',
FILE iq36 '/testdb/M036',
FILE iq37 '/testdb/M037',
FILE iq38 '/testdb/M038',
FILE iq39 '/testdb/M039',
FILE iq40 '/testdb/M040',
FILE iq41 '/testdb/M041',
FILE iq42 '/testdb/M042',
FILE iq43 '/testdb/M043',
FILE iq44 '/testdb/M044',
FILE iq45 '/testdb/M045',
FILE iq46 '/testdb/M046',
FILE iq47 '/testdb/M047',
FILE iq48 '/testdb/M048',
FILE iq49 '/testdb/M049',
FILE iq50 '/testdb/M050',
FILE iq51 '/testdb/M051',
FILE iq52 '/testdb/M052',
FILE iq53 '/testdb/M053',
FILE iq54 '/testdb/M054',
FILE iq55 '/testdb/M055',
FILE iq56 '/testdb/M056',
FILE iq57 '/testdb/M057',
FILE iq58 '/testdb/M058',
FILE iq59 '/testdb/M059',
FILE iq60 '/testdb/M060',
FILE iq61 '/testdb/M061',
FILE iq62 '/testdb/M062',
FILE iq63 '/testdb/M063',
FILE iq64 '/testdb/M064',
FILE iq65 '/testdb/M065',
FILE iq66 '/testdb/M066',
FILE iq67 '/testdb/M067',
FILE iq68 '/testdb/M068',
FILE iq69 '/testdb/M069',
FILE iq70 '/testdb/M070',
FILE iq71 '/testdb/M071',
FILE iq72 '/testdb/M072',
FILE iq73 '/testdb/M073',
FILE iq74 '/testdb/M074',
FILE iq75 '/testdb/M075',
FILE iq76 '/testdb/M076',
FILE iq77 '/testdb/M077',
FILE iq78 '/testdb/M078',
FILE iq79 '/testdb/M079',
FILE iq80 '/testdb/M080',
FILE iq81 '/testdb/M081',
FILE iq82 '/testdb/M082',
FILE iq83 '/testdb/M083',
FILE iq84 '/testdb/M084',
FILE iq85 '/testdb/M085',
FILE iq86 '/testdb/M086',
FILE iq87 '/testdb/M087',
FILE iq88 '/testdb/M088',
FILE iq89 '/testdb/M089',
FILE iq90 '/testdb/M090',
FILE iq91 '/testdb/M091',
FILE iq92 '/testdb/M092',
FILE iq93 '/testdb/M093',
FILE iq94 '/testdb/M094',
FILE iq95 '/testdb/M095',
FILE iq96 '/testdb/M096',
FILE iq97 '/testdb/M097',
FILE iq98 '/testdb/M098',
FILE iq99 '/testdb/M099',
FILE iq100 '/testdb/M100',
FILE iq101 '/testdb/M101',
FILE iq102 '/testdb/M102',
FILE iq103 '/testdb/M103',
```

```
FILE iq104 '/testdb/M104',
FILE iq105 '/testdb/M105',
FILE iq106 '/testdb/M106',
FILE iq107 '/testdb/M107',
FILE iq108 '/testdb/M108',
FILE iq109 '/testdb/M109',
FILE iq110 '/testdb/M110',
FILE iq111 '/testdb/M111',
FILE iq112 '/testdb/M112',
FILE iq113 '/testdb/M113',
FILE iq114 '/testdb/M114',
FILE iq115 '/testdb/M115',
FILE iq116 '/testdb/M116',
FILE iq117 '/testdb/M117',
FILE iq118 '/testdb/M118',
FILE iq119 '/testdb/M119',
FILE iq120 '/testdb/M120',
FILE iq121 '/testdb/M121',
FILE iq122 '/testdb/M122',
FILE iq123 '/testdb/M123',
FILE iq124 '/testdb/M124',
FILE iq125 '/testdb/M125',
FILE iq126 '/testdb/M126',
FILE iq127 '/testdb/M127',
FILE iq128 '/testdb/M128',
FILE iq129 '/testdb/M129',
FILE iq130 '/testdb/M130',
FILE iq131 '/testdb/M131',
FILE iq132 '/testdb/M132',
FILE iq133 '/testdb/M133',
FILE iq134 '/testdb/M134',
FILE iq135 '/testdb/M135',
FILE iq136 '/testdb/M136',
FILE iq137 '/testdb/M137',
FILE iq138 '/testdb/M138',
FILE iq139 '/testdb/M139',
FILE iq140 '/testdb/M140',
FILE iq141 '/testdb/M141',
FILE iq142 '/testdb/M142',
FILE iq143 '/testdb/M143',
FILE iq144 '/testdb/M144' ;

alter dbspace IQ_SYSTEM_TEMP ADD
FILE iqtmp2 '/testdb/T002',
FILE iqtmp3 '/testdb/T003',
FILE iqtmp4 '/testdb/T004',
FILE iqtmp5 '/testdb/T005',
FILE iqtmp6 '/testdb/T006',
FILE iqtmp7 '/testdb/T007',

FILE iqtmp8 '/testdb/T008',
FILE iqtmp9 '/testdb/T009',
FILE iqtmp10 '/testdb/T010',
FILE iqtmp11 '/testdb/T011',
FILE iqtmp12 '/testdb/T012',
FILE iqtmp13 '/testdb/T013',
FILE iqtmp14 '/testdb/T014',
FILE iqtmp15 '/testdb/T015',
FILE iqtmp16 '/testdb/T016',
FILE iqtmp17 '/testdb/T017',
FILE iqtmp18 '/testdb/T018',
FILE iqtmp19 '/testdb/T019',
FILE iqtmp20 '/testdb/T020',
FILE iqtmp21 '/testdb/T021',
FILE iqtmp22 '/testdb/T022',
FILE iqtmp23 '/testdb/T023',
FILE iqtmp24 '/testdb/T024',
FILE iqtmp25 '/testdb/T025',
FILE iqtmp26 '/testdb/T026',
FILE iqtmp27 '/testdb/T027',
FILE iqtmp28 '/testdb/T028',
FILE iqtmp29 '/testdb/T029',
FILE iqtmp30 '/testdb/T030',
FILE iqtmp31 '/testdb/T031',
FILE iqtmp32 '/testdb/T032',
FILE iqtmp33 '/testdb/T033',
FILE iqtmp34 '/testdb/T034',
FILE iqtmp35 '/testdb/T035',
FILE iqtmp36 '/testdb/T036',
FILE iqtmp37 '/testdb/T037',
FILE iqtmp38 '/testdb/T038',
FILE iqtmp39 '/testdb/T039',
FILE iqtmp40 '/testdb/T040',
FILE iqtmp41 '/testdb/T041',
FILE iqtmp42 '/testdb/T042',
FILE iqtmp43 '/testdb/T043',
FILE iqtmp44 '/testdb/T044',
FILE iqtmp45 '/testdb/T045',
FILE iqtmp46 '/testdb/T046',
FILE iqtmp47 '/testdb/T047',
FILE iqtmp48 '/testdb/T048',
FILE iqtmp49 '/testdb/T049',
FILE iqtmp50 '/testdb/T050',
FILE iqtmp51 '/testdb/T051',
FILE iqtmp52 '/testdb/T052',
FILE iqtmp53 '/testdb/T053',
FILE iqtmp54 '/testdb/T054',
FILE iqtmp55 '/testdb/T055',
FILE iqtmp56 '/testdb/T056',
FILE iqtmp57 '/testdb/T057',

FILE iqtmp58 '/testdb/T058',
FILE iqtmp59 '/testdb/T059',
FILE iqtmp60 '/testdb/T060',
FILE iqtmp61 '/testdb/T061',
FILE iqtmp62 '/testdb/T062',
FILE iqtmp63 '/testdb/T063',
FILE iqtmp64 '/testdb/T064',
FILE iqtmp65 '/testdb/T065',
FILE iqtmp66 '/testdb/T066',
FILE iqtmp67 '/testdb/T067',
FILE iqtmp68 '/testdb/T068',
FILE iqtmp69 '/testdb/T069',
FILE iqtmp70 '/testdb/T070',
FILE iqtmp71 '/testdb/T071',
FILE iqtmp72 '/testdb/T072',
FILE iqtmp73 '/testdb/T073',
FILE iqtmp74 '/testdb/T074',
FILE iqtmp75 '/testdb/T075',
FILE iqtmp76 '/testdb/T076',
FILE iqtmp77 '/testdb/T077',
FILE iqtmp78 '/testdb/T078',
FILE iqtmp79 '/testdb/T079',
FILE iqtmp80 '/testdb/T080',
FILE iqtmp81 '/testdb/T081',
FILE iqtmp82 '/testdb/T082',
FILE iqtmp83 '/testdb/T083',
FILE iqtmp84 '/testdb/T084',
FILE iqtmp85 '/testdb/T085',
FILE iqtmp86 '/testdb/T086',
FILE iqtmp87 '/testdb/T087',
FILE iqtmp88 '/testdb/T088',
FILE iqtmp89 '/testdb/T089',
FILE iqtmp90 '/testdb/T090',
FILE iqtmp91 '/testdb/T091',
FILE iqtmp92 '/testdb/T092',
FILE iqtmp93 '/testdb/T093',
FILE iqtmp94 '/testdb/T094',
FILE iqtmp95 '/testdb/T095',
FILE iqtmp96 '/testdb/T096',
```

```
FILE iqtmp97 '/testdb/T097',
FILE iqtmp98 '/testdb/T098',
FILE iqtmp99 '/testdb/T099',
FILE iqtmp100 '/testdb/T100',
FILE iqtmp101 '/testdb/T101',
FILE iqtmp102 '/testdb/T102',
FILE iqtmp103 '/testdb/T103',
FILE iqtmp104 '/testdb/T104',
FILE iqtmp105 '/testdb/T105',
FILE iqtmp106 '/testdb/T106',
FILE iqtmp107 '/testdb/T107',
FILE iqtmp108 '/testdb/T108',
FILE iqtmp109 '/testdb/T109',
FILE iqtmp110 '/testdb/T110',
FILE iqtmp111 '/testdb/T111',
FILE iqtmp112 '/testdb/T112',
FILE iqtmp113 '/testdb/T113',
FILE iqtmp114 '/testdb/T114',
FILE iqtmp115 '/testdb/T115',
FILE iqtmp116 '/testdb/T116',
FILE iqtmp117 '/testdb/T117',
FILE iqtmp118 '/testdb/T118',
FILE iqtmp119 '/testdb/T119',
FILE iqtmp120 '/testdb/T120',
FILE iqtmp121 '/testdb/T121',
FILE iqtmp122 '/testdb/T122',
FILE iqtmp123 '/testdb/T123',
FILE iqtmp124 '/testdb/T124',
FILE iqtmp125 '/testdb/T125',
FILE iqtmp126 '/testdb/T126',
FILE iqtmp127 '/testdb/T127',
FILE iqtmp128 '/testdb/T128',
FILE iqtmp129 '/testdb/T129',
FILE iqtmp130 '/testdb/T130',
FILE iqtmp131 '/testdb/T131',
FILE iqtmp132 '/testdb/T132',
FILE iqtmp133 '/testdb/T133',
FILE iqtmp134 '/testdb/T134',
FILE iqtmp135 '/testdb/T135',
FILE iqtmp136 '/testdb/T136',
FILE iqtmp137 '/testdb/T137',
FILE iqtmp138 '/testdb/T138',
FILE iqtmp139 '/testdb/T139',
FILE iqtmp140 '/testdb/T140',
FILE iqtmp141 '/testdb/T141',
FILE iqtmp142 '/testdb/T142',
FILE iqtmp143 '/testdb/T143',
FILE iqtmp144 '/testdb/T144' ;
```

B.3 create_tables.sql

```
set temporary option on_error = 'continue';
set temporary option Allow_nulls_by_default='on';

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

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


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


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


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

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


CREATE TABLE orders
    (
    o_orderkey           unsigned bigint ,
    o_custkey            unsigned int ,
    o_orderstatus        char(1) ,
```

```
    o_totalprice          double precision ,
    o_orderdate            date  ,
    o_orderpriority       char(15) ,
    o_clerk              char(15) ,
    o_shippriority         int ,
    o_comment              varchar(79) ,
    primary key (o_orderkey)
    )
    partition by range (o_orderdate)
    ( p1 values <= ('1992-12-31'),
      p2 values <= ('1993-12-31'),
      p3 values <= ('1994-12-31'),
      p4 values <= ('1995-12-31'),
      p5 values <= ('1996-12-31'),
      p6 values <= ('1997-12-31'),
      p7 values <= ('1998-12-31'),
      p8 values <= (MAX)
    );

CREATE TABLE lineitem
    (
    l_orderkey             unsigned bigint ,
    l_partkey             unsigned int ,
    l_suppkey              unsigned int ,
    l_linenumber           int  ,
    l_quantity            double precision  ,
    l_extendedprice        double precision  ,
    l_discount            double precision  ,
    l_tax                double precision  ,
    l_returnflag          char(1) ,
    l_linestatus          char(1) ,
    l_shipdate            date ,
    l_commitdate           date ,
    l_receiptdate         date ,
    l_shipinstruct        char(25) ,
    l_shipmode             char(10) ,
    l_comment              varchar(44)
    )
partition by range(l_shipdate)
    ( p1 values <= ('1992-12-31'),
      p2 values <= ('1993-12-31'),
      p3 values <= ('1994-12-31'),
      p4 values <= ('1995-12-31'),
      p5 values <= ('1996-12-31'),
      p6 values <= ('1997-12-31'),
      p7 values <= ('1998-12-31'),
      p8 values <= (MAX)
    );

CREATE DATE INDEX o_orderdate_date ON orders(o_orderdate) ;
CREATE DATE INDEX l_shipdate_date ON lineitem(l_shipdate) ;
CREATE DATE INDEX l_receiptdate_date ON lineitem(l_receiptdate);

create HG index n_regionkey_hg on nation ( n_regionkey );
create HG index s_nationkey_hg on supplier ( s_nationkey );
create HG index c_nationkey_hg on customer ( c_nationkey );
create HG index o_custkey_hg on orders ( o_custkey );
create HG index l_partkey_hg on lineitem ( l_partkey );

create HG index ps_suppkey_hg on partsupp ( ps_suppkey );
create HG index ps_partkey_hg on partsupp ( ps_partkey );

create HG index l_suppkey_hg on lineitem ( l_suppkey );

create HG index l_orderkey_hg on lineitem ( l_orderkey );

commit;
```

B.4 create_refresh_functions.sql

```
--------Create RF1----------

CREATE PROCEDURE DBA.tpch_rf1 (IN data_directory var-
char(128),
                    IN stream_number varchar(3))
ON EXCEPTION RESUME
BEGIN
  DECLARE sql_cmd long varchar;
  DECLARE c_lf varchar(2);
  DECLARE rf1_start timestamp;
  DECLARE rf1_stop timestamp;
  DECLARE n_seconds numeric(16,5);
  DECLARE cur_sqlstate CHAR(5);

  SET c_lf=char(10);
  SET rf1_start = dateformat(now(*) ,'yyyy-Mm-Dd hh:nn:ss.sss');
  SELECT 'Stream'||stream_number||' RF1 START TIME --,
'||dateformat(rf1_start,'yyyy-mm-dd hh:nn:ss.sss');

  SET sql_cmd='load table orders ( '+c_lf;
  SET sql_cmd=sql_cmd+' o_orderkey '+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' o_custkey '+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' o_orderstatus '+char(39)+'|'+char(39)+',
'+c_lf;
  SET sql_cmd=sql_cmd+' o_totalprice '+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' o_orderdate date('+char(39)+'YYYY-MM-
DD'+char(39)+'), filler(1), '+c_lf;
  SET sql_cmd=sql_cmd+' o_orderpriority '+char(39)+'|'+char(39)+',
'+c_lf;
  SET sql_cmd=sql_cmd+' o_clerk '+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' o_shippriority
'+char(39)+'|'+char(39)+','+c_lf;
  SET sql_cmd=sql_cmd+' o_comment '+char(39)+'|'+char(39)+' )
'+c_lf;
  SET sql_cmd=sql_cmd+'from
'+char(39)+data_directory+'orders.tbl.u'+stream_number+char(39)+c_l
f;
  SET sql_cmd=sql_cmd+'row delimited by
'+char(39)+'\\x0a'+char(39)+' quotes off escapes off preview on;';
  EXECUTE IMMEDIATE with quotes on sql_cmd;
  SELECT SQLSTATE INTO cur_sqlstate;
  IF cur_sqlstate != '00000' THEN
    ROLLBACK;
    RAISERROR 23002 'RF1 failed while inserting into orders with
SQLSTATE: ', cur_sqlstate;
    RETURN(1);
  END IF;

--------insert into lineitem------------
  SET sql_cmd='load table lineitem ( '+c_lf;
  SET sql_cmd=sql_cmd+' l_orderkey '+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' l_partkey '+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' l_suppkey '+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' l_linenumber '+char(39)+'|'+char(39)+',
'+c_lf;
  SET sql_cmd=sql_cmd+' l_quantity '+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' l_extendedprice '+char(39)+'|'+char(39)+',
'+c_lf;
  SET sql_cmd=sql_cmd+' l_discount '+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' l_tax '+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' l_returnflag '+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' l_linestatus '+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' l_shipdate date('+char(39)+'YYYY-MM-
DD'+char(39)+'), filler(1), '+c_lf;
```

```
  SET sql_cmd=sql_cmd+' l_commitdate date('+char(39)+'YYYY-MM-
DD'+char(39)+'), filler(1), '+c_lf;
  SET sql_cmd=sql_cmd+' l_receiptdate date('+char(39)+'YYYY-MM-
DD'+char(39)+'), filler(1), '+c_lf;
  SET sql_cmd=sql_cmd+' l_shipinstruct '+char(39)+'|'+char(39)+',
'+c_lf;
  SET sql_cmd=sql_cmd+' l_shipmode '+char(39)+'|'+char(39)+', '+c_lf;
  SET sql_cmd=sql_cmd+' l_comment '+char(39)+'|'+char(39)+' )'+c_lf;
  SET sql_cmd=sql_cmd+'from
'+char(39)+data_directory+'lineitem.tbl.u'+stream_number+char(39)+c
_lf;
  SET sql_cmd=sql_cmd+'row delimited by
'+char(39)+'\\x0a'+char(39)+c_lf+'quotes off escapes off preview on;';
  EXECUTE IMMEDIATE with quotes on sql_cmd;

  SELECT SQLSTATE INTO cur_sqlstate;
  IF cur_sqlstate != '00000' THEN
    rollback;
    RAISERROR 23002 'RF1 failed while inserting into lineitem with
SQLSTATE: ', cur_sqlstate;
    RETURN(1);
  END IF;
  COMMIT;
  SET rf1_stop = dateformat(now(*) ,'yyyy-Mm-Dd hh:nn:ss.sss');
  SELECT 'Stream'||stream_number||' RF1 STOP TIME --,
'||dateformat(rf1_stop ,'yyyy-mm-dd hh:nn:ss.sss');

---------Find the execution time of rf1.---------------
  SET n_seconds=cast(datediff(millisecond,rf1_start,rf1_stop) AS nu-
meric(16,5))/1000;
  SET sql_cmd='Stream_'+stream_number+' RF1 Elapsed time --,
'+cast(n_seconds AS varchar(20))+ ' seconds' ;
  SELECT sql_cmd;
  RETURN(0);
END;

---------Create RF2---------------------------
CREATE PROCEDURE DBA.tpch_rf2 (in data_directory varchar(128),
                               in stream_number varchar(3))
ON exception resume
BEGIN
  DECLARE sql_cmd long varchar;
  DECLARE c_lf varchar(2);
  DECLARE rf2_start timestamp;
  DECLARE rf2_stop timestamp;
  DECLARE n_seconds numeric(16,5);
  DECLARE cur_sqlstate CHAR(5);

  SET c_lf=char(10);
  SET rf2_start = dateformat(now(*) ,'yyyy-Mm-Dd hh:nn:ss.sss');
  SELECT 'Stream'||stream_number||' RF2 START TIME --,
'||dateformat(rf2_start,'yyyy-mm-dd hh:nn:ss.sss');

  CREATE TABLE #delete_table ( d_orderkey UNSIGNED BIGINT );

  SET sql_cmd='load table #delete_table  (d_orderkey
'+char(39)+'|'+char(39)+')'+c_lf;
  SET sql_cmd=sql_cmd+'from
'+char(39)+data_directory+'delete.'+stream_number+char(39)+c_lf;
  SET sql_cmd=sql_cmd+'row delimited by
'+char(39)+'\\x0a'+char(39)+c_lf;
  SET sql_cmd=sql_cmd+'quotes off '+c_lf;
  SET sql_cmd=sql_cmd+'escapes off; '+c_lf;

  EXECUTE IMMEDIATE with quotes on sql_cmd;
  SELECT SQLSTATE INTO cur_sqlstate;
  IF cur_sqlstate != '00000' THEN
    ROLLBACK;
    SET sql_cmd='RF2 failed at Step 1 with SQLSTATE: '+cur_sqlstate;
    RAISERROR 23002 sql_cmd;
    RETURN(1);
  END IF;

-------- Delete from LineItem---------
  DELETE lineitem FROM lineitem
     WHERE l_orderkey in (select d_orderkey from  #delete_table);
  SELECT SQLSTATE INTO cur_sqlstate;

  IF cur_sqlstate != '00000' THEN
    ROLLBACK;
    SET sql_cmd='RF2 failed at Step 2 with SQLSTATE: '+cur_sqlstate;
    RAISERROR 23002 sql_cmd;
    RETURN(1);
  END IF;

-----Delete fom Orders.----------
  DELETE orders FROM orders
     WHERE o_orderkey in (select d_orderkey from  #delete_table);
  SELECT SQLSTATE INTO cur_sqlstate;

  IF cur_sqlstate != '00000' THEN
    ROLLBACK;
    SET sql_cmd='RF2 failed at Step 3 with SQLSTATE: '+cur_sqlstate;
    RAISERROR 23002 sql_cmd;
    RETURN(1);
  END IF;

  COMMIT;
  DROP TABLE #delete_table;
  SET rf2_stop = dateformat(now(*) ,'yyyy-Mm-Dd hh:nn:ss.sss');
  SELECT 'Stream'||stream_number||' RF2 STOP TIME --,
'||dateformat(rf2_stop,'yyyy-mm-dd hh:nn:ss.sss');
--------- Calculate execution time for rf2.-------

  SET n_seconds=cast(datediff(millisecond,rf2_start,rf2_stop) as nu-
meric(16,5))/1000;
  SET sql_cmd='Stream_'+stream_number+' RF2 Elapsed time --,
'+cast(n_seconds as varchar(20))+ ' seconds' ;
  SELECT sql_cmd;
  RETURN(0);
END;
```

B.5 load_region.sql

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

B.6 load_nation.sql

```
LOAD TABLE NATION (
N_NATIONKEY                    '|',
N_NAME                         '|',
N_REGIONKEY                    '|',
N_COMMENT                      '|'
)
FROM '/rawdata/nation.tbl'
```

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

B.7 load_customer.sql

```
----------------------------------------------------------------

LOAD TABLE CUSTOMER (
C_CUSTKEY                               '|',
C_NAME                          '|',
C_ADDRESS                               '|',
C_NATIONKEY                             '|',
C_PHONE                         '|',
C_ACCTBAL                               '|',
C_MKTSEGMENT                            '|',
C_COMMENT                               '|'
)
FROM '/rawdata/customer.tbl.1','/rawdata/customer.tbl.2',
    '/rawdata/customer.tbl.3','/rawdata/customer.tbl.4',
    '/rawdata/customer.tbl.5','/rawdata/customer.tbl.6',
    '/rawdata/customer.tbl.7','/rawdata/customer.tbl.8',
    '/rawdata/customer.tbl.9','/rawdata/customer.tbl.10',
    '/rawdata/customer.tbl.11','/rawdata/customer.tbl.12'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
```

B.8 load_part.sql

```
LOAD TABLE PART (
P_PARTKEY                               '|',
P_NAME                                  '|',
P_MFGR                                  '|',
P_BRAND                                 '|',
P_TYPE                                  '|',
P_SIZE                                  '|',
P_CONTAINER                             '|',
P_RETAILPRICE                           '|',
P_COMMENT                               '|'
)
FROM '/rawdata/part.tbl.1','/rawdata/part.tbl.2',
    '/rawdata/part.tbl.3','/rawdata/part.tbl.4',
    '/rawdata/part.tbl.5','/rawdata/part.tbl.6',
    '/rawdata/part.tbl.7','/rawdata/part.tbl.8',
    '/rawdata/part.tbl.9','/rawdata/part.tbl.10',
    '/rawdata/part.tbl.11','/rawdata/part.tbl.12'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
```

B.9 load_supplier.sql

```
LOAD TABLE SUPPLIER (
S_SUPPKEY                               '|',
S_NAME                                  '|',
S_ADDRESS                               '|',
S_NATIONKEY                             '|',
S_PHONE                                 '|',
S_ACCTBAL                               '|',
S_COMMENT                               '|'
)
FROM '/rawdata/supplier.tbl.1','/rawdata/supplier.tbl.2',
    '/rawdata/supplier.tbl.3','/rawdata/supplier.tbl.4',
    '/rawdata/supplier.tbl.5','/rawdata/supplier.tbl.6',
    '/rawdata/supplier.tbl.7','/rawdata/supplier.tbl.8',

    '/rawdata/supplier.tbl.9','/rawdata/supplier.tbl.10',
    '/rawdata/supplier.tbl.11','/rawdata/supplier.tbl.12'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
```

B.10 load_partsupp.sql

```
LOAD TABLE PARTSUPP (
PS_PARTKEY                              '|',
PS_SUPPKEY                              '|',
PS_AVAILQTY                             '|',
PS_SUPPLYCOST                           '|',
PS_COMMENT                              '|'
)
FROM '/rawdata/partsupp.tbl.1','/rawdata/partsupp.tbl.2',
    '/rawdata/partsupp.tbl.3','/rawdata/partsupp.tbl.4',
    '/rawdata/partsupp.tbl.5','/rawdata/partsupp.tbl.6',
    '/rawdata/partsupp.tbl.7','/rawdata/partsupp.tbl.8',
    '/rawdata/partsupp.tbl.9','/rawdata/partsupp.tbl.10',
    '/rawdata/partsupp.tbl.11','/rawdata/partsupp.tbl.12',
    '/rawdata/partsupp.tbl.13','/rawdata/partsupp.tbl.14',
    '/rawdata/partsupp.tbl.15','/rawdata/partsupp.tbl.16',
    '/rawdata/partsupp.tbl.17','/rawdata/partsupp.tbl.18',
    '/rawdata/partsupp.tbl.19','/rawdata/partsupp.tbl.20',
    '/rawdata/partsupp.tbl.21','/rawdata/partsupp.tbl.22',
    '/rawdata/partsupp.tbl.23','/rawdata/partsupp.tbl.24',
    '/rawdata/partsupp.tbl.25','/rawdata/partsupp.tbl.26',
    '/rawdata/partsupp.tbl.27','/rawdata/partsupp.tbl.28',
    '/rawdata/partsupp.tbl.29','/rawdata/partsupp.tbl.30',
    '/rawdata/partsupp.tbl.31','/rawdata/partsupp.tbl.32',
    '/rawdata/partsupp.tbl.33','/rawdata/partsupp.tbl.34',
    '/rawdata/partsupp.tbl.35','/rawdata/partsupp.tbl.36'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
```

B.11 load_orders.sql

```
LOAD TABLE ORDERS (
O_ORDERKEY                              '|',
O_CUSTKEY                               '|',
O_ORDERSTATUS                           '|',
O_TOTALPRICE                            '|',
O_ORDERDATE                             '|',
O_ORDERPRIORITY                         '|',
O_CLERK                                 '|',
O_SHIPPRIORITY                          '|',
O_COMMENT                               '|'
```

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

B.12 load_lineitem.sql

```
LOAD TABLE LINEITEM (
L_ORDERKEY                              '|',
L_PARTKEY                       '|',
L_SUPPKEY                               '|',
L_LINENUMBER                            '|',
L_QUANTITY                              '|',
L_EXTENDEDPRICE                         '|',
L_DISCOUNT                              '|',
L_TAX                           '|',
L_RETURNFLAG                            '|',
L_LINESTATUS                            '|',
L_SHIPDATE                              '|',
L_COMMITDATE                            '|',
L_RECEIPTDATE                           '|',
L_SHIPINSTRUCT                          '|',
L_SHIPMODE                              '|',
L_COMMENT                               '|'
)
FROM '/rawdata/lineitem.tbl.1','/rawdata/lineitem.tbl.2',
    '/rawdata/lineitem.tbl.3','/rawdata/lineitem.tbl.4',
    '/rawdata/lineitem.tbl.5','/rawdata/lineitem.tbl.6',
    '/rawdata/lineitem.tbl.7','/rawdata/lineitem.tbl.8',
    '/rawdata/lineitem.tbl.9','/rawdata/lineitem.tbl.10',
    '/rawdata/lineitem.tbl.11','/rawdata/lineitem.tbl.12',
    '/rawdata/lineitem.tbl.13','/rawdata/lineitem.tbl.14',
    '/rawdata/lineitem.tbl.15','/rawdata/lineitem.tbl.16',
    '/rawdata/lineitem.tbl.17','/rawdata/lineitem.tbl.18',
    '/rawdata/lineitem.tbl.19','/rawdata/lineitem.tbl.20',
    '/rawdata/lineitem.tbl.21','/rawdata/lineitem.tbl.22',
    '/rawdata/lineitem.tbl.23','/rawdata/lineitem.tbl.24',
    '/rawdata/lineitem.tbl.25','/rawdata/lineitem.tbl.26',
    '/rawdata/lineitem.tbl.27','/rawdata/lineitem.tbl.28',
    '/rawdata/lineitem.tbl.29','/rawdata/lineitem.tbl.30',
    '/rawdata/lineitem.tbl.31','/rawdata/lineitem.tbl.32',
    '/rawdata/lineitem.tbl.33','/rawdata/lineitem.tbl.34',
    '/rawdata/lineitem.tbl.35','/rawdata/lineitem.tbl.36',
    '/rawdata/lineitem.tbl.37','/rawdata/lineitem.tbl.38',
    '/rawdata/lineitem.tbl.39','/rawdata/lineitem.tbl.40',
    '/rawdata/lineitem.tbl.41','/rawdata/lineitem.tbl.42',
    '/rawdata/lineitem.tbl.43','/rawdata/lineitem.tbl.44',
    '/rawdata/lineitem.tbl.45','/rawdata/lineitem.tbl.46',
    '/rawdata/lineitem.tbl.47','/rawdata/lineitem.tbl.48',
    '/rawdata/lineitem.tbl.49','/rawdata/lineitem.tbl.50',
    '/rawdata/lineitem.tbl.51','/rawdata/lineitem.tbl.52',
    '/rawdata/lineitem.tbl.53','/rawdata/lineitem.tbl.54',
    '/rawdata/lineitem.tbl.55','/rawdata/lineitem.tbl.56',
    '/rawdata/lineitem.tbl.57','/rawdata/lineitem.tbl.58',
    '/rawdata/lineitem.tbl.59','/rawdata/lineitem.tbl.60',
    '/rawdata/lineitem.tbl.61','/rawdata/lineitem.tbl.62',
    '/rawdata/lineitem.tbl.63','/rawdata/lineitem.tbl.64',
    '/rawdata/lineitem.tbl.65','/rawdata/lineitem.tbl.66',
    '/rawdata/lineitem.tbl.67','/rawdata/lineitem.tbl.68',
    '/rawdata/lineitem.tbl.69','/rawdata/lineitem.tbl.70',
    '/rawdata/lineitem.tbl.71','/rawdata/lineitem.tbl.72'
escapes off
quotes off
row delimited by '\x0a'
WITH CHECKPOINT ON;
commit;
checkpoint;
commit;
```

B.13 update_power.sql

```
-- THIS IS A GENERATED FILE.
-- PLEASE CHANGE gen_update_power.sh FOR MODIFICATIONS
create variable c_path varchar(128);
create variable lineitem_start bigint;
create variable lineitem_end bigint;
create variable lineitem_diff bigint;
create variable orders_start bigint;
create variable orders_end bigint;
create variable orders_diff bigint;
go


set c_path='/testdb/data/'
go

set lineitem_start=(select count(*) from lineitem);
set orders_start=(select count(*) from orders);
go
call tpch_rf1 (c_path,1)
go
set lineitem_end=(select count(*) from lineitem);
set orders_end=(select count(*) from orders);
set lineitem_diff = lineitem_end - lineitem_start;
set orders_diff = orders_end - orders_start;
select 'In lineitem Rf1 Inserted ',lineitem_diff;
select 'In orders Rf1 Inserted ',orders_diff;
go
-- Sleep Until the query stream completes
select 'Stream 0 RF WAITING -- ,', dateformat (now(*) ,'yyyy-Mm-Dd
hh:nn:ss.sss')
go
!!check_stream.sh
--Stream execution is completed.

set lineitem_start=(select count(*) from lineitem);
go
call tpch_rf2 (c_path,1)
go
set lineitem_end=(select count(*) from lineitem);
set lineitem_diff = lineitem_start - lineitem_end;
select 'In lineitem Rf2 Deleted ',lineitem_diff;
go
```

B.14 update_throughput.sql

```
-- THIS IS A GENERATED FILE.
-- PLEASE CHANGE gen_update_throughput.sh FOR
MODIFICATIONS
create variable c_path varchar(128);
create variable qstart timestamp;
create variable qstop timestamp;
create variable n_seconds numeric(16,5);
create variable lineitem_start bigint;
create variable lineitem_end bigint;
create variable lineitem_diff bigint;
create variable orders_start bigint;
create variable orders_end bigint;
create variable orders_diff bigint;
go


set qstart = now(*)
set c_path='/testdb/data/'
select @@servername, db_name()
go
select 'Throughput Stream1 starts';

set lineitem_start=(select count(*) from lineitem);
set orders_start=(select count(*) from orders);
go
call tpch_rf1 (c_path,2)
go
set lineitem_end=(select count(*) from lineitem);
set orders_end=(select count(*) from orders);
set lineitem_diff = lineitem_end - lineitem_start;
set orders_diff = orders_end - orders_start;
select 'In lineitem Rf1 Inserted ',lineitem_diff;
select 'In orders Rf1 Inserted ',orders_diff;
go
commit
go
tpch_wait
go

set lineitem_start=(select count(*) from lineitem);
go
call tpch_rf2 (c_path,2)
go
commit
go
set lineitem_end=(select count(*) from lineitem);
set lineitem_diff = lineitem_start - lineitem_end;
select 'In lineitem Rf2 Deleted ',lineitem_diff;
go
tpch_wait
go
select 'Throughput Stream1 ends';
go

select 'Throughput Stream2 starts';

set lineitem_start=(select count(*) from lineitem);
set orders_start=(select count(*) from orders);
go
call tpch_rf1 (c_path,3)
go
set lineitem_end=(select count(*) from lineitem);
set orders_end=(select count(*) from orders);
set lineitem_diff = lineitem_end - lineitem_start;
set orders_diff = orders_end - orders_start;
select 'In lineitem Rf1 Inserted ',lineitem_diff;
select 'In orders Rf1 Inserted ',orders_diff;
go
commit
go
tpch_wait
go

set lineitem_start=(select count(*) from lineitem);
go
call tpch_rf2 (c_path,3)
go
commit
go
set lineitem_end=(select count(*) from lineitem);
set lineitem_diff = lineitem_start - lineitem_end;
select 'In lineitem Rf2 Deleted ',lineitem_diff;
go
tpch_wait
go
select 'Throughput Stream2 ends';
go

select 'Throughput Stream3 starts';

set lineitem_start=(select count(*) from lineitem);
set orders_start=(select count(*) from orders);
go
call tpch_rf1 (c_path,4)
go
set lineitem_end=(select count(*) from lineitem);
set orders_end=(select count(*) from orders);
set lineitem_diff = lineitem_end - lineitem_start;
set orders_diff = orders_end - orders_start;
select 'In lineitem Rf1 Inserted ',lineitem_diff;
select 'In orders Rf1 Inserted ',orders_diff;
go
commit
go
tpch_wait
go

set lineitem_start=(select count(*) from lineitem);
go
call tpch_rf2 (c_path,4)
go
commit
go
set lineitem_end=(select count(*) from lineitem);
set lineitem_diff = lineitem_start - lineitem_end;
select 'In lineitem Rf2 Deleted ',lineitem_diff;
go
tpch_wait
go
select 'Throughput Stream3 ends';
go

select 'Throughput Stream4 starts';

set lineitem_start=(select count(*) from lineitem);
set orders_start=(select count(*) from orders);
go
call tpch_rf1 (c_path,5)
go
set lineitem_end=(select count(*) from lineitem);
set orders_end=(select count(*) from orders);
set lineitem_diff = lineitem_end - lineitem_start;
set orders_diff = orders_end - orders_start;
select 'In lineitem Rf1 Inserted ',lineitem_diff;
```

```
select 'In orders Rf1 Inserted ',orders_diff;
go
commit
go
tpch_wait
go

set lineitem_start=(select count(*) from lineitem);
go
call tpch_rf2 (c_path,5)
go
commit
go
set lineitem_end=(select count(*) from lineitem);
set lineitem_diff = lineitem_start - lineitem_end;
select 'In lineitem Rf2 Deleted ',lineitem_diff;
go
tpch_wait
go
select 'Throughput Stream4 ends';
go

select 'Throughput Stream5 starts';

set lineitem_start=(select count(*) from lineitem);
set orders_start=(select count(*) from orders);
go
call tpch_rf1 (c_path,6)
go
set lineitem_end=(select count(*) from lineitem);
set orders_end=(select count(*) from orders);
set lineitem_diff = lineitem_end - lineitem_start;
set orders_diff = orders_end - orders_start;
select 'In lineitem Rf1 Inserted ',lineitem_diff;
select 'In orders Rf1 Inserted ',orders_diff;
go
commit
go
tpch_wait
go

set lineitem_start=(select count(*) from lineitem);
go
call tpch_rf2 (c_path,6)
go
commit
go
set lineitem_end=(select count(*) from lineitem);
set lineitem_diff = lineitem_start - lineitem_end;
select 'In lineitem Rf2 Deleted ',lineitem_diff;
go
tpch_wait
go
select 'Throughput Stream5 ends';
go

select 'Throughput Stream6 starts';

set lineitem_start=(select count(*) from lineitem);
set orders_start=(select count(*) from orders);
go
call tpch_rf1 (c_path,7)
go
set lineitem_end=(select count(*) from lineitem);
set orders_end=(select count(*) from orders);
set lineitem_diff = lineitem_end - lineitem_start;
set orders_diff = orders_end - orders_start;
select 'In lineitem Rf1 Inserted ',lineitem_diff;
select 'In orders Rf1 Inserted ',orders_diff;
go
commit
go
tpch_wait
go

set lineitem_start=(select count(*) from lineitem);
go
call tpch_rf2 (c_path,7)
go
commit
go
set lineitem_end=(select count(*) from lineitem);
set lineitem_diff = lineitem_start - lineitem_end;
select 'In lineitem Rf2 Deleted ',lineitem_diff;
go
tpch_wait
go
select 'Throughput Stream6 ends';
go

select 'Throughput Stream7 starts';

set lineitem_start=(select count(*) from lineitem);
set orders_start=(select count(*) from orders);
go
call tpch_rf1 (c_path,8)
go
set lineitem_end=(select count(*) from lineitem);
set orders_end=(select count(*) from orders);
set lineitem_diff = lineitem_end - lineitem_start;
set orders_diff = orders_end - orders_start;
select 'In lineitem Rf1 Inserted ',lineitem_diff;
select 'In orders Rf1 Inserted ',orders_diff;
go
commit
go
tpch_wait
go

set lineitem_start=(select count(*) from lineitem);
go
call tpch_rf2 (c_path,8)
go
commit
go
set lineitem_end=(select count(*) from lineitem);
set lineitem_diff = lineitem_start - lineitem_end;
select 'In lineitem Rf2 Deleted ',lineitem_diff;
go
tpch_wait
go
select 'Throughput Stream7 ends';
go

select 'Throughput Stream8 starts';

set lineitem_start=(select count(*) from lineitem);
set orders_start=(select count(*) from orders);
go
call tpch_rf1 (c_path,9)
go
set lineitem_end=(select count(*) from lineitem);
set orders_end=(select count(*) from orders);
set lineitem_diff = lineitem_end - lineitem_start;
set orders_diff = orders_end - orders_start;
select 'In lineitem Rf1 Inserted ',lineitem_diff;
select 'In orders Rf1 Inserted ',orders_diff;
go
commit
go
```

```
tpch_wait
go

set lineitem_start=(select count(*) from lineitem);
go
call tpch_rf2 (c_path,9)
go
commit
go
set lineitem_end=(select count(*) from lineitem);
set lineitem_diff = lineitem_start - lineitem_end;
select 'In lineitem Rf2 Deleted ',lineitem_diff;
go
tpch_wait
go
select 'Throughput Stream8 ends';
go

select 'Throughput Stream9 starts';

set lineitem_start=(select count(*) from lineitem);
set orders_start=(select count(*) from orders);
go
call tpch_rf1 (c_path,10)
go
set lineitem_end=(select count(*) from lineitem);
set orders_end=(select count(*) from orders);
set lineitem_diff = lineitem_end - lineitem_start;
set orders_diff = orders_end - orders_start;
select 'In lineitem Rf1 Inserted ',lineitem_diff;
select 'In orders Rf1 Inserted ',orders_diff;
go
commit
go
tpch_wait
go

set lineitem_start=(select count(*) from lineitem);
go
call tpch_rf2 (c_path,10)
go
commit
go
set lineitem_end=(select count(*) from lineitem);
set lineitem_diff = lineitem_start - lineitem_end;
select 'In lineitem Rf2 Deleted ',lineitem_diff;
go
tpch_wait
go
select 'Throughput Stream9 ends';
go

set qstop = now(*)

select 'Refresh Stream START -- ,', dateformat (qstart, 'yyyy-mm-dd
hh:nn:ss.sss')
select 'Refresh Stream STOP -- ,', dateformat (qstop, 'yyyy-mm-dd
hh:nn:ss.sss')
SET n_seconds=cast(datediff(millisecond,qstart,qstop) AS nu-
meric(16,5))/1000
select 'Refresh Stream Elapsed time -- ',+cast(n_seconds AS var-
char(20))+ ' seconds'
go
```

B.15 update_refresh.sql

```
-- THIS IS A GENERATED FILE.
-- PLEASE CHANGE gen_update_refresh.sh FOR MODIFICATIONS
create variable c_path varchar(128);
create variable qstart timestamp;
create variable qstop timestamp;
create variable n_seconds numeric(16,5);
create variable lineitem_start bigint;
create variable lineitem_end bigint;
create variable lineitem_diff bigint;
create variable orders_start bigint;
create variable orders_end bigint;
create variable orders_diff bigint;
go


set qstart = now(*)
set c_path='/testdb/data/'
select @@servername, db_name()
go

set lineitem_start=(select count(*) from lineitem)
set orders_start=(select count(*) from orders)
go
call tpch_rf1 (c_path,1)
go
set lineitem_end=(select count(*) from lineitem)
set orders_end=(select count(*) from orders)
set lineitem_diff=lineitem_end-lineitem_start
set orders_diff=orders_end-orders_start
select 'RF1 Inserted Lineitem records ',lineitem_diff
select 'RF1 Inserted Orders records ',orders_diff
go
commit
go
tpch_wait
go

set lineitem_start=(select count(*) from lineitem)
go
call tpch_rf2 (c_path,1)
go
set lineitem_end=(select count(*) from lineitem)
set lineitem_diff=lineitem_start-lineitem_end
select 'RF2 Deleted Lineitem records ',lineitem_diff
go
commit
go
tpch_wait

go

set qstop = now(*)

select 'Refresh Stream START -- ,', dateformat (qstart, 'yyyy-mm-dd
hh:nn:ss.sss')
select 'Refresh Stream STOP -- ,', dateformat (qstop, 'yyyy-mm-dd
hh:nn:ss.sss')
SET n_seconds=cast(datediff(millisecond,qstart,qstop) AS nu-
meric(16,5))/1000
select 'Refresh Stream Elapsed time -- ',+cast(n_seconds AS var-
char(20))+ ' seconds'
go
```

B.16 tpch_wait.sql

```
-- THIS IS A GENERATED FILE.
```

```
-- PLEASE CHANGE gen_tpch_wait.sh FOR MODIFICATIONS

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

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

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

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

  select 'xp_cmdshell "sleep 120"' into command;

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

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

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

             if ( mainfree > 372000000 )
                       then leave waitloop;
                       end if;

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

  END LOOP waitloop;

  drop table t_iq_spaceused;
  commit;
end
;
```

Appendix - C Query Text and Output

C.1 Qualification Query Output

Query 1

Query Text

```
9> select
10>         l_returnflag,
11>         l_linestatus,
12>         sum(l_quantity) as sum_qty,
13>         sum(l_extendedprice) as sum_base_price,
14>         sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
15>         sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as
sum_charge,
16>         avg(l_quantity) as avg_qty,
17>         avg(l_extendedprice) as avg_price,
18>         avg(l_discount) as avg_disc,
19>         count(*) as count_order
20> from
21>         lineitem
22> where
23>         l_shipdate <= dateadd(day, -90, '1998-12-01')
24> group by
25>         l_returnflag,
26>         l_linestatus
27> order by
28>         l_returnflag,
29>         l_linestatus
30>
```

Query Result

```
l_returnflag l_linestatus sum_qty                 sum_base_price
sum_disc_price            sum_charge              avg_qty
avg_price                 avg_disc                count_order
------------ ------------ -------------------------- -------------------------- --------------
-------------- ---------------------------- ---------------------------- --------------------------
-- -------------------------- --------------------
A            F                 37734107.000000
56586554400.729141        53758257134.869675
55909065222.828072                25.522006              38273.129735
0.049985            1478493
N            F                   991417.000000         1487504710.379999
1413082168.054105         1469649223.194362
25.516472             38284.467761                 0.050093
38854
N            O                 74476040.000000
111701729697.741302       106118230307.604965
110367043872.497467               25.502227              38249.117989
0.049997            2920374
R            F                 37719753.000000
56568041380.899460        53741292684.604080
55889619119.833153                25.505794              38250.854626
0.050009            1478870

(4 rows affected)
```

Query 2

Query Text

```
3>
4> select top 100
5>         s_acctbal,
6>         s_name,
7>         n_name,
8>         p_partkey,
9>         p_mfgr,
10>         s_address,
11>         s_phone,
12>         s_comment
13> from
14>         part,
15>         supplier,
16>         partsupp,
17>         nation,
18>         region
19> where
20>         p_partkey = ps_partkey
21>         and s_suppkey = ps_suppkey
22>         and p_size = 15
23>         and p_type like '%BRASS'
24>         and s_nationkey = n_nationkey
25>         and n_regionkey = r_regionkey
26>         and r_name = 'EUROPE'
27>         and ps_supplycost = (
28>                 select
29>                         min(ps_supplycost)
30>                 from
31>                         partsupp,
32>                         supplier,
33>                         nation,
34>                         region
35>                 where
36>                         p_partkey = ps_partkey
37>                         and s_suppkey = ps_suppkey
38>                         and s_nationkey = n_nationkey
39>                         and n_regionkey = r_regionkey
40>                         and r_name = 'EUROPE'
41>         )
42> order by
43>         s_acctbal desc,
44>         n_name,
45>         s_name,
46>         p_partkey
```

Query Result

```
s_acctbal               s_name                    n_name
p_partkey  p_mfgr                  s_address                      s_phone
s_comment
------------------------- ------------------------- ------------------------- ---------- -----
-------------------- ---------------------------------------- --------------- -----------------
-------------------------------------------------------------------------------
9938.530000 Supplier#000005359        UNITED KINGDOM
185358 Manufacturer#4           QKuHYh,vZGiwu2FWEJoLDx04
33-429-790-6131 uriously regular requests hag
9937.840000 Supplier#000005969        ROMANIA
108438 Manufacturer#1           ANDEN-
SOSmk,miq23Xfb5RWt6dvUcvt6Qa        29-520-692-3537 efully ex-
press instructions. regular requests against the slyly fin
9936.220000 Supplier#000005250        UNITED KINGDOM
249 Manufacturer#4           B3rqp0xbSEim4Mpy2RH J                  33-
320-228-2957 etect about the furiously final accounts. slyly ironic pinto
beans sleep inside the furiously
9923.770000 Supplier#000002324        GERMANY
29821 Manufacturer#4           y3OD9UywSTOk                   17-
779-299-1839 ackages boost blithely. blithely regular deposits c
9871.220000 Supplier#000006373        GERMANY
43868 Manufacturer#5           J8fcXWsTqM                     17-
```

```
813-485-8637 etect blithely bold asymptotes. fluffily ironic platelets
wake furiously; blit
9870.780000 Supplier#000001286        GERMANY
81285 Manufacturer#2
YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH    17-516-924-4574
regular accounts. furiously unusual courts above the fi
9870.780000 Supplier#000001286        GERMANY
181285 Manufacturer#4
YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH    17-516-924-4574
regular accounts. furiously unusual courts above the fi
9852.520000 Supplier#000008973        RUSSIA                    18972
Manufacturer#2          t5L67YdBYYH6o,Vz24jpDyQ9          32-
188-594-7038 rns wake final foxes. carefully unusual depende
9847.830000 Supplier#000008097        RUSSIA
130557 Manufacturer#2           xMe97bpE69NzdwLoX
32-375-640-3593  the special excuses. silent sentiments serve care-
fully final ac
9847.570000 Supplier#000006345        FRANCE
86344 Manufacturer#1
VSt3rzk3qG698u6ld8HhOByvrTcSTSvQlDQDag   16-886-766-7945
ges. slyly regular requests are. ruthless, express excuses cajole
blithely across the unu

(100 rows affected)
```

Query 3

```
Query Text
----------

3>
4> select top 10
5>          l_orderkey,
6>          sum(l_extendedprice * (1 - l_discount)) as revenue,
7>          dateformat (o_orderdate,'yyyy-mm-dd'),
8>          o_shippriority
9> from
10>         customer,
11>         orders,
12>         lineitem
13> where
14>         c_mktsegment = 'BUILDING'
15>         and c_custkey = o_custkey
16>         and l_orderkey = o_orderkey
17>         and o_orderdate < '1995-03-15'
18>         and l_shipdate > '1995-03-15'
19> group by
20>         l_orderkey,
21>         o_orderdate,
22>         o_shippriority
23> order by
24>         revenue desc,
25>         o_orderdate
26>
Query Result
------------

l_orderkey          revenue                   datefor-
mat(orders.o_orderdate,'yyyy-mm-dd') o_shippriority
------------------- -------------------------- ---------------------------------------- --
------------
2456423              406181.011100 1995-03-05
0
3459808              405838.698900 1995-03-04
0
492164              390324.061000 1995-02-19
0
1188320              384537.935900 1995-03-09
0
2435712              378673.055800 1995-02-26
0
4878020              378376.795200 1995-03-12
0
5521732              375153.921500 1995-03-13
0
2628192              373133.309400 1995-02-22
0
993600              371407.459500 1995-03-05
0
2300070              367371.145200 1995-03-13
0

(10 rows affected)
```

Query 4

```
Query Text
----------

2>
3> select
4>          o_orderpriority,
5>          count(*) as order_count
6> from
7>          orders
8> where
9>          o_orderdate >= '1993-07-01'
10>         and o_orderdate < dateadd(month, 3, '1993-07-01')
11>         and exists (
12>                 select *
13>                 from
14>                         lineitem
15>                 where
16>                         l_orderkey = o_orderkey
17>                         and l_commitdate < l_receiptdate
18>         )
19> group by
20>         o_orderpriority
21> order by
22>         o_orderpriority
Query Result

------------

o_orderpriority order_count
--------------- --------------------
1-URGENT                 10594
2-HIGH                   10476
3-MEDIUM                 10410
4-NOT SPECIFIED          10556
5-LOW                    10487

(5 rows affected)
```

Query 5

```
Query Text
----------

2>
3> select
4>          n_name,
5>          sum(l_extendedprice * (1 - l_discount)) as revenue
6> from
7>          customer,
8>          orders,
9>          lineitem,
10>         supplier,
11>         nation,
12>         region
13> where
```

```
14>         c_custkey = o_custkey
15>         and l_orderkey = o_orderkey
16>         and l_suppkey = s_suppkey
17>         and c_nationkey = s_nationkey
18>         and s_nationkey = n_nationkey
19>         and n_regionkey = r_regionkey
20>         and r_name = 'ASIA'
21>         and o_orderdate >= '1994-01-01'
22>         and o_orderdate < dateadd(year, 1, '1994-01-01')
23> group by
24>         n_name
25> order by
26>         revenue desc
27>
Query Result
------------

n_name                    revenue
------------------------- --------------------------
INDONESIA                      55502041.169700
VIETNAM                        55295086.996700
CHINA                          53724494.256600
INDIA                          52035512.000200
JAPAN                          45410175.695400

(5 rows affected)
```

Query 6

```
Query Text
----------

2>
3> select
4>         sum(l_extendedprice * l_discount) as revenue
5> from
6>         lineitem
7> where
8>         l_shipdate >= '1994-01-01'
9>         and l_shipdate < dateadd(year, 1, '1994-01-01')
10>         and l_discount between .06 - 0.01 and .06 + 0.01
11>         and l_quantity < 24
12>
Query Result
------------

revenue
---------------------------
123141078.228300

(1 row affected)
```

Query 7

```
Query Text
----------

3>
4> select
5>         supp_nation,
6>         cust_nation,
7>         l_year,
8>         sum(volume) as revenue
9> from
10>         (
11>                 select
12>                         n1.n_name as supp_nation,
13>                         n2.n_name as cust_nation,
14>                         year(l_shipdate) as l_year,
15>                         l_extendedprice * (1 - l_discount) as
volume
16>                 from
17>                         supplier,
18>                         lineitem,
19>                         orders,
20>                         customer,
21>                         nation n1,
22>                         nation n2
23>                 where
24>                         s_suppkey = l_suppkey
25>                         and o_orderkey = l_orderkey
26>                         and c_custkey = o_custkey
27>                         and s_nationkey = n1.n_nationkey
28>                         and c_nationkey = n2.n_nationkey
29>                         and (
30>                                 (n1.n_name = 'FRANCE' and
n2.n_name = 'GERMANY')
31>                                 or (n1.n_name = 'GERMANY'
and n2.n_name = 'FRANCE')
32>                         )
33>                         and l_shipdate between '1995-01-01'
and '1996-12-31'
34>         ) as shipping
35> group by
36>         supp_nation,
37>         cust_nation,
38>         l_year
39> order by
40>         supp_nation,
41>         cust_nation,
42>         l_year
43>
Query Result
------------

supp_nation          cust_nation          l_year      revenue
-------------------- -------------------- ----------- --------------------------
FRANCE               GERMANY                    1995
54639732.733600
FRANCE               GERMANY                    1996
54633083.307600
GERMANY              FRANCE                     1995
52531746.669700
GERMANY              FRANCE                     1996
52520549.022400

(4 rows affected)
```

Query 8

```
Query Text
----------

4>
5> select
6>         o_year,
7>         sum(case
8>                 when nation = 'BRAZIL' then volume
9>                 else 0
10>         end) / sum(volume) as mkt_share
11> from
12>         (
13>                 select
14>                         year(o_orderdate) as o_year,
15>                         l_extendedprice * (1 - l_discount) as
volume,
16>                         n2.n_name as nation
17>                 from
18>                         part,
19>                         supplier,
20>                         lineitem,
21>                         orders,
22>                         customer,
```

```
23>                             nation n1,
24>                             nation n2,
25>                             region
26>                     where
27>                             p_partkey = l_partkey
28>                             and s_suppkey = l_suppkey
29>                             and l_orderkey = o_orderkey
30>                             and o_custkey = c_custkey
31>                             and c_nationkey = n1.n_nationkey
32>                             and n1.n_regionkey = r_regionkey
33>                             and r_name = 'AMERICA'
34>                             and s_nationkey = n2.n_nationkey
35>                             and o_orderdate between '1995-01-01'
and '1996-12-31'
36>                             and p_type = 'ECONOMY ANODIZED
STEEL'
37>             ) as all_nations
38> group by
39>             o_year
40> order by
41>             o_year
42>
Query Result
------------

o_year      mkt_share
----------- --------------------------
1995                0.034436
1996                0.041486

(2 rows affected)
```

Query 9

```
Query Text
----------

3>
4> select
5>          nation,
6>          o_year,
7>          sum(amount) as sum_profit
8> from
9>          (
10>                     select
11>                             n_name as nation,
12>                             year(o_orderdate) as o_year,
13>                             l_extendedprice * (1 - l_discount) -
ps_supplycost * l_quantity as amount
14>                     from
15>                             part,
16>                             supplier,
17>                             lineitem,
18>                             partsupp,
19>                             orders,
20>                             nation
21>                     where
22>                             s_suppkey = l_suppkey
23>                             and ps_suppkey = l_suppkey
24>                             and ps_partkey = l_partkey
25>                             and p_partkey = l_partkey
26>                             and o_orderkey = l_orderkey
27>                             and s_nationkey = n_nationkey
28>                             and p_name like '%green%'
29>             ) as profit
30> group by
31>             nation,
32>             o_year
33> order by
34>             nation,
35>             o_year desc
36>
Query Result
------------

nation                    o_year      sum_profit
------------------------- ----------- ---------------------------
ALGERIA                         1998          31342867.234500
ALGERIA                         1997          57138193.023300
ALGERIA                         1996          56140140.133000
ALGERIA                         1995          53051469.653400
ALGERIA                         1994          53867582.128600
ALGERIA                         1993          54942718.132400
ALGERIA                         1992          54628034.712700
ARGENTINA                         1998            30211185.708100
ARGENTINA                         1997            50805741.752300
ARGENTINA                         1996            51923746.575500

(175 rows affected)
```

Query 10

```
Query Text
----------

3>
4> select top 20
5>          c_custkey,
6>          c_name,
7>          sum(l_extendedprice * (1 - l_discount)) as revenue,
8>          c_acctbal,
9>          n_name,
10>         c_address,
11>         c_phone,
12>         c_comment
13> from
14>         customer,
15>         orders,
16>         lineitem,
17>         nation
18> where
19>         c_custkey = o_custkey
20>         and l_orderkey = o_orderkey
21>         and o_orderdate >= '1993-10-01'
22>         and o_orderdate < dateadd(month, 3, '1993-10-01')
23>         and l_returnflag = 'R'
24>         and c_nationkey = n_nationkey
25> group by
26>         c_custkey,
27>         c_name,
28>         c_acctbal,
29>         c_phone,
30>         n_name,
31>         c_address,
32>         c_comment
33> order by
34>         revenue desc
35>
Query Result
------------

c_custkey  c_name                  revenue                 c_acctbal
n_name                    c_address                       c_phone
c_comment
---------- ------------------------ ------------------------- ------------------------- ---
------------------------- ---------------------------------- -------------- --------------
--------------------------------------------------------------------------------------------
-------
```

```
57040 Customer#000057040                734235.245500
632.870000 JAPAN                 Eioyzjf4pp                22-
895-641-3466 sits. slyly regular requests sleep alongside of the regu-
lar inst
143347 Customer#000143347               721002.694800
2557.470000 EGYPT                1aReFYv,Kw4
14-742-935-3718 ggle carefully enticing requests. final deposits use
bold, bold pinto beans. ironic, idle re
60838 Customer#000060838                679127.307700
2454.770000 BRAZIL
64EaJ5vMAHWJlBOxJklpNc2RJiWE            12-913-494-9813  need
to boost against the slyly regular account
101998 Customer#000101998               637029.566700
3790.890000 UNITED KINGDOM           01c9CILnNtfOQYmZj
33-593-865-6378 ress foxes wake slyly after the bold excuses. ironic
platelets are furiously carefully bold theodolites
125341 Customer#000125341               633508.086000
4983.510000 GERMANY              S29ODD6bceU8QSuuEJznkNaK
17-582-695-5962 arefully even depths. blithely even excuses sleep fu-
riously. foxes use except the dependencies. ca
25501 Customer#000025501                620269.784900
7725.040000 ETHIOPIA
W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ     15-874-808-6793 he
pending instructions wake carefully at the pinto beans. regular, final
instructions along the slyly fina
115831 Customer#000115831               596423.867200
5098.100000 FRANCE               rFeBbEEyk dl
ne7zV5fDrmiq1oK09wV7pxqCgIc 16-715-386-3788 l somas sleep. fu-
riously final deposits wake blithely regular pinto b
84223 Customer#000084223                594998.023900
528.650000 UNITED KINGDOM           nAVZCs6BaWap rrM27N
2qBnzc5WBauxbA       33-442-824-8191  slyly final deposits haggle
regular, pending dependencies. pending escapades wake
54289 Customer#000054289                585603.391800
5583.020000 IRAN                 vXCxoCsU0Bad5JQI ,oobkZ
20-834-292-4707 ely special foxes are quickly finally ironic p
39922 Customer#000039922                584878.113400
7321.110000 GERMANY
Zgy4s50l2GKN4pLDPBU8m342gIw6R           17-147-757-8036 y final
requests. furiously final foxes cajole blithely special platelets. f

(20 rows affected)
```

Query 11

Query 11
Query Text

```
----------

3>
4>
5> select
6>          ps_partkey,
7>          sum(ps_supplycost * ps_availqty) as value
8> from
9>          partsupp,
10>         supplier,
11>         nation
12> where
13>         ps_suppkey = s_suppkey
14>         and s_nationkey = n_nationkey
15>         and n_name = 'GERMANY'
16> group by
17>         ps_partkey having
18>                 sum(ps_supplycost * ps_availqty) > (
19>                         select
20>                                 sum(ps_supplycost *
ps_availqty) * 0.0001000000
21>                         from
22>                                 partsupp,
23>                                 supplier,
24>                                 nation
25>                         where
26>                                 ps_suppkey = s_suppkey
27>                                 and s_nationkey =
n_nationkey
28>                                 and n_name = 'GERMANY'
29>                 )
30> order by
31>         value desc
32>
Query Result
------------

ps_partkey value
---------- --------------------------
129760           17538456.860000
166726           16503353.920000
191287           16474801.970000
161758           16101755.540000
34452            15983844.720000
139035           15907078.340000
9403             15451755.620000
154358           15212937.880000
38823            15064802.860000
85606            15053957.150000

(1048 rows affected)
```

Query 12

Query Text

```
----------

3>
4> select
5>          l_shipmode,
6>          sum(case
7>                  when o_orderpriority = '1-URGENT'
8>                          or o_orderpriority = '2-HIGH'
9>                          then 1
10>                 else 0
11>         end) as high_line_count,
12>         sum(case
13>                 when o_orderpriority <> '1-URGENT'
14>                         and o_orderpriority <> '2-HIGH'
15>                         then 1
16>                 else 0
17>         end) as low_line_count
18> from
19>         orders,
20>         lineitem
21> where
22>         o_orderkey = l_orderkey
23>         and l_shipmode in ('MAIL', 'SHIP')
24>         and l_commitdate < l_receiptdate
25>         and l_shipdate < l_commitdate
26>         and l_receiptdate >= '1994-01-01'
27>         and l_receiptdate < dateadd(year, 1, '1994-01-01')
28> group by
29>         l_shipmode
30> order by
31>         l_shipmode
32>
Query Result
------------

l_shipmode high_line_count      low_line_count
---------- -------------------- --------------------
```

```
MAIL                    6202            9324
SHIP                    6200            9262

(2 rows affected)
```

Query 13

```
Query Text
----------

3>
4> select
5>          c_count,
6>          count(*) as custdist
7> from
8>          (
9>                      select
10>                             c_custkey,
11>                             count(o_orderkey)
12>                     from
13>                             customer left outer join orders on
14>                                     c_custkey = o_custkey
15>                                     and o_comment not like
'%special%requests%'
16>                     group by
17>                             c_custkey
18>         ) as c_orders (c_custkey, c_count)
19> group by
20>         c_count
21> order by
22>         custdist desc,
23>         c_count desc
24>
Query Result
------------

c_count             custdist
------------------- -------------------
0                    50005
9                     6641
10                    6532
11                    6014
8                     5937
12                    5639
13                    5024
19                    4793
7                     4687
17                    4587

(42 rows affected)
```

Query 14

```
Query Text
----------

3>
4> select
5>          100.00 * sum(case
6>                      when p_type like 'PROMO%'
7>                              then l_extendedprice * (1 - l_discount)
8>                      else 0
9>          end) / sum(l_extendedprice * (1 - l_discount)) as
promo_revenue
10> from
11>         lineitem,
12>         part
13> where
14>         l_partkey = p_partkey
15>         and l_shipdate >= '1995-09-01'
16>         and l_shipdate < dateadd(month, 1, '1995-09-01')
17>
Query Result
------------

promo_revenue
----------------------------
16.380779

(1 row affected)
```

Query 15

```
Query Text
----------

3>
4> create view revenue0 (supplier_no, total_revenue) as
5>          select
6>                      l_suppkey,
7>                      sum(l_extendedprice * (1 - l_discount))
8>          from
9>                      lineitem
10>         where
11>                     l_shipdate >= '1996-01-01'
12>                     and l_shipdate < dateadd(month,3,'1996-01-01')
13>         group by
14>                     l_suppkey
1>
2> select
3>          s_suppkey,
4>          s_name,
5>          s_address,
6>          s_phone,
7>          total_revenue
8> from
9>          supplier,
10>         revenue0
11> where
12>         s_suppkey = supplier_no
13>         and total_revenue = (
14>                     select
15>                             max(total_revenue)
16>                     from
17>                             revenue0
18>         )
19> order by
20>         s_suppkey
21>
s_suppkey  s_name                   s_address
s_phone         total_revenue
---------- ------------------------- ---------------------------------------- --------------- --
-------------------------
8449 Supplier#000008449        Wp34zim9qYFbVctdW
20-469-856-8873              1772627.208700

(1 row affected)
1>
2> drop view revenue0
Query Result
------------
```

Query 16

```
Query Text
----------

3>
4>
5> select
```

```
6>          p_brand,
7>          p_type,
8>          p_size,
9>          count(distinct ps_suppkey) as supplier_cnt
10> from
11>         partsupp,
12>         part
13> where
14>         p_partkey = ps_partkey
15>         and p_brand <> 'Brand#45'
16>         and p_type not like 'MEDIUM POLISHED%'
17>         and p_size in (49, 14, 23, 45, 19, 3, 36, 9)
18>         and ps_suppkey not in (
19>                     select
20>                                 s_suppkey
21>                     from
22>                                 supplier
23>                     where
24>                                 s_comment like
'%Customer%Complaints%'
25>         )
26> group by
27>         p_brand,
28>         p_type,
29>         p_size
30> order by
31>         supplier_cnt desc,
32>         p_brand,
33>         p_type,
34>         p_size
35>
Query Result
------------

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

(18314 rows affected)
```

Query 17

Query Text

```
----------

3>
4> select
5>          sum(l_extendedprice) / 7.0 as avg_yearly
6> from
7>          lineitem,
8>          part
9> where
10>         p_partkey = l_partkey
11>         and p_brand = 'Brand#23'
12>         and p_container = 'MED BOX'
13>         and l_quantity < (
14>                     select
15>                                 0.2 * avg(l_quantity)
16>                     from
17>                                 lineitem
18>                     where
19>                                 l_partkey = p_partkey
20>         )
21>
Query Result
------------

avg_yearly
--------------------------
348406.054286

(1 row affected)
```

Query 18

Query Text

```
----------

3>
4>
5> select top 100
6>          c_name,
7>          c_custkey,
8>          o_orderkey,
9>          dateformat (o_orderdate,'yyyy-mm-dd'),
10>         o_totalprice,
11>         sum(l_quantity)
12> from
13>         customer,
14>         orders,
15>         lineitem
16> where
17>         o_orderkey in (
18>                     select
19>                                 l_orderkey
20>                     from
21>                                 lineitem
22>                     group by
23>                                 l_orderkey having
24>                                             sum(l_quantity) > 300
25>         )
26>         and c_custkey = o_custkey
27>         and o_orderkey = l_orderkey
28> group by
29>         c_name,
30>         c_custkey,
31>         o_orderkey,
32>         o_orderdate,
33>         o_totalprice
34> order by
35>         o_totalprice desc,
36>         o_orderdate
37>
Query Result
------------

c_name                  c_custkey  o_orderkey          datefor-
mat(orders.o_orderdate,'yyyy-mm-dd') o_totalprice
sum(lineitem.l_quantity)
----------------------- ---------- -------------------- ----------------------------------
------ -------------------------- --------------------------
Customer#000128120          128120              4722021 1994-04-07
544089.090000              323.000000
Customer#000144617          144617              3043270 1997-02-12
530604.440000              317.000000
Customer#000013940           13940              2232932 1997-04-13
522720.610000              304.000000
Customer#000066790           66790              2199712 1996-09-30
515531.820000              327.000000
```

```
Customer#000046435          46435          4745607 1997-07-03
508047.990000           309.000000
Customer#000015272          15272          3883783 1993-07-28
500241.330000           302.000000
Customer#000146608          146608          3342468 1994-06-12
499794.580000           303.000000
Customer#000096103          96103          5984582 1992-03-16
494398.790000           312.000000
Customer#000024341          24341          1474818 1992-11-15
491348.260000           302.000000
Customer#000137446          137446          5489475 1997-05-23
487763.250000           311.000000

(57 rows affected)
```

Query 19

Query Text

```
----------

3>
4>
5> select
6>          sum(l_extendedprice* (1 - l_discount)) as revenue
7> from
8>          lineitem,
9>          part
10> where
11>         (
12>                     p_partkey = l_partkey
13>                     and p_brand = 'Brand#12'
14>                     and p_container in ('SM CASE', 'SM BOX', 'SM
PACK', 'SM PKG')
15>                     and l_quantity >= 1 and l_quantity <= 1 + 10
16>                     and p_size between 1 and 5
17>                     and l_shipmode in ('AIR', 'AIR REG')
18>                     and l_shipinstruct = 'DELIVER IN PERSON'
19>         )
20>         or
21>         (
22>                     p_partkey = l_partkey
23>                     and p_brand = 'Brand#23'
24>                     and p_container in ('MED BAG', 'MED BOX', 'MED
PKG', 'MED PACK')
25>                     and l_quantity >= 10 and l_quantity <= 10 + 10
26>                     and p_size between 1 and 10
27>                     and l_shipmode in ('AIR', 'AIR REG')
28>                     and l_shipinstruct = 'DELIVER IN PERSON'
29>         )
30>         or
31>         (
32>                     p_partkey = l_partkey
33>                     and p_brand = 'Brand#34'
34>                     and p_container in ('LG CASE', 'LG BOX', 'LG
PACK', 'LG PKG')
35>                     and l_quantity >= 20 and l_quantity <= 20 + 10
36>                     and p_size between 1 and 15
37>                     and l_shipmode in ('AIR', 'AIR REG')
38>                     and l_shipinstruct = 'DELIVER IN PERSON'
39>         )
40>
Query Result
------------

revenue
---------------------------
3083843.057800

(1 row affected)
```

Query 20

Query Text

```
----------

3>
4> select
5>          s_name,
6>          s_address
7> from
8>          supplier,
9>          nation
10> where
11>         s_suppkey in (
12>                     select
13>                                 ps_suppkey
14>                     from
15>                                 partsupp
16>                     where
17>                                 ps_partkey in (
18>                                             select
19>                                                         p_partkey
20>                                             from
21>                                                         part
22>                                             where
23>                                                         p_name like 'for-
est%'
24>                                 )
25>                                 and ps_availqty > (
26>                                             select
27>                                                         0.5 *
sum(l_quantity)
28>                                             from
29>                                                         lineitem
30>                                             where
31>                                                         l_partkey =
ps_partkey
32>                                                         and l_suppkey =
ps_suppkey
33>                                                         and l_shipdate >=
'1994-01-01'
34>                                                         and l_shipdate <
dateadd(year,1,'1994-01-01')
35>                                 )
36>         )
37>         and s_nationkey = n_nationkey
38>         and n_name = 'CANADA'
39> order by
40>         s_name
41>
Query Result
------------

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

(204 rows affected)
```

Query 21

Query Text

```
4>
5> select
6>      top 100 s_name,
7>      count(*) as numwait
8> from
9>      supplier,
10>       lineitem l1,
11>       orders,
12>       nation
13> where
14>      s_suppkey = l1.l_suppkey
15>      and o_orderkey = l1.l_orderkey
16>      and o_orderstatus = 'F'
17>      and l1.l_receiptdate > l1.l_commitdate
18>      and exists (
19>            select
20>                  *
21>            from
22>                  lineitem l2
23>            where
24>                  l2.l_orderkey = l1.l_orderkey
25>                  and l2.l_suppkey <> l1.l_suppkey
26>      )
27>      and not exists (
28>            select
29>                  *
30>            from
31>                  lineitem l3
32>            where
33>                  l3.l_orderkey = l1.l_orderkey
34>                  and l3.l_suppkey <> l1.l_suppkey
35>                  and l3.l_receiptdate > l3.l_commitdate
36>      )
37>      and s_nationkey = n_nationkey
38>      and n_name = 'SAUDI ARABIA'
39> group by
40>      s_name
41> order by
42>      numwait desc,
43>      s_name
44>
```

Query Result
---

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

(100 rows affected)

Query 22

Query Text

```
3>
4> select
5>      cntrycode,
6>      count(*) as numcust,
7>      sum(c_acctbal) as totacctbal
8> from
9>      (
10>           select
11>                substr(c_phone,1,2) as cntrycode,
12>                c_acctbal
13>           from
14>                customer
15>           where
16>                substr(c_phone,1,2) in
17>                     ('13', '31', '23', '29', '30', '18', '17')
18>                and c_acctbal > (
19>                     select
20>                          avg(c_acctbal)
21>                     from
22>                          customer
23>                     where
24>                          c_acctbal > 0.00
25>                          and substr(c_phone,1,2) in
26>                               ('13', '31', '23', '29', '30', '18', '17')
27>                )
28>                and not exists (
29>                     select
30>                          *
31>                     from
32>                          orders
33>                     where

34>                          o_custkey = c_custkey
35>                )
36>     ) as custsale
37> group by
38>      cntrycode
39> order by
40>      cntrycode
41>
```

Query Result
---

cntrycode	numcust	totacctbal
13	888	6737713.990000
17	861	6460573.720000
18	964	7236687.400000
23	892	6701457.950000
29	948	7158866.630000
30	909	6808436.130000
31	922	6806670.180000

(7 rows affected)

C.2 Query Substitution Parameters

Opts.0

```
14      1994-03-01
2       37      COPPER ASIA
9       chocolate
20      honeydew        1996-01-01      UNITED STATES
6       1994-01-01      0.02    24
17      Brand#15 WRAP CASE
18      315
8       BRAZIL  AMERICA         SMALL ANODIZED TIN
```

21 MOROCCO
13 unusual packages
3 MACHINERY 1995-03-17
22 34 19 29 14 26 25
22
16 Brand#21 PROMO BURNISHED 41 20
31 10 22 3 8 33
4 1995-08-01
11 UNITED STATES 0.0000000333
15 1993-11-01
1 89
10 1993-12-01
19 Brand#52 Brand#45 Brand#34 7 11 22
5 ASIA 1994-01-01
7 ALGERIA BRAZIL
12 FOB TRUCK 1994-01-01

Opts.1

21 GERMANY
3 FURNITURE 1995-03-02
18 313
5 EUROPE 1994-01-01
11 JAPAN 0.0000000333
7 PERU ROMANIA
6 1994-01-01 0.07 25
20 salmon 1994-01-01 KENYA
17 Brand#12 WRAP JAR
12 TRUCK MAIL 1994-01-01
16 Brand#11 SMALL PLATED 16 5 15
29 18 27 9 11
15 1996-06-01
13 unusual requests
10 1994-10-01
2 25 STEEL AFRICA
8 ROMANIA EUROPE STANDARD POLISHED TIN
14 1994-06-01
19 Brand#55 Brand#23 Brand#24 2 12 29
9 blush
22 20 18 25 14 15 32
31
1 97
4 1993-05-01

Opts.2

6 1994-01-01 0.05 24
17 Brand#14 WRAP PKG
14 1994-09-01
16 Brand#41 LARGE BRUSHED 29 12
49 17 1 24 45 21
19 Brand#52 Brand#51 Brand#23
7 13 26
10 1993-07-01
9 azure
2 12 BRASS ASIA
15 1994-03-01
8 IRAQ MIDDLE EAST STANDARD BURNISHED TIN
5 MIDDLE EAST 1994-01-01
22 31 15 19 14 32 23
20
12 RAIL MAIL 1994-01-01
7 INDONESIA IRAQ
13 unusual requests
18 314
1 105
4 1995-12-01
20 cyan 1997-01-01 CANADA
3 MACHINERY 1995-03-19
11 ALGERIA 0.0000000333
21 UNITED STATES

Opts.3

8 CANADA AMERICA PROMO BRUSHED NICKEL
5 AFRICA 1994-01-01
4 1993-09-01
6 1994-01-01 0.02 24
17 Brand#11 SM CASE
7 ARGENTINA CANADA
1 113
18 312
22 20 15 21 11 19 24
31
14 1994-12-01
9 wheat
10 1994-04-01
15 1996-10-01
11 JORDAN 0.0000000333
20 orchid 1996-01-01 CHINA
2 50 NICKEL AFRICA
21 MOZAMBIQUE
19 Brand#14 Brand#44 Brand#22 3 14 22
13 unusual requests
16 Brand#21 STANDARD ANODIZED 1 14
34 38 22 37 12 20
12 AIR MAIL 1994-01-01
3 BUILDING 1995-03-05

Opts.4

5 ASIA 1995-01-01
21 INDIA
14 1995-03-01
19 Brand#11 Brand#22 Brand#11 8 15 29
15 1994-06-01
17 Brand#13 SM JAR
12 REG AIR MAIL 1995-01-01
6 1995-01-01 0.07 25
4 1996-04-01
9 steel
8 SAUDI ARABIA MIDDLE EAST PROMO PLATED
NICKEL
16 Brand#11 MEDIUM PLATED 27 28 34
49 8 5 38 25
11 ARGENTINA 0.0000000333
2 38 TIN EUROPE
10 1995-01-01
18 313
1 60
13 unusual requests
7 CHINA SAUDI ARABIA
22 17 13 11 28 22 24
16
3 MACHINERY 1995-03-21
20 bisque 1994-01-01 INDIA

Opts.5

21 ALGERIA
15 1997-01-01
4 1994-01-01
6 1995-01-01 0.05 24
7 IRAN JAPAN
16 Brand#41 ECONOMY POLISHED 6 3
17 2 12 8 14 9
19 Brand#13 Brand#15 Brand#11 3 16 25
18 315
14 1995-07-01

22 12 20 33 21 24 22
31
11 KENYA 0.0000000333
13 unusual accounts

3 BUILDING 1995-03-07
1 68
2 26 STEEL AFRICA
5 EUROPE 1995-01-01
8 JAPAN ASIA PROMO ANODIZED NICKEL
20 lemon 1993-01-01 UNITED KINGDOM
12 SHIP FOB 1995-01-01
17 Brand#14 SM PKG
10 1993-10-01
9 sienna

Opts.6

10 1994-08-01
3 HOUSEHOLD 1995-03-23
15 1994-10-01
13 express accounts
6 1995-01-01 0.02 24
8 EGYPT MIDDLE EAST ECONOMY POLISHED NICKEL
9 rosy
7 BRAZIL EGYPT
4 1996-08-01
11 BRAZIL 0.0000000333
22 24 22 16 18 15 26
31
18 312
12 FOB TRUCK 1994-01-01
1 76
5 MIDDLE EAST 1995-01-01
16 Brand#21 SMALL ANODIZED 25 45 37
18 10 43 28 15
2 14 BRASS EUROPE
14 1995-10-01
19 Brand#25 Brand#43 Brand#15 9 17 21
20 spring 1996-01-01 JAPAN
17 Brand#11 LG CASE
21 CHINA

Opts.7

18 314
8 VIETNAMASIA ECONOMY BURNISHED NICKEL
20 forest 1995-01-01 BRAZIL
21 IRAN
2 1 NICKEL AMERICA
4 1994-05-01
22 11 26 17 18 29 30
22
17 Brand#13 LG JAR
1 84
11 MOROCCO 0.0000000333
9 plum
19 Brand#22 Brand#31 Brand#54 4 18 29
3 AUTOMOBILE 1995-03-09
13 express accounts
5 AFRICA 1995-01-01
7 ROMANIA VIETNAM
10 1993-05-01
16 Brand#11 LARGE BURNISHED 24 27
21 22 48 11 17 34
6 1995-01-01 0.08 25
14 1996-01-01
15 1997-05-01
12 TRUCK FOB 1996-01-01

Opts.8

19 Brand#24 Brand#14 Brand#53 9 19 25
1 92
15 1995-01-01
17 Brand#15 LG PKG
5 AMERICA 1996-01-01
8 JORDAN MIDDLE EAST LARGE BRUSHED BRASS
9 orchid
12 RAIL FOB 1996-01-01
14 1996-04-01
7 IRAQ JORDAN
4 1996-11-01
3 HOUSEHOLD 1995-03-25
20 puff 1993-01-01 PERU
16 Brand#41 PROMO POLISHED 13 37
4 35 5 25 40 16
6 1996-01-01 0.05 25
22 27 12 16 24 13 21
25
10 1994-02-01
13 express accounts
2 39 TIN EUROPE
21 BRAZIL
18 315
11 CANADA 0.0000000333

Opts.9

8 ETHIOPIA AFRICA LARGE PLATED BRASS
13 express accounts
2 27 COPPER AMERICA
20 chartreuse 1996-01-01 FRANCE
17 Brand#12 MED CASE
3 AUTOMOBILE 1995-03-11
6 1996-01-01 0.03 24
21 ROMANIA
18 313
11 MOZAMBIQUE 0.0000000333
19 Brand#31 Brand#42 Brand#52 4 20 21
10 1994-11-01
15 1997-08-01
4 1994-08-01
22 34 30 21 15 14 12
31
1 100
7 CANADA ETHIOPIA
12 AIR SHIP 1996-01-01
9 misty
14 1996-07-01
5 ASIA 1996-01-01
16 Brand#21 SMALL BRUSHED 17 3 40
14 41 15 9 48

Appendix - D
Driver Source Code

D.1 run_tpch

```
#!/bin/ksh

if (( $# < 1 ))
then
        echo
        echo "Usage: run_tpch scope scale_factor (plus additional
args depending upon scope)"
        echo
        echo "scope values:"
        echo ""
        echo "              - load              load the tpch test
database"
        echo ""
        echo "              - power     do a single power run "
        echo "                                  (with or without re-
fresh streams)"
        echo
        echo "              - throughput        do a single
throughput run"
        echo "                                  (with or without re-
fresh streams)"
        echo
        echo "              - refresh   run refresh pair(s)"
        echo
        echo "              - perf              run a single power-
throughput cycle (without a load)"
        echo
        echo "              - all               do a full-audit run
(load + 2 power-throughput cycles)"
        echo
        echo "              -audit_only         Runs only au-
dit_scripts."
        echo
        echo "To get usage help on individual scope options, type"
        echo
        echo "                          run_tpch scope help"
        echo
        exit 1
fi
unset STREAM_COUNT
nParams=$#
i=2

if [ ! -z "$2" ] && [ $2 != help ]
then
    while [ $i -le $nParams ]
    do
        eval j=\$$i
        junk=`expr $j + 1` 2>/dev/null
        if [ $? != 0 ] || [ -z "$junk" ]
        then
            echo "input parameter $i supplied is not correct"
            echo " Please run ./run_tpch help to get usage informa-
tion."
            exit
        fi
        i=`expr $i + 1`
    done
fi

SCOPE=$1

if [ $SCOPE = audit_only ]
then
        if [ $# -lt 2 ] || [ $2 = help ]
        then

                echo
                echo "Usage: run_tpch $SCOPE scale_factor"
                echo ""
                echo "      -- This runs the audit scripts and these
need to be run before and after perf test."
                echo
                exit
        fi
        SCALE_FACTOR=$2
        . ./bm_setup $SCALE_FACTOR
        echo
        echo "run_tpch SCOPE=$SCOPE"

elif [ $SCOPE = load ]
then
        if [ $# -lt 2 ] || [ $2 = help ]
        then
                echo
                echo "Usage: run_tpch $SCOPE scale_factor [en-
able_monitoring] [stream_count] "
                echo ""
                echo "                    -- Default value for "
                echo "                                enable_monitoring:
0 (no monitoring)"
                echo "                                stream_count: will
be set to minimum required "
                echo "
        stream count for the supplied "
                echo "
        Scale Factor          "
                echo
                echo '                    -- This generates
$stream_count number of query streams'
                echo
                exit
        else
                SCALE_FACTOR=$2

                if [ -z "$3" ]
                then
                        Enable_Monitoring=0; export En-
able_Monitoring
                else
                        Enable_Monitoring=$3; export En-
able_Monitoring
                fi

                if [ ! -z "$4" ]
                then
                        STREAM_COUNT=$4; export
STREAM_COUNT
                fi

                . ./bm_setup $SCALE_FACTOR      #bm_setup
call shouldn't be before $# variables assignment.

                echo
                echo
                echo "run_tpch SCOPE=$SCOPE
SCALE_FACTOR=$SCALE_FACTOR
ENABLE_MONITORING=$Enable_Monitoring
STREAM_COUNT=$STREAM_COUNT"
                echo
                echo
```

```
        fi
elif [ $SCOPE = power ]
then
        if [ $# -lt 2 ] || [ $2 = help ]
        then
                echo
                echo "Usage: run_tpch $SCOPE scale_factor
[with_rf] [enable_monitoring]"
                echo ""
                echo "                -- Default value for "
                echo "                        a) with_rf option: 1
(run with refresh streams)"
                echo "                        b) en-
able_monitoring: 0 (no monitoring)"
                echo
                exit
        else
        SCALE_FACTOR=$2

                if [ -z "$3" ]
                then
                        WITH_RF=1
                else
                        WITH_RF=$3
                fi

                if [ -z "$4" ]
                then
                        Enable_Monitoring=0; export En-
able_Monitoring
                else
                        Enable_Monitoring=$4; export En-
able_Monitoring
                fi

                . ./bm_setup $SCALE_FACTOR

                echo
                echo
                echo "run_tpch SCOPE=$SCOPE
SCALE_FACTOR=$SCALE_FACTOR WITH_RF=$WITH_RF
ENABLE_MONITORING=$Enable_Monitoring "
                echo
                echo
        fi

elif [ $SCOPE = throughput ] || [ $SCOPE = all ]
then
        if [ $# -lt 2 ] || [ $2 = help ]
        then
                echo
                echo "Usage: run_tpch $SCOPE scale_factor
[with_rf] [enable_monitoring]"
                echo "
        [stream-count]"
                echo "                -- Default value for "
                echo "                        a) with_rf option: 1
(run with refresh streams)"
                echo "                        b) en-
able_monitoring: 0 (no monitoring)"
                echo "                        c) stream_count:
will be set to minimum required"
                echo "
stream count for the respective"
                echo "
scale factor."
                echo ""
                echo
                exit
        else
        SCALE_FACTOR=$2

                if [ -z "$3" ]
                then
                        WITH_RF=1
                else
                        WITH_RF=$3
                fi

                if [ -z "$4" ]
                then
                        Enable_Monitoring=0; export En-
able_Monitoring
                else
                        Enable_Monitoring=$4; export En-
able_Monitoring
                fi

                if [ ! -z "$5" ]
                then
                        STREAM_COUNT=$5; export
STREAM_COUNT
                fi

                . ./bm_setup $SCALE_FACTOR

                echo
                echo
                echo "run_tpch SCOPE=$SCOPE
SCALE_FACTOR=$SCALE_FACTOR WITH_RF=$WITH_RF
ENABLE_MONITORING=$Enable_Monitoring
STREAM_COUNT=$STREAM_COUNT"
                echo
                echo
        fi

elif [ $SCOPE = perf ]
then
        if [ $# -lt 3 ] || [ $2 = help ]
        then
                echo
                echo "Usage: run_tpch $SCOPE scale_factor
run_number [with_rf] [enable_monitoring]"
                echo "
        [stream-count]"
                echo "                -- Default value for "
                echo "                        a) with_rf option: 1
(run with refresh streams)"
                echo "                        b) en-
able_monitoring: 0 (no monitoring)"
                echo "                        c) stream_count:
will be set to minimum required"
                echo "
stream count for the respective"
                echo "
scale factor."
                echo ""
                echo
                exit
        else
        SCALE_FACTOR=$2
                RUN_NUMBER=$3
                if [ -z "$4" ]
                then
                        WITH_RF=1
                else
                        WITH_RF=$4
                fi

                if [ -z "$5" ]
```

```
                then
                        Enable_Monitoring=0; export En-
able_Monitoring
                else
                        Enable_Monitoring=$5; export En-
able_Monitoring
                fi

                if [ ! -z "$6" ]
                then
                        STREAM_COUNT=$6; export
STREAM_COUNT
                fi

                . ./bm_setup $SCALE_FACTOR

                echo
                echo
                echo "run_tpch SCOPE=$SCOPE
SCALE_FACTOR=$SCALE_FACTOR
RUN_NUMBER=$RUN_NUMBER WITH_RF=$WITH_RF
ENABLE_MONITORING=$Enable_Monitoring
STREAM_COUNT=$STREAM_COUNT"
                echo
                echo
        fi

elif [ $SCOPE = refresh ]
then
        if [ $2 = help ]
        then
                echo
                echo "Usage: run_tpch $SCOPE
$SCALE_FACTOR [stream_count] [enable_monitoring]"
                echo ""
                echo "                -- Default value for "
                echo "                        a) stream_count: 1
(runs one refresh pair)"
                echo "                        b) en-
able_monitoring: 0 (no monitoring)"
                echo ""
                exit
        else
                SCALE_FACTOR=$2
                if [ -z "$3" ]
                then
                        STREAM_COUNT=1; export
STREAM_COUNT
                else
                        STREAM_COUNT=$3; export
STREAM_COUNT
                fi

                if [ -z "$4" ]
                then
                        Enable_Monitoring=0; export En-
able_Monitoring
                else
                        Enable_Monitoring=$4; export En-
able_Monitoring
                fi

                . ./bm_setup $SCALE_FACTOR

                echo "run_tpch SCOPE=$SCOPE
SCALE_FACTOR=$SCALE_FACTOR
STREAM_COUNT=$STREAM_COUNT
ENABLE_MONITORING=$Enable_Monitoring"
        fi
else
        echo "Please supply valid name for scope"
        exit 1
fi

if [ -z "$RUN_NUMBER" ]
then
     RUN_NUMBER=0
fi


#Set Results directory with seq_number
SEQ_NUMBER=`cat ${DEV_DIR}/data/seq_number`
RESULTS=${RESULTS}/run_${SEQ_NUMBER};export RESULTS
mkdir -p $RESULTS > /dev/null 2>&1
SEQ_NUMBER=`expr $SEQ_NUMBER + 1 `
echo ${SEQ_NUMBER}>${DEV_DIR}/data/seq_number

$STOP_SERVER -stop all
rm $TPCH_ROOT/plans/* > /dev/null 2>&1

os=`uname`
if [ $os = AIX ]
then
        rm -f $RESULTS/sysunused
        echo "The following disks are assigned to the indicated vol-
ume groups">$RESULTS/sysunused
        lspv>>$RESULTS/sysunused
        echo "The following volume groups are currently
online">>$RESULTS/sysunused
        echo `date` >>$RESULTS/sysunused
        lsvg -o >>$RESULTS/sysunused
fi

sleep 10
echo Restart IQ with TPCH database: `date`
$START_SERVER @tpch.cfg $DEV_DIR/tpch.db
echo IQ restarted: `date`
echo "  "


if [ $SCOPE = 'all'  -o $SCOPE = 'load' ]
then
        # Load the tpch test database
        load_test $Enable_Monitoring
fi

if [ $SCOPE = 'all' ]
then
        start_log_details=`ls -l $DEV_DIR/tpch.log`
        echo "$start_log_details" > $RESULTS/mlog
fi

#Generate streams & set the Debug mode.
cwd=`pwd`
cd $QUERYGEN
SEED=`cat $DEV_DIR/data/seed`
. ./querygen.sh $SEED $SCALE_FACTOR ${STREAM_COUNT}
echo ''
cd $cwd

if [ $SCOPE = 'refresh' ]
then
        run_refresh.sh $STREAM_COUNT $Enable_Monitoring
fi

echo
echo
# After the load Completes run the Audit SQL if SCOPE=all
if [ $SCOPE = 'all' -o $SCOPE = 'audit_only' -o $SCOPE = 'perf' -a
$RUN_NUMBER = '1' ]
```

```
then
        echo "Running the Audit Script   `date` "
        dbisqlc -c "DSN=tpch_$VERSION" -q $SQL/dbtables-
syb.sql > $RESULTS/rdbtablest
        dbisqlc -c "DSN=tpch_$VERSION" -q $SQL/dew_cat1.sql >
$RESULTS/dew_cat1_start.out
        dbisqlc -c "DSN=tpch_$VERSION" -q $SQL/dew_cat2.sql >
$RESULTS/dew_cat2_start.out
        dbisqlc -c "DSN=tpch_$VERSION" -q $SQL/dew_cat3.sql >
$RESULTS/dew_cat3_start.out
fi

if [ $SCOPE = 'all' -o $SCOPE = 'power' -o $SCOPE = 'perf' -o
$SCOPE = 'throughput' ]
then
        SEED=`cat $DEV_DIR/data/seed`;export SEED
fi

if [ $SCOPE = 'all' -o $SCOPE = 'perf' ]
then
        ################## FIRST RUN ################

        #Run power test.
        powertest.sh Run $SCALE_FACTOR $WITH_RF $En-
able_Monitoring

        #Run throughput test.
        throughputtest.sh Run $SCALE_FACTOR $WITH_RF $En-
able_Monitoring $STREAM_COUNT

        echo "  "
        if [ $SCOPE = 'all' ]
        then
                tpch_report.sh $SCOPE $SCALE_FACTOR
$WITH_RF 0 0 1 $SEED $STREAM_COUNT 0  0 0 >
$RESULTS/mrun1_report.out
        fi

        if [ $SCOPE = 'perf' ]
        then
                tpch_report.sh $SCOPE $SCALE_FACTOR
$WITH_RF 0 0 $RUN_NUMBER $SEED $STREAM_COUNT >
$RESULTS/perf.out
        fi
fi

if [ $SCOPE = 'all' ]
then
        #Move the $TPCH_ROOT/plans to $RESULTS/plans.
        plan_count=$(ls -1 $TPCH_ROOT/plans/|wc -l)

        if [ $plan_count -ne 0 ]
        then
        mkdir $RESULTS/plans > /dev/null 2>&1
        mv $TPCH_ROOT/plans/*.html $RESULTS/plans/ >
/dev/null 2>&1
        fi

        move_out_files 1 $WITH_RF > /dev/null 2>&1
fi

if [ $SCOPE = 'all' ]
then
        ################SECOND RUN
####################
        #Run power test.
        powertest.sh Run $SCALE_FACTOR $WITH_RF $En-
able_Monitoring

        #Run throughput test.
        throughputtest.sh Run $SCALE_FACTOR $WITH_RF $En-
able_Monitoring $STREAM_COUNT

        echo "  "
        tpch_report.sh all $SCALE_FACTOR $WITH_RF 0 0 2
$SEED $STREAM_COUNT 0 0 0 > $RESULTS/mrun2_report.out
        ################################################
fi
if [ $SCOPE = 'all' -o $SCOPE = 'perf' -a $RUN_NUMBER = '2' ]
then
        dbisqlc -c "DSN=tpch_$VERSION" -q $SQL/dew_cat1.sql >
$RESULTS/dew_cat1_end.out
        dbisqlc -c "DSN=tpch_$VERSION" -q $SQL/dew_cat2.sql >
$RESULTS/dew_cat2_end.out
        dbisqlc -c "DSN=tpch_$VERSION" -q $SQL/dew_cat3.sql >
$RESULTS/dew_cat3_end.out
fi

if [ $SCOPE = 'all' -o $SCOPE = 'load' ]
then
        cat $RESULTS/ldeventlog |tr -s ' ' |tr -s '\n'|tr -s
'='>$RESULTS/ldeventlog.`date '+%y%m%d_%H%M%S'`
        rm $RESULTS/ldeventlog
        if [ $SCOPE = 'all' ]
        then
                move_out_files 2 $WITH_RF > /dev/null 2>&1
        fi
fi
if [ $SCOPE = 'all' -o $SCOPE = 'perf' ]
then
        generate_mlog.sh
        show_rf_impl
        aixmachine=`uname |grep -i aix`
        if [ ! -z "$aixmachine" ]
        then
        aix_sys_conf.sh $RESULTS/sysconfig

                machinename=`uname -n`
                cp $RESULTS/sysconfig/SoftwareConfig*
$RESULTS/syssw.`date '+%y%m%d_%H%M%S'`.$machinename
                cp $RESULTS/sysconfig/Tunables*
$RESULTS/sysostune.`date '+%y%m%d_%H%M%S'`.$machinename

                echo "Device Config info" > $RESULTS/syshw
                echo "==================" >>
$RESULTS/syshw
                cat $RESULTS/sysconfig/DeviceConfig* >>
$RESULTS/syshw

                echo "File System  info" >> $RESULTS/syshw
                echo "==================" >>
$RESULTS/syshw
                cat  $RESULTS/sysconfig/FilesystemConfig*>>
$RESULTS/syshw

                echo "Hardware confi info" >> $RESULTS/syshw
                echo "===================" >>
$RESULTS/syshw
                cat $RESULTS/sysconfig/HardwareConfig* >>
$RESULTS/syshw

                echo "Physical volume List info" >>
$RESULTS/syshw
                echo "=========================" >>
$RESULTS/syshw
                cat $RESULTS/sysconfig/Physical_Volume_List*
>> $RESULTS/syshw


                echo "Volume Detail info" >> $RESULTS/syshw
```

```
                        echo "==================" >>
$RESULTS/syshw
                        cat $RESULTS/sysconfig/Volume_Detail* >>
$RESULTS/syshw

                        echo "Volume_Group_List info" >>
$RESULTS/syshw
                        echo "======================" >>
$RESULTS/syshw
                        cat $RESULTS/sysconfig/Volume_Group_List* >>
$RESULTS/syshw

                        echo "Volume_Group_Member_list info" >>
$RESULTS/syshw
                        echo "============================" >>
$RESULTS/syshw
                        cat
$RESULTS/sysconfig/Volume_Group_Member_list* >>
$RESULTS/syshw

                        echo "Volume_Group_Pdisk_list info" >>
$RESULTS/syshw
                        echo "==========================" >>
$RESULTS/syshw
                        cat
$RESULTS/sysconfig/Volume_Group_Pdisk_list* >>
$RESULTS/syshw
                        mv $RESULTS/syshw $RESULTS/syshw.`date
'+%y%m%d_%H%M%S'`.$machinename
                fi
fi

if [ $SCOPE = 'power' ]
then
                #Run power test.
                powertest.sh All $SCALE_FACTOR $WITH_RF $En-
able_Monitoring
fi

if [ $SCOPE = 'throughput' ]
then
                #Run throughput test.
                throughputtest.sh All $SCALE_FACTOR $WITH_RF $En-
able_Monitoring $STREAM_COUNT
fi

#Clean up
rm $TEMPFILES/* > /dev/null 2>&1

#Move the $TPCH_ROOT/plans to $RESULTS/plans.
plan_count=$(ls -1 $TPCH_ROOT/plans/|wc -l)

if [ $plan_count -ne 0 ]
then
                mkdir $RESULTS/plans > /dev/null 2>&1
                mv $TPCH_ROOT/plans/*.html $RESULTS/plans/ >
/dev/null 2>&1
fi
if [ $os = AIX ]
then
                mv $RESULTS/sysunused $RESULTS/sysunused.`date
'+%y%m%d_%H%M%S'`
fi

#unset STREAM_COUNT. Otherwise same value will be remain in
consecutive runs.
unset STREAM_COUNT
```

Appendix - E ACID Transaction Source Code

E.1 Acid_atomic_main.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%
% Purpose of this test is to run and verify the pass of the ACID Atom-
icity
% test.

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

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

stringconnect "dsn=qual_15_0;"

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

include 'acid_functions.tst'
commit

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

%include 'acid_atomic_setup.tst'
run test 'acid_atomic_setup.tst'

stringconnect "dsn=qual_15_0;"
let counter=0

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


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

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

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

print   'Before Rollback History table count=',total_history_count


let counter = counter+1

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

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

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

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

print   'After Rollback History table count=',total_history_count

} ENDLOOP

commit

%
% Atomicity test with commit
%
stringconnect "dsn=qual_15_0;"
print ' '
print 'Starting atomicity test with commit '
print ' '
%include 'acid_atomic_setup.tst'
run test 'acid_atomic_setup.tst'

stringconnect "dsn=qual_15_0;"

open cur1 {select ordr, line, delta from aa_whattodo}
LOOP {
fetch cur1 into ordr, line, delta
if ROWSTATUS != FOUND then { BREAK LOOP } endif
print 'Acid transaction for: o_key-',ordr,'  l_key-', line,'  delta-',delta
execute {select o_totalprice, l_quantity, l_extendedprice
        from orders, lineitem
```

```
            where o_orderkey = l_orderkey and o_orderkey =^ and
l_linenumber = ^}
            substitute ordr, line
            into o_total, l_quan, l_price
print       'Initial values:'
print       'o_totalprice = ',o_total,'          l_quantity = ',l_quan,
            '               l_extendedprice = ',l_price
print       ''
print       ''
print   'Before Commit:'
print       '               l_extendedprice = ',l_price

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


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

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


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

print   'Before Commit History table count=',total_history_count

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

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

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

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

print ' '

} ENDLOOP

close cur1
commit

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

End Test
```

E.2 Acid_atomic_setup.tst

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

stringconnect "dsn=qual_15_0;"

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

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

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

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

print 'Number of rows before load: ',ROWS

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

commit

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

print 'Number of rows after load: ',ROWS

disconnect

End Test
```

E.3 Acid_functions.tst

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

```
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

print 'Creating history table'

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

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

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


print 'creating the sleep procedure'

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

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

print 'creating the Acid Transaction'

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

execute{ CREATE PROCEDURE acid_transaction(
                IN  o_key      INT,
                IN  l_key      INT,
                IN  delta      INT,
                OUT rprice     Numeric(18,8),
                OUT quantity   INT,
                OUT tax        Numeric(18,8),
                OUT disc       Numeric(18,8),
                OUT extprice   Numeric(18,8),
                OUT ototal     Numeric(18,8)
                )
ON EXCEPTION RESUME
BEGIN
 DECLARE pkey          INT ;
 DECLARE skey          INT ;
 DECLARE cost          NUMERIC(18,8) ;
 DECLARE new_extprice   NUMERIC(18,8) ;
 DECLARE new_ototal     NUMERIC(18,8) ;
 DECLARE new_quantity   INT ;
 DECLARE c_sqlstate     char(5);
 DECLARE num            INT ;

 LOOP1: LOOP
 COMMIT;

 acid1:
 BEGIN ATOMIC

   SELECT o_totalprice
     INTO ototal
     FROM orders
    WHERE o_orderkey = o_key ;
   SELECT l_quantity,
          l_extendedprice,
          l_partkey,
          l_suppkey,
          l_tax,
          l_discount
     INTO quantity,
          extprice,
          pkey,
          skey,
          tax,
          disc
     FROM lineitem
    WHERE l_orderkey   = o_key
      AND l_linenumber = l_key;
   -- CLEAN UP IMPRECICE NUMBERS
   SET ototal = ototal - "TRUNCATE"("truncate"(extprice*(1-
disc),2)*(1+tax),2);
   SET rprice = "TRUNCATE"((extprice  / quantity),2);
   SET cost = "TRUNCATE"((rprice * delta),2);
   SET new_extprice = extprice + cost;
   SET new_ototal = "TRUNCATE"(new_extprice * (1.0 - disc),2);
   SET new_ototal = "TRUNCATE"(new_ototal * (1.0 + tax),2);
   SET new_ototal = ototal + new_ototal ;
   SET new_quantity = quantity + delta ;

   --
   -- Update LineItem
   --
   UPDATE lineitem
      SET l_quantity      = new_quantity,
          l_extendedprice = new_extprice
    WHERE l_orderkey=o_key
      AND l_linenumber=l_key;
   SELECT SQLSTATE INTO c_sqlstate;
   IF c_sqlstate = '00000' THEN
       --
       -- Update Orders
       --
       UPDATE orders
          SET o_totalprice = new_ototal
        WHERE o_orderkey = o_key;
       SELECT SQLSTATE INTO c_sqlstate;
       IF c_sqlstate = '00000'  THEN
           INSERT INTO history VALUES ( pkey, skey, o_key, l_key,
delta, now(*)) ;
           SELECT SQLSTATE INTO c_sqlstate;
           IF c_sqlstate = '00000'
                          then message 'Completed ',o_key,' .......';
                END IF;
            END IF;
   END IF;

 END acid1;

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

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

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

END LOOP LOOP1;

-- commit ;
 RETURN(0);
END;
}

print 'Acid transaction created'
print ' '

print 'Creating Acid query'

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

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

print 'Acid query created'
print ' '

print 'Creating Generate_acid_values function'

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

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

BEGIN

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

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

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

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

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

          close random;
          close get_order;
 end loop loop1;

 set orderkey = out_key;

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

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

END
}
commit

print 'Generate_acid_values function created'
print ' '

print 'Creating Generate_Ps_Values function'

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

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

BEGIN

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

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

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

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

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

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

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

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

set partkey = out_key;
close random;
close get_part;

END
}
commit

print 'Generate_Ps_Values function created'
print ' '



print 'Creating Generate_acid_values2 function'

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

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

BEGIN

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

 declare times cursor for
        select datediff(millisecond,convert(char(10),getdate(),
116),now(*));
 declare random1 cursor for select rand(seed);
 declare random cursor for select rand();

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

 open random1;
 fetch next random1 into rand_dbl;

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

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

   end loop loop2;
 end loop loop1;
 commit;

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


END
}
commit

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

E.4 Run_atomicity

```
#!/bin/ksh
cd $ACID_ROOT/atomicity
dbtest $ACID_ROOT/atomicity/acid_atomic_main.tst >
$ACID_RESULTS/acid_atomic_main.out

roll_back=1

rm -f $ACID_RESULTS/atomc $ACID_RESULTS/atomr

while read line
do
        if [ roll_back -eq 1 ]
        then
                commit_started=`echo $line |grep "Starting atom-
icity test with commit"`
                if [ ! -z "$commit_started" ]
                then
                        roll_back=0
```

```
			echo "$line" > $ACID_RESULTS/atomc
		else
			echo "$line" >>
$ACID_RESULTS/atomr
		fi
	else
		echo "$line" >> $ACID_RESULTS/atomc
	fi
done < $ACID_RESULTS/acid_atomic_main.out
mv $ACID_RESULTS/atomr $ACID_RESULTS/atomr.`date
'+%y%m%d_%H%M%S'`
mv $ACID_RESULTS/atomc $ACID_RESULTS/atomc.`date
'+%y%m%d_%H%M%S'`
```

E.5 Acid_consistency_main.tst

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

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

stringconnect "dsn=qual_15_0;"

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

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

%run test 'acid_consistency_setup.tst'

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

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

let i = 1
LOOP {
	if i > 22 then { BREAK LOOP } endif
		let ot_file = "acid_consist_user", i, ".ot"
		let my_str = "stream=", i
		print ot_file, my_str
		start test '-o' ot_file my_str 'acid_consistency_txn.tst'
		sleep 900
		let i = i + 1
} ENDLOOP


synchronize 23

% let the log flush...
sleep 900
stringconnect "dsn=qual_15_0;"
%include 'acid_consistency_query.tst'
run test '-o' 'acid_consistency_q2.ot' 'acid_consistency_query.tst'
execute {select now(*)} into times
print 'Consistency test end = ', times
print ' '

End Test
```

E.6 Acid_consistency_query.lst

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

stringconnect "dsn=qual_15_0;"

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

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

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

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

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

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

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

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

disconnect

END Test




Acid_consistency_setup.lstTest
		"acid_consistency_setup.tst"
Description		"Creates acid_table table"

stringconnect "dsn=qual_15_0;"

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

% Drop Table if found
allow error -141
execute { drop table acid_table }
execute {drop table latest}
```

```
allow no error

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

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

print 'acid_table created'
execute {create table latest(stream int ,last int null) on SYSTEM }
LOOP ({let j = 1}; {j <= 22}; {let j = j + 1})
   {
            execute { insert into latest(stream,last) values (^,0) }
                        substitute j
   } endloop
commit

print 'latest created'

LOOP ({let i = 1}; {i <= 22}; { let i = i + 1})
{
  lOOP ({let j = 1}; {j <= 100}; {let j = j + 1})
   {
            execute { call generate_acid_values()} into ordr, line, delta
            execute { insert into acid_table values (^,^,^,^,^) }
                        substitute i,j,ordr,line,delta
   } endloop
  print (j-1)*i
} endloop

commit

fetch {select count(*) from acid_table } into ROWS
assert ROWS = 2200
print 'Number of rows after load: ',ROWS

End Test
```

E.7 Acid_consistency_setup.tst

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

stringconnect "dsn=qual_15_0;"

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

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

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

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

print 'acid_table created'
execute {create table latest(stream int ,last int null) on SYSTEM }
LOOP ({let j = 1}; {j <= 22}; {let j = j + 1})
   {
            execute { insert into latest(stream,last) values (^,0) }
                        substitute j
   } endloop
commit

print 'latest created'

LOOP ({let i = 1}; {i <= 22}; { let i = i + 1})
{
  lOOP ({let j = 1}; {j <= 100}; {let j = j + 1})
   {
            execute { call generate_acid_values()} into ordr, line, delta
            execute { insert into acid_table values (^,^,^,^,^) }
                        substitute i,j,ordr,line,delta
   } endloop
  print (j-1)*i
} endloop

commit

fetch {select count(*) from acid_table } into ROWS
assert ROWS = 2200
print 'Number of rows after load: ',ROWS

End Test
```

E.8 Acid_durability_main.tst

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

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

stringconnect "dsn=qual_15_0;"

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

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

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

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

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

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

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

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

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

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

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

End Test
```

E.9 Acid_durability_query.tst

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

stringconnect "dsn=qual_15_0;"

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

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

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

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

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

  fetch {select cast(^ as numeric(12,2)) } substitute o_price into
o_price
  fetch {select cast(^ as numeric(12,2)) } substitute o_total into l_total
  print 'orderkey = ', ord, '      o_totalprice = ', o_price,
        '           acid query = ' , l_total

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

 } ENDLOOP

disconnect

END Test
```

E.10 Acid_durability_setup.tst

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

stringconnect "dsn=qual_15_0;"

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

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

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

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

print 'acid_table created'

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

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

print 'latest created'
```

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

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

End Test
```

E.11 Acid_durability_txn.tst

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

stringconnect "dsn=qual_15_0;"
execute {select now(*)} into times
print 'Durability test start = ', times
print ' '
print 'stream          trans.      o_key      l_key       p_key      s_key
            delta       date_t '
allow no error

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

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

 if i=101 then {
     let smallest=0
     allow error -210
            commit
     execute { set temporary option isolation_level=1 }
     execute { select min(last) from latest } into smallest
     execute { set temporary option isolation_level=3 }
     commit
     LOOP{
          if smallest >= 100 then {
                                    print 'Stream ', stream,
                                            ' Entering the Second phase
with delays'
                                    break loop}
                          endif
          let sleep_time =10
          sleep sleep_time
                commit
                execute { set temporary option isolation_level=1 }
          execute { select min(last) from latest } into smallest
                execute { set temporary option isolation_level=3 }
                commit
     }
     ENDLOOP
     allow no error
     let commit_delay=5}
 endif


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

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

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

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

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

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

```
 execute { set temporary option isolation_level=3 }

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

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


End Test
```

E.12 Acid_functions.tst

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

print 'Creating history table'

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

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

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


print 'creating the sleep procedure'

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

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

print 'creating the Acid Transaction'

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

execute{ CREATE PROCEDURE acid_transaction(
               IN  o_key      INT,
               IN  l_key      INT,
               IN  delta      INT,
               OUT rprice     Numeric(18,8),
               OUT quantity   INT,
               OUT tax        Numeric(18,8),
               OUT disc       Numeric(18,8),
               OUT extprice   Numeric(18,8),
               OUT ototal     Numeric(18,8)
               )
ON EXCEPTION RESUME
BEGIN
 DECLARE pkey           INT ;
 DECLARE skey           INT ;
 DECLARE cost           NUMERIC(18,8) ;
 DECLARE new_extprice   NUMERIC(18,8) ;
 DECLARE new_ototal     NUMERIC(18,8) ;
 DECLARE new_quantity   INT ;
 DECLARE c_sqlstate     char(5);
 DECLARE num            INT ;

 LOOP1: LOOP

 COMMIT;

 acid1:
 BEGIN ATOMIC

   SELECT o_totalprice
     INTO ototal
     FROM orders
    WHERE o_orderkey = o_key ;
    SELECT l_quantity,
          l_extendedprice,
          l_partkey,
          l_suppkey,
          l_tax,
          l_discount
     INTO quantity,
          extprice,
          pkey,
          skey,
          tax,
          disc
     FROM lineitem
    WHERE l_orderkey   = o_key
      AND l_linenumber = l_key;
    -- CLEAN UP IMPRECICE NUMBERS
    SET ototal = ototal - "TRUNCATE"("truncate"(extprice*(1-
disc),2)*(1+tax),2);
    SET rprice = "TRUNCATE"((extprice  / quantity),2);
    SET cost = "TRUNCATE"((rprice * delta),2);
    SET new_extprice = extprice + cost;
    SET new_ototal = "TRUNCATE"(new_extprice * (1.0 - disc),2);
    SET new_ototal = "TRUNCATE"(new_ototal * (1.0 + tax),2);
    SET new_ototal = ototal + new_ototal ;
    SET new_quantity = quantity + delta ;

    --
    -- Update LineItem
    --
    UPDATE lineitem
       SET l_quantity      = new_quantity,
           l_extendedprice = new_extprice
     WHERE l_orderkey=o_key
```

```
      AND l_linenumber=l_key;
   SELECT SQLSTATE INTO c_sqlstate;
   IF c_sqlstate = '00000' THEN
      --
      -- Update Orders
      --
      UPDATE orders
        SET o_totalprice = new_ototal
       WHERE o_orderkey = o_key;
      SELECT SQLSTATE INTO c_sqlstate;
      IF c_sqlstate = '00000'  THEN
         INSERT INTO history VALUES ( pkey, skey, o_key, l_key,
delta, now(*)) ;
         SELECT SQLSTATE INTO c_sqlstate;
         IF c_sqlstate = '00000'
                     then message 'Completed ',o_key,' .....';
            END IF;
          END IF;
   END IF;

 END acid1;

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

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

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

END LOOP LOOP1;

-- commit ;
 RETURN(0);
END;
}

print 'Acid transaction created'
print ' '

print 'Creating Acid query'

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

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

print 'Acid query created'
print ' '

print 'Creating Generate_acid_values function'

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

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

BEGIN

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

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

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

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

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

          close random;
          close get_order;
 end loop loop1;

 set orderkey = out_key;

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

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

END
}
commit

print 'Generate_acid_values function created'
print ' '

print 'Creating Generate_Ps_Values function'

allow error -265
```

```
execute { DROP PROCEDURE generate_ps_values }
allow no error

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

BEGIN

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

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

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

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

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

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

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

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

 set partkey = out_key;
 close random;
 close get_part;

END
}
commit

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

```
print 'Creating Generate_acid_values2 function'

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

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

BEGIN

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

 declare times cursor for
             select datediff(millisecond,convert(char(10),getdate(),
116),now(*));
 declare random1 cursor for select rand(seed);
 declare random cursor for select rand();

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

 open random1;
 fetch next random1 into rand_dbl;

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

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

   end loop loop2;
 end loop loop1;
 commit;

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


END
}
commit

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

E.13 Acid_isolation_main1.tst

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

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

stringconnect "dsn=qual_15_0;"

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

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

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

End Test
```

E.14 Acid_isolation_main2.tst

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

%  Created Date: 5/24/1999

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

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

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

stringconnect "dsn=qual_15_0;"

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


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

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

End Test
```

E.15 Acid_isolation_main3.txt

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

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

stringconnect "dsn=qual_15_0;"

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

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

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

End Test

E.16 Acid_isolation_main4.txt

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

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

stringconnect "dsn=qual_15_0;"

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

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

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

End Test

E.17 Acid_isolation_main5.tst

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

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

stringconnect "dsn=qual_15_0;"

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

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

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

End Test

E.18 Acid_isolation_main6.tst

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

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

stringconnect "dsn=qual_15_0;"

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

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

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

End Test
```

E.19 Acid_isolation_setup.tst

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

stringconnect "dsn=qual_15_0;"

% Drop Table if found

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

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

execute {checkpoint}

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

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

print 'Number of rows before load: ',ROWS

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

commit

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

print 'Number of rows after load: ',ROWS

disconnect

End Test
```

E.20 Acid_isolation_test1.tst

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

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

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

stringconnect "dsn=qual_15_0;"

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

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

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

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

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

End Test
```

E.21 Acid_isolation_test1_query.tst

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

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

```
stringconnect "dsn=qual_15_0;"

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

 execute {select ordr from acid_isolation_table}
        into ordr

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

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

disconnect

END Test
```

E.22 Acid_isolation_test2.tst

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

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

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

stringconnect "dsn=qual_15_0;"

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

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

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

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

End Test
```

E.23 Acid_ isolation_test2_query.tst

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

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

stringconnect "dsn=qual_15_0;"

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

 execute {select ordr from acid_isolation_table}
        into ordr

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

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

disconnect

END Test
```

E.24 Acid_isolation_test3_transaction1.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%  Created by: Masood Dirin                                %
%  Created Date: 5/25/1999                                 %
```

```
%  Script name: tpcd_acid_isolation_test3_transaction1.tst
%
%  ------------------------------------------------------------------  %
%                                                                      %
%  This test could be run by itself, but it is recommended to run it as
%
%  part of tpcd_acid_isolation_main3.tst file.                         %
%                                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

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

stringconnect "dsn=qual_15_0;"

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

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

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

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

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

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

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

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

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

 fetch { select cast(^ as numeric(18,6))
                        + (cast(^ as numeric(18,6))*(cast (^ as nu-
meric(18,6))
                        /cast (^ as numeric(18,6)))) }
             substitute extprice, delta, extprice, quantity
             into result1
 % make it format nicely...
 execute { select cast(^ as numeric(18,2)) } substitute result1 into re-
sult2

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

 disconnect
 End Test
```

E.25 Acid_isolation_test3_transaction2.tst

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


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

stringconnect "dsn=qual_15_0;"

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

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

SYNCHRONIZE 2

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

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

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

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

print ' '

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

```
print ' '

 fetch { select cast(^ as numeric(18,6))
             + (cast(^ as numeric(18,6))*(cast (^ as numeric(18,6))
             /cast (^ as numeric(18,6)))) }
       substitute extprice, delta, extprice, quantity
       into result1
 % make it format nicely...
 execute { select cast(^ as numeric(18,2)) } substitute result1 into re-
sult2

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

End Test
```

E.26 Acid_isolation_test4_transaction1.tst

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

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

stringconnect "dsn=qual_15_0;"

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

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

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

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

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

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

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

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

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

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

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

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

 disconnect
 End Test
```

E.27 Acid_isolation_test4_transaction2.tst

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


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

stringconnect "dsn=qual_15_0;"

execute {select ordr, line, delta from acid_isolation_table}
       into ordr, line, delta
% generate a new set of values; we only use delta2
```

```
execute { call generate_acid_values()} into ordr2, line2, delta2

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

SYNCHRONIZE
sleep 5000

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

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

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

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

 fetch { select cast(^ as numeric(18,6))
              + (cast(^ as numeric(18,6))*(cast (^ as numeric(18,6))
              /cast (^ as numeric(18,6)))) }
       substitute extprice, delta2, extprice, quantity
       into result1
 % make it format nicely...
 execute { select cast(^ as numeric(18,2)) } substitute result1 into re-
sult2

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

End Test
```

E.28 Acid_isolation_test5_query.tst

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Created by: Masood Dirin
%  Created Date: 5/27/1999
%  Script name: tpcd_acid_isolation_query_test5.tst
%  ------------------------------------------------------------------
%
%  This test could be run by itself, but it is recommended to run
% it as part of tpcd_acid_isolation_main5.tst file.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

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

stringconnect "dsn=qual_15_0;"

synchronize 2

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

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

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

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

print ' '

End Test
```

E.29 Acid_isolation_test5_transaction1.tst

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

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

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

stringconnect "dsn=qual_15_0;"


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

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

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

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

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

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

print ' '
execute {select * from history where h_o_key=^
                and h_date_t=(select max(h_date_t) from history where
h_o_key=^)}
                substitute ordr, ordr
                into hpk, hsk, hok, hlk, hda, hdt
print 'User 1 history entry:'
print '  h_p_key = ', hpk
print '  h_s_key = ', hsk
print '  h_o_key = ', hok
print '  h_l_key = ', hlk
print '  h_delta = ', hda
print '  h_date_t = ', hdt

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

End Test
```

E.30 Acid_isolation_test6_query.tst

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

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

stringconnect "dsn=qual_15_0;"

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

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

End Test
```

E.31 Acid_isolation_test6_transaction1.tst

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

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

stringconnect "dsn=qual_15_0;"

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

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

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

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

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

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

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

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

End Test
```

Appendix - F Pricing Information

International Business Machines Corporation *11500 BURNET RD*
AUSTIN TX 78758

November 20, 2009

Dear Lotus,

Here is the requested quote for the System IBM Power 595 Server for the TPC-H benchmark using IBM System Storage DS4800.

Description	Part No.	Source	Unit Price	Qty	Ext Price	Maint Price
Server Hardware						
Server 1:9119 Model 595	9119-FHA	1	91,000	1	91,000	63,192
GX Dual-port 12x HCA	1816	1	4,000	6	24,000	
2.5M Enhanced 12X Cable (to connect to a gx card in	1831	1	650	12	7,800	
146.8 GB Ultra320 15K RPM	3279	1	1,299	5	6,495	
0/8 - core POWER6 5.0GHz CoD, 0-core Active	4695	1	67,700	8	541,600	99,840
Single Processor Activation for FC 4695 (5.0 GHz)	4755	1	33,900	64	2,169,600	430,080
256GB Bundle DDR2 Activations	5681	1	83,200	2	166,400	
0/8GB DDR2 Memory (4x2GB) DIMMS - 667 MHz	5694	1	650	64	41,600	
2-PORT BASETX ETHT.PCI-X A	5767	1	699	1	699	
4.7 GB IDE Slimline DVD-RAM Drive	5757	1	660	1	660	
4 Gigabit Dual-port Fibre Channel PCI-E Adapter	5774	1	3,308	24	79,392	
PCI-X DDR Dual - x4 SAS Adapter	5912	1	1,092	1	1,092	
12xI/O Drawer PCI-E, SFF disk	5803	1	32,000	3	96,000	26,712
Bulk Power Regulator	6333	1	4,200	8	33,600	
Bulk Power Distribution Assembly	6334	1	2,500	4	10,000	
Slimline Doors (F&R), H CEC AND I/O Expansion Rac	6869	1	8,000	1	8,000	
UPIC Cable Group, BPD1 to I/O Drawer at A01	6941	1	500	1	500	
UPIC Cable Group, BPD1 to I/O Drawer A05	6942	1	500	1	500	
UPIC Cable Group, BPD1 to I/O Drawer A09	6943	1	500	1	500	
3rd & 4th AMDs + UPIC Y-Cable Group (BPC to Fans)	6952	1	5,000	1	5,000	
UPIC Y-Cable Group BPD1 to 1st Processor Node	6961	1	650	1	650	
UPIC Y-Cable Group BPD1 to 2nd Processor Node	6962	1	650	1	650	
UPIC Y-Cable Group BPD1 to 3rd Processor Node	6963	1	650	1	650	
UPIC Y-Cable Group BPD1 to 4th Processor Node	6964	1	650	1	650	
UPIC Y-Cable Group BPD1 to 5th Processor Node	6965	1	650	1	650	
UPIC Y-Cable Group BPD1 to 6th Processor Node	6966	1	650	1	650	
UPIC Y-Cable Group BPD1 to 7th Processor Node	6967	1	650	1	650	
UPIC Y-Cable Group BPD1 to 8th Processor Node	6968	1	650	1	650	
Ethernet Cable, 15M, Hardware Management	7802	1	34	2	68	
Line Cord, 6AWG/Type W, 14ft, IEC309 100A Plug	8686	1	2,500	4	10,000	
HMC 1:7042-C07 Desktop Hardw.Mgmt.Console	7042-C07	1	1,830	1	1,830	1,344
IBM T117 TFT 17 inch Color Display	3645	1	875	1	875	
Quiet Touch Keyboard - USB, Business Black,	5951	1	104	1	104	
Power Cord (6-foot), To Wall (125V, 15A), Plug Type #	6470	1	18	2	36	
Ethernet Cable, 6M, Hardware Management	7801	1	15	1	15	
Mouse - Business Black with Keyboard Attachment	8841	1	78	1	78	
PCI-e 1Gb Ethernet UTP 2Port	5767	1	699	1	699	
				Subtotal	**3,303,343**	**621,168**

Storage						
DS4800 Disk System Model 82 (4 GB Cache)	1815-82A	1	53,995	12	647,940	
DS4800 8-Storage Partitions	8870	1	10,000	12	120,000	
(22R4255) DS4800 AIX Host Kit	7711	1	7,000	12	84,000	
DS4000 EXP810 Enclosure	1812-81A	1	6,000	48	288,000	
Short Wave SFP	2410	1	998	96	95,808	
16-PAK 4 GBPS 73GB GB/15K	5433	1	14,816	18	266,688	
Fiber Cable 25m	5625	1	189	48	9,072	
Fiber Cable 1m	5601	1	79	96	7,584	
3 Year Warranty Service Upgrade 1812-81A 24x7x4		1	960	48	0	46,080
3 Year Warranty Service Upgrade 1815-82A 24x7x4		1	3,200	12	0	38,400
ServicePac for 3-Year 24x7x4 Support (Rack)	41L2760	1	300	5	0	1,500
IBM S2 42U Standard Rack	93074RX	1	1,489	5	7,445	
				Subtotal	**1,526,537**	**85,980**
Server Software						
AIX 6.1 (media only)	5692-A6P	1	50	1	50	
AIX Software per Processor	5765-G62	1	2,495	64	159,680	
AIX IBM Management Edition for AIX	5765-AME	1	490	64	31,360	
AIX Software Maintanance (3Y)	73-SM3-474	1	2,836	64		181,504
AIX Software Maintanance 24x7 Upgrade (3Y)	73-SM3-476	1	732	64		46,848
AIX IBM Mgmt Edition of AIX Software Maintanance	(373-AME-989	1	196	64		12,544
HMC Software SUB (3Y)	5773-0570	1	236	1		236
HMC Software Support (3Y)	5773-0569	1	675	1		675
C for AIX user Lic+SW maint 12 MO	D5A1DLL	1	975	1	975	
C for AIX user annual SW maint renewal	E1A1FLL	1	195	2		390
				Subtotal	**192,065**	**242,197**
				Total	**5,021,945**	**949,345**
				Total IBM Discounts*		**-2,981,645**
					0	
					Three-Year Cost of Ownership	**2,989,645**

Please note that the discount used is based on US list prices for similar quantities & configurations including pre-payment for maintenance. The discount applies to the totality of this quote and it does not apply to individual items listed.

For more information on:

All products: http://www.ibm.com/products

For Storage products: http://www-03.ibm.com/systems/storage/disk/index.html

For System p products: http://www-03.ibm.com/systems/p

For additional information, please contact me directly:

Dan Hebrank

IBM Sales & Distribution STG Sales
1-314-283-4674

For:

Company IBM
Contact Lotus Douglas
Phone (512) 286-6005
Fax lotus@us.ibm.com
Address 11500 Burnet Road, Austin, TX 78758

Quotation for Software and Support

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

	Catalogue Number	Product Description	License Type	Machine	P/S	List Price Per Unit	Quantity	Price	Discount	Extended Price
1	11518	Sybase IQ Single App Svr, per cpu core	CP	IBM AIX	P	2,595	64	166,080	24,912	141,168.00
3	98477	3 yr support Single App Svr, per cpu core				1,713	64	109,632	16,445	93,187.20
4		Discounts:								
5		15.00%								
6										
7										
8										
9										
10										
11										
12										

Quote Date: ***11/23/09***
Valid thru:

Total 234,355.20

Licence + 3 year support

Payment terms : Net 30 Days