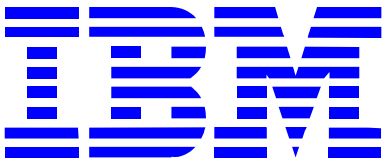


TPC Benchmark™ H Full Disclosure Report

IBM @server OpenPower 720

Using

IBM DB2 Universal Database V8.2

Second Edition

February 9, 2005

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
the IBM logo
@server
pSeries
POWER5
OpenPower
TotalStorage
DB2
DB2 Universal Database

Second Edition February 9, 2005

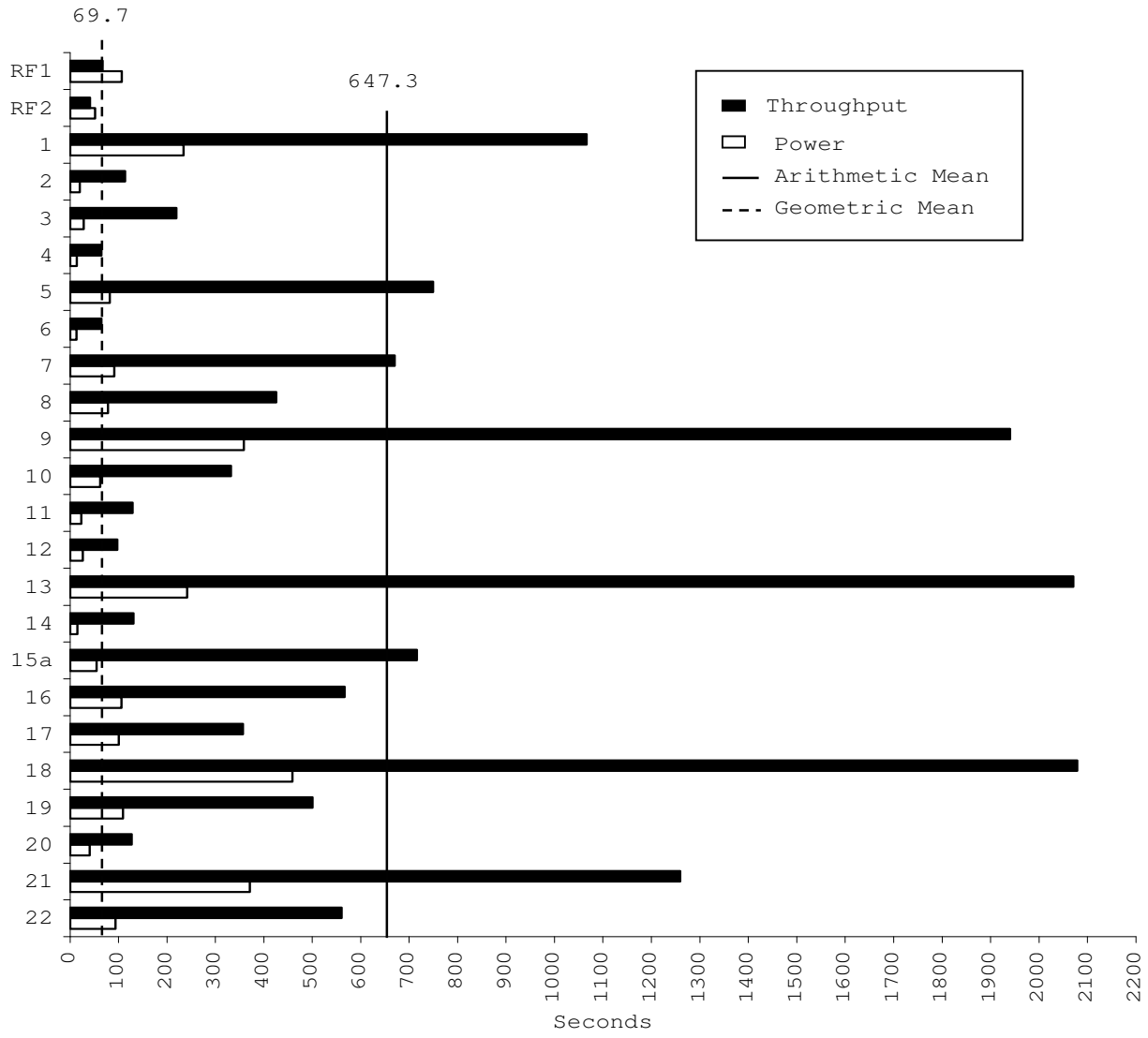
IBM @server OpenPower 720 with DB2 UDB V8.2

TPC-H Rev. 2.1.0

Report Date: Dec. 16, 2004
Revision Date: Feb. 9, 2005

Total System Cost	Composite Query per Hour Rating	Price/Performance
$484,157	12,006.8 QphH@300GB	40 $/QphH@300GB

Database Size	Database Manager	Operating System	Other Software	Availability Date
300 GB	DB2 UDB V8.2	SUSE LINUX Enterprise Server 9	None	January 28, 2005



Database Load Time: 3:26:33	Load included backup: Y	Total Data Storage/Database Size: 8.74
RAID (base tables): N	RAID (Base Tables and Auxiliary Data Structures): N	RAID (All): N

System Configurations

2 IBM @server OpenPower 720, each server with

Processors	4 x 1650MHz POWER5 with 2x 36MB L3 Cache
Memory	32GB
Disk Controllers	8 x 2Gigabit Fibre Channel PCI-X Adapters & 2 x IBM TotalStorage 7311 Model DS20 I/O drawers & 4 x IBM TotalStorage DS4300 dual controllers
Disk Drives	34 x 36.4GB 15K 2Gb FC drives; 3 x 36.4GB 10K Ultra320 SCSI drives on node 1, 1 x 36.4 GB 10K Ultra320 SCSI drives on node 2
Total Disk Storage	2,620.80 GB (GB is defined as 1024 * 1024 * 1024 bytes)

IBM @server OpenPower 720 with DB2 UDB V8.2

TPC-H Rev. 2.1.0

Report Date: Dec. 16, 2004
Revision Date: Feb. 9, 2005

Description	Part No.	Source	Unit Price	Qty	Ext Price	Maint Price
Server Hardware						
IBM eServer OpenPower 720	9124-720	1	50	2	100	2,736
Media Backplane Card	7877	1	120	2	240	0
Power Supply	7889	1	775	4	3,100	0
2-way 1.65GHz POWER5 Processor Card	5262	1	6,590	4	26,360	0
CPU Power Regulator	7876	1	160	4	640	0
IDE Slimline DVD-ROM Drive	2640	1	149	2	298	0
8GB (4x2048MB) DIMMS	1942	1	6,596	8	52,768	0
36.4GB 10,000K RPM Ultra320 SCSI Disk Drive	3273	1	275	4	1,100	0
Line Cords	6458	1	14	4	56	0
Rack Mount Drawer Rail Kit	7914	1	150	2	300	0
Hardware Management Console - 3631, 6458, 7801, 8800, 8841	7310-C03	1	2251	1	2,251	1,152
Model D20 I/O Drawer	7311-D20	1	4,270	4	17,080	27,028
Remote I/O Cable, 3.5M	3147	1	550	6	3,300	0
SPCN 2m Cable: Drawer to Drawer	6001	1	25	6	150	0
AC Power Supply, 435W	6268	1	625	4	2,500	0
RIO-2 Ports to I/O Planar Riser Card	6417	1	800	4	3,200	0
2 Gigabit Fibre Channel PCI-X Adapter	5716	1	2,267	16	36,272	0
Rack Model T00 - 6098, 6246, 7176	7014-T00	1	4,188	2	8,376	1,536
				Subtotal	**158,091**	**32,452**
Storage						
Storage Server DS4300	1722-60U	1	14,999	8	119,992	19,992
SHORT WAVE SFP GBIC	2210	1	499	16	7,984	0
2GB FC, 36.4GB / 15K Drive Module	5212	1	1,115	68	75,820	0
Linux for POWER Host Kit	7714	1	4,000	1	4,000	0
5m Fibre Optic Cable	5605	1	129	16	2,064	0
DS4300 4-Storage Partitions Activation	8804	1	3,000	8	24,000	0
				Subtotal	**233,860**	**19,992**
Software						
HMC Software Support 3 Year	5773-RS3	1	675	1	675	0
HMC Software Support 24x7 3 Year	5773-0570	1	236	1	0	236
DB2 UDB ESE 8.2 SW Lic. & Maint. 12 months for OpenPower 720		1	22,260	8	178,080	0
DB2 UDB EXE SW Maint. Renewal 1 year for OpenPower 720		1	1,060	16	0	16,960
DB2 UDB ESE DPF 8.2 SW Lic. & Maint. 12 months for OpenPower 720		1	6,686	8	53,488	0
DB2 UDB ESE DPF SW Maint. Renewal 1 year for OpenPower 720		1	318	16	0	5,088
				Subtotal	**232,243**	**22,284**
Third Party Software						
SUSE LINUX Enterprise Server 9 for POWER, to 16 CPUs/ YR		2	1,299	6	7,794	0
SUSE LINUX Enterprise Server 9 SW Media Kit for IBM POWER		2	35	2	70	0
Novell SUSE LINUX Server Support 24x7x4		2	900	6	0	5,400
				Subtotal	**7,864**	**5,400**
				Total	**632,058**	**80,128**
				IBM Total System Discounts*		**-228029**
				Three-Year Cost of Ownership		**484,157**
					QphH	**12,006**
					$/QphH	**40**

*Discounts are based on US list prices for similar quantities & configurations

Pricing Sources:

1 IBM: Bala Ekambaram, WW Marketing Manager, OpenPower Offerings bala.ekambaram@us.ibm.com, 1-512-838-7137

1 IBM: Jonathan Prial, Vice President, Marketing, Information Management, jonprial@us.ibm.com, 1-914-766-1576

2 Novell SLES 9: http://www.novell.com/products/linuxenterpriseserver/pricing.html

2 Novell 24x7x4 Support: http://support.novell.com/linux/linux_server_support.html

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

The System as configured for the test was available on January 28, 2005.

Prices used in the 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 based on assumptions about past or future purchases are not permitted. All discounts reflect standard pricing policies for the listed components. For complete details see the pricing sections of the TPC benchmark specifications. If you find 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)	=	300
Total Data Storage/Database Size	=	8.74
Start of Database Load	=	12/7/2004 19:07:16
End of Database Load	=	12/7/2004 22:33:49
Database Load Time	=	3:26:33
Query Streams for Throughput Test	=	6
TPC-H Power Metric (QppH@300GB)	=	15,502.5
TPC-H Throughput Metric (QthH@300GB)	=	9,299.4
Composite Query-per-Hour Rating (QphH@300GB)	=	12,006.8
Total System Price over 3 years	=	$484,157
TPC-H Price/Performance Metric ($/QphH@300GB)	=	40

Measurement Intervals

Measurement Interval in Throughput Test (Ts) = 15,330 seconds

Duration of Stream Execution

Stream Id	Seed Used	Query Start Query End	Date/Time Date/Time	RF1 Start RF1 End	Date/Time Date/Time	RF2 Start RF2 End	Date/Time Date/Time	Duration
Stream 00	1206223349	12/6/2004 12/6/2004	23:14:53 23:58:37	12/6/2004 12/6/2004	23:13:06 23:14:53	12/6/2004 12/6/2004	23:58:37 23:59:29	0:46:23
Stream 01	1206223350	12/6/2004 12/7/2004	23:59:34 4:04:11	12/6/2004 12/7/2004	23:59:40 4:05:53	12/7/2004 12/7/2004	4:05:53 4:06:44	4:04:37
Stream 02	1206223351	12/6/2004 12/7/2004	23:59:34 4:02:08	12/7/2004 12/7/2004	4:06:44 4:07:37	12/7/2004 12/7/2004	4:07:37 4:08:13	4:02:34
Stream 03	1206223352	12/6/2004 12/7/2004	23:59:34 3:49:55	12/7/2004 12/7/2004	4:08:13 4:09:05	12/7/2004 12/7/2004	4:09:05 4:09:39	3:50:21
Stream 04	1206223353	12/6/2004 12/7/2004	23:59:34 3:48:00	12/7/2004 12/7/2004	4:09:39 4:10:31	12/7/2004 12/7/2004	4:10:31 4:11:05	3:48:26
Stream 05	1206223354	12/6/2004 12/7/2004	23:59:35 3:58:45	12/7/2004 12/7/2004	4:11:05 4:12:27	12/7/2004 12/7/2004	4:12:27 4:13:25	3:59:10
Stream 06	1206223355	12/6/2004 12/7/2004	23:59:34 3:58:28	12/7/2004 12/7/2004	4:13:25 4:14:29	12/7/2004 12/7/2004	4:14:29 4:15:04	3:58:54

Timing Interval (in seconds)

Stream ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Stream 00	234.4	19.9	28.4	14.1	82.4	13.6	91.1	78.0	358.5	62.2	22.9	26.2
Stream 01	535.9	149.3	340.4	14.7	679.6	47.5	633.1	505.1	2293.5	459.3	77.3	66.2
Stream 02	1188.8	171.8	199.4	46.0	640.4	114.3	741.4	331.8	2247.7	240.0	137.8	156.9
Stream 03	865.7	93.5	231.7	50.0	848.7	52.1	593.1	650.0	2138.0	219.2	79.3	114.3
Stream 04	927.1	67.7	196.3	32.4	886.6	48.6	700.8	390.6	1510.9	369.9	87.8	89.7
Stream 05	1390.3	119.5	113.1	157.0	687.4	70.0	666.7	277.2	1663.7	270.8	164.6	80.7
Stream 06	1487.8	78.1	236.8	85.8	755.6	55.9	685.5	398.8	1787.5	432.9	228.1	77.0
Minimum	535.9	67.7	113.1	14.7	640.4	47.5	593.1	277.2	1510.9	219.2	77.3	66.2
Average	1065.9	113.3	219.6	64.3	749.7	64.7	670.1	425.6	1940.2	332.0	129.2	97.5
Maximum	1487.8	171.8	340.4	157.0	886.6	114.3	741.4	650.0	2293.5	459.3	228.1	156.9

Stream ID	Q13	Q14	Q15a	Q16	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream 00	242.2	15.6	54.9	105.6	100.6	459.1	109.2	40.7	370.8	93.8	107.0	51.3
Stream 01	1907.3	117.8	1018.4	516.2	221.6	2523.8	444.7	109.9	1469.6	545.3	102.0	51.7
Stream 02	1804.9	224.2	587.8	693.5	555.6	2291.1	604.3	119.8	847.5	608.9	52.6	35.6
Stream 03	1624.7	95.5	344.9	602.4	544.9	2362.5	449.7	114.5	1130.6	614.7	52.7	33.6
Stream 04	2290.2	104.2	371.6	528.0	314.8	1820.0	589.7	180.5	1550.6	647.2	52.4	33.7
Stream 05	2153.2	146.1	1132.2	487.0	231.9	1886.5	506.1	143.3	1493.7	509.5	81.8	58.6
Stream 06	2646.7	96.7	840.1	572.7	271.7	1588.9	407.6	95.0	1066.0	438.8	63.4	34.9
Minimum	1624.7	95.5	344.9	487.0	221.6	1588.9	407.6	95.0	847.5	438.8	52.4	33.6
Average	2071.2	130.8	715.8	566.6	356.8	2078.8	500.4	127.2	1259.7	560.7	67.5	41.4
Maximum	2646.7	224.2	1132.2	693.5	555.6	2523.8	604.3	180.5	1550.6	647.2	102.0	58.6

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

Benchmark Sponsor:

Haider Rizvi
Mgr., DB2 Data Warehouse Perf.
IBM Canada Ltd;
8200 Warden Avenue
Markham, Ontario L6G 1C7

Ray Venditti
Mgr, Linux Perf.
IBM eServer Performance
11501 Burnet Road
Austin, TX 78758

December 13, 2004

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

Platform: **IBM @server OpenPower 720 2-node cluster**

Database Manager: **IBM DB2 UDB V8.2**

Operating System: **SUSE LINUX Enterprise Server 9**

The results were:

CPU (Speed)	Memory	Disks	**QphH@300GB**
IBM @server OpenPower 720 2-node cluster (each node with)			
4 x POWER5 (1.65 GHz)	2 x 36 MB L3 32 GB Main	34 x 36.4 GB ext. 3 x 36.4 GB SCSI int. (node 1)	12,006.8

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:

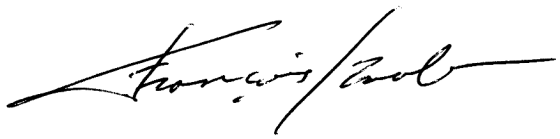
- The database records were defined with the proper layout and size
- The database population was generated using DBGEN
- The database was properly scaled to 300GB and populated accordingly
- The compliance of the database auxiliary data structures was verified
- The database load time was correctly 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 an approved 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 6 query streams
- The ratio between the longest and the shortest query was such that no query timing was adjusted
- The execution times for queries and refresh functions were correctly measured and reported
- The repeatability of the measured results was verified
- The required amount of database log was configured
- The system pricing was verified for major components and maintenance
- The major pages from the FDR were verified for accuracy

Additional Audit Notes:

While the priced system includes a sets of 16 x 2 GB DIMMs on each node, in the tested system only one of the two nodes was configured as such. The other node was configured with a set of 8 x 4 GB DIMMs. According to the results of additional experiments and performance measurements it is my opinion that there are no significant performance differences between these two memory configurations.

Respectfully Yours,



François Raab
President

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

Abstract

This report documents the full disclosure information required by the TPC Benchmark™ H Standard Specification Revision 2.1.0 dated August 14, 2003 for measurements on the IBM @server OpenPower 720. The software used includes SUSE LINUX Enterprise Server 9 operating system with DB2 Universal Database V8.2.

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 2.1.0) on August 14, 2003. This is the full disclosure report for benchmark testing of the IBM @server OpenPower 720 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.*

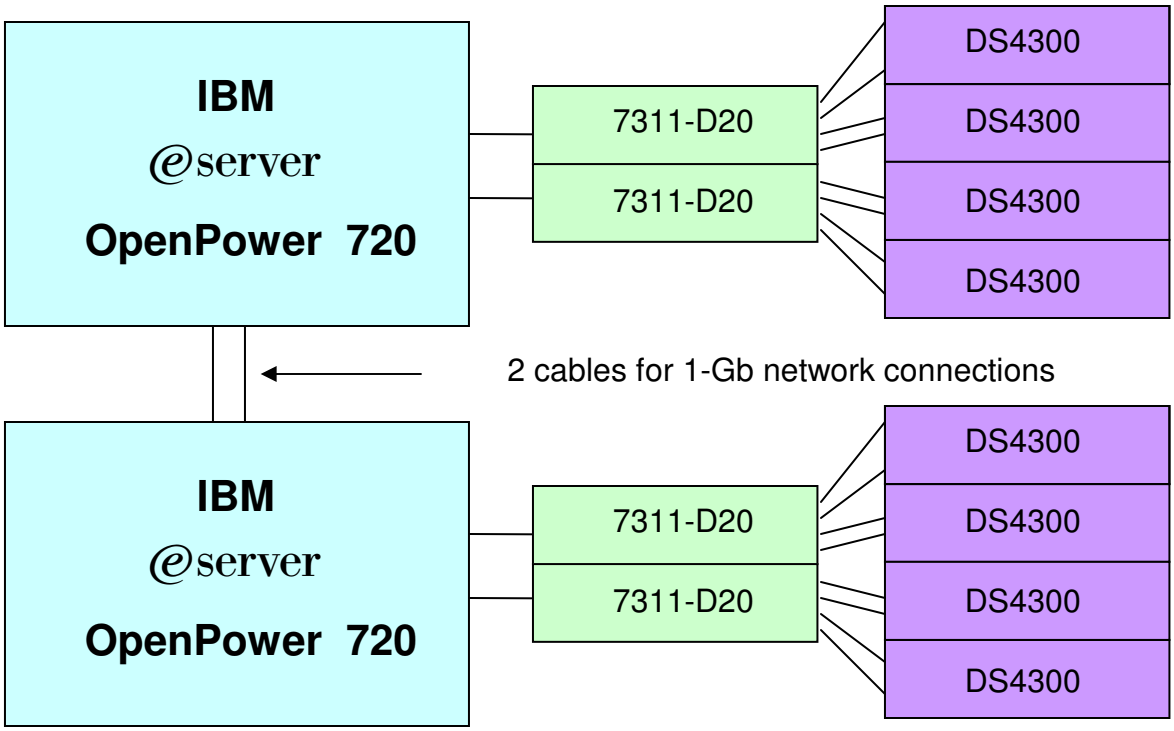
Appendix A "Tunable Parameters" contains a list of all DB2 parameters and operating system parameters. Session initialization parameters can be set during or immediately after establishing the connection to the database within the tpcdbatch program documented in Appendix D "Driver Source Code". This 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 @server OpenPower 720 Benchmark Configuration



The system was a 2-node cluster of **IBM @server** OpenPower 720 systems, each with:

- 4 POWER5 1650MHz processors with 2 x 36MB L3 cache
- 32 GB of memory
- 8 IBM 2Gb Fibre Channel PCI-X Adapters
- 3 36.4 GB 15K RPM Internal disk drives (node 1)
- 34 36.4 GB 15K RPM external disk drives
- 2 IBM TotalStorage 7311-DS20 controllers
- 4 IBM TotalStorage DS4300 dual controllers

For full details of the priced configuration, see the pricing spreadsheet in the Executive Summary.

2.0 Clause 1: Logical Database Design Related Items

Appendix B "Database Build 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.

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 "Database Build 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 "Database Build 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 all tables except for the nation and region tables, see Appendix B "Database Build 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 2: 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 1.3.0 and DBGEN 1.3.0 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 1.3.0 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.

Query variant Q15a was used in this disclosure.

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

1. Table names are fully qualified. For example, the nation table is referred to as "TPCD.NATION".
2. The standard IBM SQL date syntax is used for date arithmetic.
   For example: DATE('1996-01-01') + 3 MONTHS.
3. 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 C3, "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 D "Driver Source Code".

4.0 Clause 3: 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 @server OpenPower 720. 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.

SELECT DECIMAL(SUM(DECIMAL(INTEGER(INTEGER(DECIMAL

(INTEGER(100*DECIMAL(L_EXTENDEDPRICE,20,3)),20,3)*

(1-L_DISCOUNT)) * (1+L_TAX)),20,3)/100.0),20,3)

FROM TPCD.LINEITEM WHERE L_ORDERKEY = okey

SELECT DECIMAL(SUM(O_TOTALPRICE, 20, 3)) from TPCD.ORDERS WHERE O_ORDERKEY = okey

4.2.2. Consistency Tests

Verify that the ORDER and LINEITEM tables are initially consistent as defined in Clause 3.3.2.1, based on a random sample of at least 10 distinct values of O_ORDERKEY.

The queries defined in 4.2.1 , "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 6 streams of 100 ACID transactions each, the queries defined in 4.2.1 , "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: Start an ACID transaction with a randomly selected O_KEY,L_KEY and DELTA. The transaction is delayed for 60 seconds just prior to the Commit.
2. 2nd session: Start an ACID query for the same O_KEY as in the ACID transaction.

3. 2nd session: The ACID query attempts to read the file but is locked out by the ACID transaction waiting to complete.
4. 1st session: The ACID transaction is released and the Commit is executed releasing the record. With the LINEITEM record now released, the ACID query can now complete.
5. 2nd session: Verify that the ACID query delays for approximately 60 seconds and that the results displayed for the ACID query match the input for the ACID transaction.

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: Perform the ACID transaction for a random O_KEY, L_KEY and DELTA. The transaction is delayed for 60 seconds just prior to the Rollback.
2. 2nd session: Start an ACID query for the same O_KEY as in the ACID transaction. The ACID query attempts to read the LINEITEM table but is locked out by the ACID transaction.
3. 1st session: The ACID transaction is released and the Rollback is executed, releasing the read.
4. 2nd session: With the LINEITEM record now released, the ACID query completes.

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: Start an ACID transaction T1 for a randomly selected O_KEY, L_KEY and DELTA. The transaction is delayed for 60 seconds just prior to the COMMIT.
2. 2nd session: Start a second ACID transaction T2 for the same O_KEY, L_KEY, and for a randomly selected DELTA2. This transaction is forced to wait while the 1st session holds a lock on the LINEITEM record requested by the second session.
3. 1st session: The ACID transaction T1 is released and the Commit is executed, releasing the record. With the LINEITEM record now released, the ACID transaction T2 can now complete.
4. Verify 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: Start an ACID transaction T1 for a randomly selected O_KEY, L_KEY, and DELTA. The transaction is delayed for 60 seconds just prior to the rollback.
2. 2nd session: Start a second ACID transaction T2 for the same O_KEY, L_KEY used by the 1st session. This transaction is forced to wait while the 1st session holds a lock on the LINEITEM record requested by the second session.
3. 1st session: Rollback the ACID transaction T1. With the LINEITEM record now released, the ACID transaction T2 completes.
4. Verify that T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE

4.3.5. Isolation Test 5

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

1. 1st session: Start 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: Start a second ACID transaction, T2, which selects random values of PS_PARTKEY and PS_SUPPKEY and returns all columns of the PARTSUPP table for which PS_PARTKEY and PS_SUPPKEY are 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 executes TPC-H query 1 with DELTA=0, 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. 3rd session: Before T1 completed, a transaction T3, which executes modified TPC-H query 1 with a randomly selected value of DELTA (not equal to 0), was started.
4. T1 completed.
5. T2 completed.
6. T3 completed.
7. It was verified that the appropriate rows in the ORDERS, LINEITEM and HISTORY tables were changed.

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. Loss of System Power

This test was 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 7 execution streams was started such that each stream executes a different set of transactions.
4. The system was shut down by powering off the system after at least 100 ACID transactions had completed.
5. The system was powered back on and rebooted, and the database was restarted.
6. 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.
7. Consistency condition described in 4.2.1 was verified.

4.4.2. Permanent Failure of a Single Log Disk

This test was 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 7 execution streams was started such that each stream executes a different set of transactions.
4. One of the log disks containing the DB2 transaction log recovery data 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. 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.
7. Consistency condition described in 4.2.1 was verified.

4.4.3. Permanent Failure of a Single Data Disk

This test was 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 7 execution streams was started such that each stream executes a different set of transactions.
4. One of the RAID 0 disks containing the database table data, database index and temporary space was removed from the enclosure after at least 100 ACID transactions had completed from each of the execution streams.
5. The RAID 0 disks were rebuilt and the database was restored.
6. 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.
7. Consistency condition described in 4.2.1 was verified.

4.4.4. Loss of Network Connection

This test was 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 7 execution streams was started such that each stream executes a different set of transactions.
4. The Gigabit network cables was disconnected from the system.
5. Database detected the network loss and terminated processing.
6. Network connections were reestablished and the database was restarted.
7. 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.
8. Consistency condition described in 4.2.1 was verified.

5.0 Clause 4: 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	1,799,989,091
Orders	450,000,000
Customer	45,000,000
Supplier	3,000,000
Part	60,000,000
Partsupp	240,000,000
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.

DB2 was configured on a 2-node cluster of IBM @server OpenPower 720 servers. Each node has:

- 8 IBM 2Gb Fibre Channel PCI-X adapters
- 34 36.4GB external disk drives.
- 3 36.4GB internal drives (node 1)

The IBM @server OpenPower 720 connects to two IBM TotalStorage 7311-DS20 controllers via eight FC adapters. Each 7311-DS20 controller in turn connects to two IBM TotalStorage DS4300 dual controllers. In each IBM TotalStorage DS4300 dual controller, there are two RAID 0 4-disk arrays, where permanent tables, their auxiliary data structures and temporary tablespaces reside. In one DS4300 dual controller, it has an additional 2 disks with RAID 1 for the database log. See Appendix B "Database Build Scripts". Two nodes are connected with 2 cables for 1-Gb network connections.

The Operating System and database software reside on an internal disk. Two additional disks were used for the benchmark executable programs, results and file system log. The second node only has one internal disk to host the operarting system and database software.

5.3. Mapping of Database Partitions/Replications

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

The database was not replicated. The database was logically partitioned into 8 logical data partitions.

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 0 was used for database tables and indexes. RAID level 1 was used for 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 1.3.0 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 3:26:33.

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:

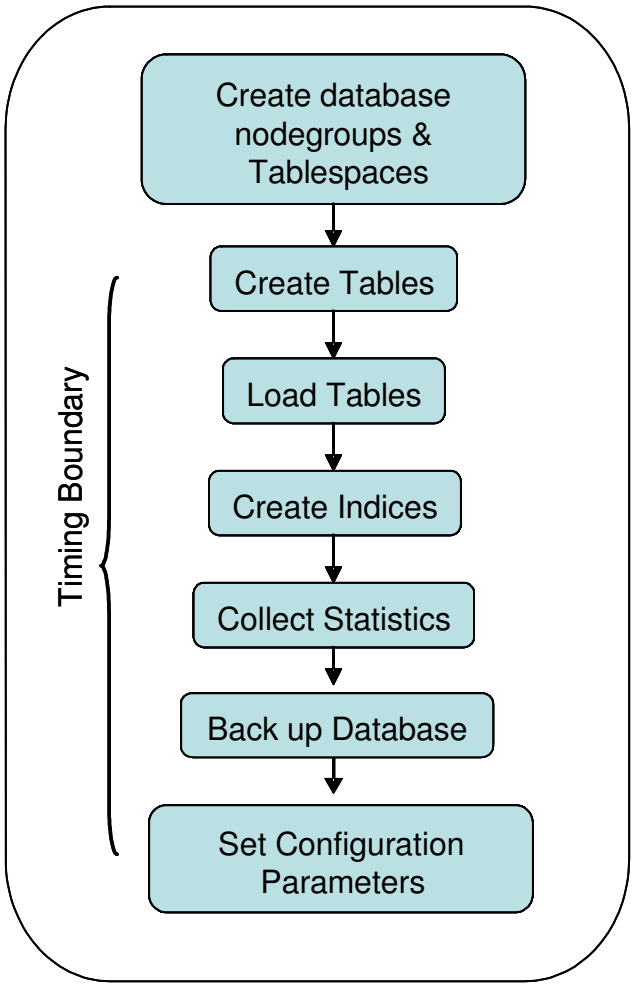
Disk Type	Number of Disks	Space per Disk	Sub-Total Disk Space	Database Size	Data Storage Ratio
2Gb FC 36.4GB	68	36.40 GB	2475.20 GB		
Ultra320 SCSI 36.4 GB	4	36.40 GB	145.60 GB		
Total			2,620.80 GB	300 GB	8.74

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 "Database Build Scripts" contains the programs and input files used to load the database.

Database Load Procedure:



6.0 Clause 5: Performance Metrics and Execution-Rules Related Items

6.1. System Activity Between Load and Performance Tests

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

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

6.2. Steps in the Power Test

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

The following steps were used to implement the power test:

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

6.3. Timing Intervals for Each Query and Refresh Function

The timing intervals for each query of the measured set and for both 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@300GB	QthH@300GB	QphH@300GB
Run 1	15,502.5	9,299.4	12,006.8
Run 2	15,525.8	9,286.1	12,007.3

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 6: 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 is invoked by calling tpcdbatch with the stream number 0 specified. The refresh function is initiated as a separate call to tpcdbatch with the SQL script for the refresh functions.

The throughput test is invoked by initiating a call to tpcdbatch for every query stream that will be run. Tpcdbatch gets the stream number for each of the streams, and the SQL file specific to that stream number as the queries to execute. The refresh function is initiated as a separate call to tpcdbatch with the SQL script for the refresh functions and the total number of query streams specified.

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 implementation specific layer is a single executable SQL application that uses embedded dynamic SQL to process the EQT generated by QGEN. The application is called tpcdbatch to indicate that it processes a batch of TPC-H queries, although it is completely capable of processing any arbitrary SQL statement (both DML and DDL).

A separate instance of tpcdbatch is invoked for each stream. Each instance establishes a distinct connection to the database server through which the EQT is transmitted to the database and the results are returned through the implementation specific layer to the driver. When an instance of tpcdbatch is invoked, it is provided with a context of whether it is running a power test, query stream or refresh stream, as well as an input file containing the 22 queries and/or refresh functions. tpcdbatch then connects to the database, performs any session initialization as well as preparing output files required by the auditor. Then it proceeds to read from the input file and processes each query or refresh function in turn.

For queries, each query is prepared, described, and a cursor is opened and used to fetch the required number of rows. After the last row has been retrieved a commit is issued. For the refresh functions, during the database build all data is first split for each node using the db2split utility. For RF1, the data for each node is further split into n equal portions for both the lineitem and orders tables taking care that the records for the same orderkey remain in the same set. For RF2, the data for each node is further split into m equal portions. During the run, when tpcdbatch encounters a call to execute RF1, it first calls a shell script which loads these n sets of data into staging tables. Then tpcdbatch forks off q children (where q * y = n), with each child to do y sets of insert with fullselect from the staging tables into the original lineitem and orders tables. When tpcdbatch encounters a call to execute RF2, it calls a shell script that loads these data into a single staging table. Then tpcdbatch forks off p children (where p * x =m) to do x sets of deletes from the orders and lineitem tables with a subselect from the staging table.

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 7: 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 32% 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:,

Bala Ekambaram, WW Marketing Manager, OpenPower Offerings
email: bala.ekambaram@us.ibm.com, phone 1-512-838-7137
Jonathan Prial, Vice President, Marketing, Information Management
email: jonprial@us.ibm.com, phone 1-914-766-1576

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 (hardware components: server and storage, operating system software, and database software) is available immediately.

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 DB2 Database Configuration

get database configuration for TPCD

Database Configuration for Database TPCD

```
 Database configuration release level            = 0x0a00
 Database release level                          = 0x0a00

 Database territory                              = US
 Database code page                              = 819
 Database code set                               = ISO8859-1
 Database country/region code                    = 1
 Database collating sequence                     = BINARY
 Alternate collating sequence            (ALT_COLLATE) =

 Dynamic SQL Query management         (DYN_QUERY_MGMT) =
DISABLE

 Discovery support for this database      (DISCOVER_DB) = ENABLE

 Default query optimization class       (DFT_QUERYOPT) = 7
 Degree of parallelism                  (DFT_DEGREE) = 1
 Continue upon arithmetic exceptions   (DFT_SQLMATHWARN) = NO
 Default refresh age                (DFT_REFRESH_AGE) = 0
 Default maintained table types for opt (DFT_MTTB_TYPES) =
SYSTEM
 Number of frequent values retained    (NUM_FREQVALUES) = 0
 Number of quantiles retained          (NUM_QUANTILES) = 300

 Backup pending                                  = NO

 Database is consistent                          = YES
 Rollforward pending                             = NO
 Restore pending                                 = NO

 Multi-page file allocation enabled              = YES

 Log retain for recovery status                  = RECOVERY
 User exit for logging status                    = NO

 Data Links Token Expiry Interval (sec)     (DL_EXPINT) = 60
 Data Links Write Token Init Expiry Intvl(DL_WT_IEXPINT) = 60
 Data Links Number of Copies            (DL_NUM_COPIES) = 1
 Data Links Time after Drop (days)       (DL_TIME_DROP) = 1
 Data Links Token in Uppercase             (DL_UPPER) = NO
 Data Links Token Algorithm               (DL_TOKEN) = MAC0

 Database heap (4KB)                        (DBHEAP) = 20000
 Size of database shared memory (4KB)  (DATABASE_MEMORY) =
AUTOMATIC
 Catalog cache size (4KB)            (CATALOGCACHE_SZ) = 386
 Log buffer size (4KB)                  (LOGBUFSZ) = 256
 Utilities heap size (4KB)             (UTIL_HEAP_SZ) = 10000
 Buffer pool size (pages)                (BUFFPAGE) = 5000
 Extended storage segments size (4KB)    (ESTORE_SEG_SZ) =
16000
 Number of extended storage segments   (NUM_ESTORE_SEGS) = 0
 Max storage for lock list (4KB)           (LOCKLIST) = 20480

 Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) =
5000
 Percent of mem for appl. group heap   (GROUPHEAP_RATIO) = 70
 Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 2058

 Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) =
(SHEAPTHRES)
 Sort list heap (4KB)                     (SORTHEAP) = 5000
 SQL statement heap (4KB)                 (STMTHEAP) = 20000
 Default application heap (4KB)         (APPLHEAPSZ) = 16000
 Package cache size (4KB)               (PCKCACHESZ) = 640
 Statistics heap size (4KB)           (STAT_HEAP_SZ) = 10000

 Interval for checking deadlock (ms)      (DLCHKTIME) = 5000
 Percent. of lock lists per application     (MAXLOCKS) = 5
 Lock timeout (sec)                     (LOCKTIMEOUT) = -1

 Changed pages threshold            (CHNGPGS_THRESH) = 60
 Number of asynchronous page cleaners  (NUM_IOCLEANERS) = 6
 Number of I/O servers                (NUM_IOSERVERS) = 12
 Index sort flag                        (INDEXSORT) = YES
 Sequential detect flag                 (SEQDETECT) = YES
 Default prefetch size (pages)      (DFT_PREFETCH_SZ) = 16

 Track modified pages                    (TRACKMOD) = OFF

 Default number of containers                    = 1
 Default tablespace extentsize (pages)  (DFT_EXTENT_SZ) = 32

 Max number of active applications        (MAXAPPLS) = 40
 Average number of active applications     (AVG_APPLS) = 1
 Max DB files open per application        (MAXFILOP) = 1024

 Log file size (4KB)                    (LOGFILSIZ) = 16384
 Number of primary log files             (LOGPRIMARY) = 20
 Number of secondary log files           (LOGSECOND) = 2
 Changed path to log files              (NEWLOGPATH) =
 Path to log files                               = /dev/raw/raw42
 Overflow log path                  (OVERFLOWLOGPATH) =
 Mirror log path                    (MIRRORLOGPATH) =
 First active log file                           = S0000002.LOG
 Block log on disk full             (BLK_LOG_DSK_FUL) = NO
 Percent of max active log space by transaction(MAX_LOG) = 0
 Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0

 Group commit count                      (MINCOMMIT) = 1
 Percent log file reclaimed before soft chckpt (SOFTMAX) = 500
 Log retain for recovery enabled         (LOGRETAIN) = RECOVERY
 User exit for logging enabled           (USEREXIT) = OFF

 HADR database role                              = STANDARD
 HADR local host name               (HADR_LOCAL_HOST) =
 HADR local service name             (HADR_LOCAL_SVC) =
 HADR remote host name              (HADR_REMOTE_HOST) =
 HADR remote service name            (HADR_REMOTE_SVC) =
 HADR instance name of remote server  (HADR_REMOTE_INST) =
 HADR timeout value                 (HADR_TIMEOUT) = 120
 HADR log write synchronization mode     (HADR_SYNCMODE) =
NEARSYNC

 First log archive method           (LOGARCHMETH1) =
LOGRETAIN
 Options for logarchmeth1             (LOGARCHOPT1) =
 Second log archive method           (LOGARCHMETH2) = OFF
 Options for logarchmeth2             (LOGARCHOPT2) =
 Failover log archive path          (FAILARCHPATH) =
 Number of log archive retries on error   (NUMARCHRETRY) = 5
 Log archive retry Delay (secs)      (ARCHRETRYDELAY) = 20
 Vendor options                      (VENDOROPT) =

 Auto restart enabled                (AUTORESTART) = ON
 Index re-creation time and redo index build  (INDEXREC) = ACCESS
 Log pages during index build         (LOGINDEXBUILD) = OFF
 Default number of loadrec sessions    (DFT_LOADREC_SES) = 1
```

```
 Number of database backups to retain   (NUM_DB_BACKUPS) = 12
 Recovery history retention (days)      (REC_HIS_RETENTN) = 366

 TSM management class                    (TSM_MGMTCLASS) =
 TSM node name                           (TSM_NODENAME) =
 TSM owner                               (TSM_OWNER) =
 TSM password                            (TSM_PASSWORD) =

 Automatic maintenance                   (AUTO_MAINT) = OFF
   Automatic database backup             (AUTO_DB_BACKUP) = OFF
   Automatic table maintenance           (AUTO_TBL_MAINT) = OFF
     Automatic runstats                  (AUTO_RUNSTATS) = OFF
     Automatic statistics profiling      (AUTO_STATS_PROF) = OFF
       Automatic profile updates         (AUTO_PROF_UPD) = OFF
     Automatic reorganization            (AUTO_REORG) = OFF
```

A.2 DB2 Database Manager Configuration

get database manager configuration

```
        Database Manager Configuration

    Node type = Enterprise Server Edition with local and remote clients

 Database manager configuration release level            = 0x0a00

 CPU speed (millisec/instruction)             (CPUSPEED) = 2.000000e-
06
 Communications bandwidth (MB/sec)      (COMM_BANDWIDTH) =
1.000000e+01

 Max number of concurrently active databases     (NUMDB) = 1
 Data Links support                      (DATALINKS) = NO
 Federated Database System Support          (FEDERATED) = NO
 Transaction processor monitor name        (TP_MON_NAME) =

 Default charge-back account          (DFT_ACCOUNT_STR) =

 Java Development Kit installation path       (JDK_PATH) =
/opt/IBMJava2-ppc64-141

 Diagnostic error capture level             (DIAGLEVEL) = 3
 Notify Level                          (NOTIFYLEVEL) = 3
 Diagnostic data directory path              (DIAGPATH) =

 Default database monitor switches
   Buffer pool                         (DFT_MON_BUFPOOL) = OFF
   Lock                                 (DFT_MON_LOCK) = OFF
   Sort                                (DFT_MON_SORT) = OFF
   Statement                             (DFT_MON_STMT) = OFF
   Table                               (DFT_MON_TABLE) = OFF
   Timestamp                            (DFT_MON_TIMESTAMP) = OFF
   Unit of work                          (DFT_MON_UOW) = OFF
 Monitor health of instance and databases   (HEALTH_MON) = OFF

 SYSADM group name                       (SYSADM_GROUP) =
 SYSCTRL group name                      (SYSCTRL_GROUP) =
 SYSMAINT group name                     (SYSMAINT_GROUP) =
 SYSMON group name                       (SYSMON_GROUP) =

 Client Userid-Password Plugin          (CLNT_PW_PLUGIN) =
 Client Kerberos Plugin                (CLNT_KRB_PLUGIN) =
 Group Plugin                          (GROUP_PLUGIN) =
 GSS Plugin for Local Authorization    (LOCAL_GSSPLUGIN) =
 Server Plugin Mode                    (SRV_PLUGIN_MODE) =
UNFENCED
 Server List of GSS Plugins       (SRVCON_GSSPLUGIN_LIST) =
 Server Userid-Password Plugin         (SRVCON_PW_PLUGIN) =
 Server Connection Authentication           (SRVCON_AUTH) =
NOT_SPECIFIED
 Database manager authentication         (AUTHENTICATION) =
SERVER
 Cataloging allowed without authority   (CATALOG_NOAUTH) = NO
 Trust all clients                     (TRUST_ALLCLNTS) = YES
 Trusted client authentication         (TRUST_CLNTAUTH) = CLIENT
 Bypass federated authentication           (FED_NOAUTH) = NO

 Default database path                      (DFTDBPATH) = /home/db2inst1

 Database monitor heap size (4KB)          (MON_HEAP_SZ) = 90
 Java Virtual Machine heap size (4KB)      (JAVA_HEAP_SZ) = 1024
 Audit buffer size (4KB)                 (AUDIT_BUF_SZ) = 0
 Size of instance shared memory (4KB)  (INSTANCE_MEMORY) =
AUTOMATIC
 Backup buffer default size (4KB)           (BACKBUFSZ) = 1024
 Restore buffer default size (4KB)          (RESTBUFSZ) = 1024

 Sort heap threshold (4KB)                (SHEAPTHRES) = 140000

 Directory cache support                    (DIR_CACHE) = YES

 Application support layer heap size (4KB)   (ASLHEAPSZ) = 15
 Max requester I/O block size (bytes)         (RQRIOBLK) = 32767
 Query heap size (4KB)                   (QUERY_HEAP_SZ) = 1000

 Workload impact by throttled utilities(UTIL_IMPACT_LIM) = 10

 Priority of agents                        (AGENTPRI) = SYSTEM
 Max number of existing agents             (MAXAGENTS) = 200
 Agent pool size                      (NUM_POOLAGENTS) = 64
 Initial number of agents in pool       (NUM_INITAGENTS) = 4
 Max number of coordinating agents     (MAX_COORDAGENTS) =
(MAXAGENTS - NUM_INITAGENTS)
 Max no. of concurrent coordinating agents  (MAXCAGENTS) =
MAX_COORDAGENTS
 Max number of client connections      (MAX_CONNECTIONS) =
MAX_COORDAGENTS

 Keep fenced process                     (KEEPFENCED) = YES
 Number of pooled fenced processes         (FENCED_POOL) =
MAX_COORDAGENTS

 Initial number of fenced processes     (NUM_INITFENCED) = 0

 Index re-creation time and redo index build  (INDEXREC) = ACCESS

 Transaction manager database name         (TM_DATABASE) =
1ST_CONN
 Transaction resync interval (sec)     (RESYNC_INTERVAL) = 180

 SPM name                                  (SPM_NAME) =
 SPM log size                         (SPM_LOG_FILE_SZ) = 256
 SPM resync agent limit                (SPM_MAX_RESYNC) = 20
 SPM log path                           (SPM_LOG_PATH) =

 TCP/IP Service name                       (SVCENAME) = db2cdb2inst1
 Discovery mode                          (DISCOVER) = SEARCH
 Discover server instance                (DISCOVER_INST) = ENABLE

 Maximum query degree of parallelism   (MAX_QUERYDEGREE) =
ANY
 Enable intra-partition parallelism     (INTRA_PARALLEL) = NO

 No. of int. communication buffers(4KB)(FCM_NUM_BUFFERS) =
8192
 Number of FCM request blocks              (FCM_NUM_RQB) = 4096
```

```
 Number of FCM connection entries      (FCM_NUM_CONNECT) =
AUTOMATIC
 Number of FCM message anchors         (FCM_NUM_ANCHORS) =
AUTOMATIC

 Node connection elapse time (sec)         (CONN_ELAPSE) = 10
 Max number of node connection retries (MAX_CONNRETRIES) = 5
 Max time difference between nodes (min) (MAX_TIME_DIFF) = 60

 db2start/db2stop timeout (min)        (START_STOP_TIME) = 10
```

A.3 DB2 Registry Settings

```
DB2_SCATTERED_IO=Y
DB2_LGPAGE_BP=yes
DB2NOLIOAIO=no
DB2_EXTENDED_OPTIMIZATION=Y
DB2_ANTIJOIN=Y
DB2OPTIONS=-t -v +c
DB2COMM=tcpip
DB2_PARALLEL_IO=*
DB2AUTOSTART=FALSE
```

A.4 Linux Parameters

```
net.ipv4.ip_forward = 0
net.ipv4.conf.default.rp_filter = 1
kernel.sem = 500 512000 64 2048
kernel.msgmni = 1024
kernel.shmmax = 4278190080
kernel.shmall =  4278190080
fs.file-max = 131072
net.core.rmem_max = 131071
net.core.wmem_max = 131071
kernel.msgmax = 65536

echo 1700 > /proc/sys/vm/nr_hugepages

ifconfig eth0 mtu 9000
ifconfig eth2 mtu 9000

rpc.nfsd 32

Used glibc-64-bit-9-200407271122.
```

Appendix - B Database Build Scripts

B.1 db2nodes.cfg

```
0 192.168.0.152 0 192.168.0.152
1 192.168.0.152 1 192.168.0.152
2 192.168.0.152 2 192.168.0.152
3 192.168.0.152 3 192.168.0.152
4 192.168.0.152 5 192.168.1.152
5 192.168.0.152 6 192.168.1.152
6 192.168.0.152 7 192.168.1.152
7 192.168.0.152 8 192.168.1.152
8 192.168.0.153 0 192.168.0.153
9 192.168.0.153 1 192.168.0.153
10 192.168.0.153 2 192.168.0.153
11 192.168.0.153 3 192.168.0.153
12 192.168.0.153 5 192.168.1.153
13 192.168.0.153 6 192.168.1.153
14 192.168.0.153 7 192.168.1.153
15 192.168.0.153 8 192.168.1.153
```

B.2 run_db2set.pl

```
#!/usr/bin/perl


@SETTINGS=(
“DB2_SCATTERED_IO=Y”,
"DB2_LGPAGE_BP=yes",
"DB2NOLIOAIO=no",
"DB2COMM=tcpip",
"DB2AUTOSTART=FALSE",
"DB2_EXTENDED_OPTIMIZATION=Y",
"DB2_ANTIJOIN=Y",
"DB2OPTIONS=\"-t -v +c\"",
"DB2_PARALLEL_IO=*");

foreach $_ (@SETTINGS)
{
      system "db2set $_";
}
print "db2set settings updated\n";
```

B.3 load_dbm_cfg.sql

```
update dbm cfg using
NUM_POOLAGENTS 4
SVCENAME db2cdb2inst1
NUMDB 1
CPUSPEED -1
DIAGLEVEL 3
NOTIFYLEVEL 0
SHEAPTHRES 40000
MAX_QUERYDEGREE ANY
INTRA_PARALLEL YES
FCM_NUM_BUFFERS 10000
FCM_NUM_RQB 10000
HEALTH_MON OFF
DFT_MON_TIMESTAMP OFF;
```

B.4 load_db_cfg.sql

```
update db cfg for tpcd using
NUM_IOCLEANERS  6
NUM_IOSERVERS   12
LOGFILSIZ       8192
LOGPRIMARY      10
LOGSECOND       200
LOGBUFSZ        2048
LOGRETAIN       NO
DBHEAP          4000
DFT_QUERYOPT    7
DFT_DEGREE      2
NUM_FREQVALUES  0
NUM_QUANTILES   600
BUFFPAGE        10000
SORTHEAP        10000
UTIL_HEAP_SZ    5000
APPGROUP_MEM_SZ 5000;
```

B.5 create_nodegroups

```
create nodegroup catalog_node on node(0);
```

B.6 bufferpools.ddl

```
alter bufferpool IBMDEFAULTBP size 25000;
create bufferpool BP32KTEMP ALL NODES SIZE 10000 PAGESIZE
32K;
create bufferpool BP32K size 70000 NUMBLOCKPAGES 40000
BLOCKSIZE 16 pagesize 32K;
commit;
terminate;
db2stop;
db2start;
```

B.7 dss.ddl300GB.tbsp

```
DROP TABLESPACE TPCDTEMP;


create regular tablespace small_tables
in nodegroup catalog_node
managed by system
using ('/vol3/small_tables_mln')
on node(0)
prefetchsize 256
overhead 12.67
transferrate 0.18;

CREATE TEMPORARY TABLESPACE TPCDTEMP
PAGESIZE 32K
MANAGED BY DATABASE
   USING( DEVICE '/dev/sdd1' 1048576) on NODE(0)
   USING( DEVICE '/dev/sde1' 1048576) on NODE(1)
   USING( DEVICE '/dev/sdg1' 1048576) on NODE(2)
   USING( DEVICE '/dev/sdh1' 1048576) on NODE(3)
   USING( DEVICE '/dev/sdj1' 1048576) on NODE(4)
   USING( DEVICE '/dev/sdk1' 1048576) on NODE(5)
   USING( DEVICE '/dev/sdm1' 1048576) on NODE(6)
   USING( DEVICE '/dev/sdn1' 1048576) on NODE(7)
   USING( DEVICE '/dev/sdd1' 1048576) on NODE(8)
   USING( DEVICE '/dev/sde1' 1048576) on NODE(9)
   USING( DEVICE '/dev/sdg1' 1048576) on NODE(10)
   USING( DEVICE '/dev/sdh1' 1048576) on NODE(11)
   USING( DEVICE '/dev/sdj1' 1048576) on NODE(12)
   USING( DEVICE '/dev/sdk1' 1048576) on NODE(13)
```

```
   USING( DEVICE '/dev/sdm1' 1048576) on NODE(14)
   USING( DEVICE '/dev/sdn1' 1048576) on NODE(15)
     bufferpool    BP32KTEMP
     EXTENTSIZE 16
     PREFETCHSIZE 256
     overhead 12.67
     transferrate 0.18;
COMMIT WORK;

CREATE TABLESPACE line_tbsp
PAGESIZE 32K
MANAGED BY DATABASE
   USING( DEVICE '/dev/sdd2' 570000) on NODE(0)
   USING( DEVICE '/dev/sde2' 570000) on NODE(1)
   USING( DEVICE '/dev/sdg2' 570000) on NODE(2)
   USING( DEVICE '/dev/sdh2' 570000) on NODE(3)
   USING( DEVICE '/dev/sdj2' 570000) on NODE(4)
   USING( DEVICE '/dev/sdk2' 570000) on NODE(5)
   USING( DEVICE '/dev/sdm2' 570000) on NODE(6)
   USING( DEVICE '/dev/sdn2' 570000) on NODE(7)
   USING( DEVICE '/dev/sdd2' 570000) on NODE(8)
   USING( DEVICE '/dev/sde2' 570000) on NODE(9)
   USING( DEVICE '/dev/sdg2' 570000) on NODE(10)
   USING( DEVICE '/dev/sdh2' 570000) on NODE(11)
   USING( DEVICE '/dev/sdj2' 570000) on NODE(12)
   USING( DEVICE '/dev/sdk2' 570000) on NODE(13)
   USING( DEVICE '/dev/sdm2' 570000) on NODE(14)
   USING( DEVICE '/dev/sdn2' 570000) on NODE(15)
     bufferpool    BP32K
     EXTENTSIZE 16
     PREFETCHSIZE 256
     overhead 12.67
     transferrate 0.18;

COMMIT WORK;

CREATE TABLESPACE lineidx_tbsp
PAGESIZE 32K
MANAGED BY DATABASE
   USING( DEVICE '/dev/sdd3' 70000) on NODE(0)
   USING( DEVICE '/dev/sde3' 70000) on NODE(1)
   USING( DEVICE '/dev/sdg3' 70000) on NODE(2)
   USING( DEVICE '/dev/sdh3' 70000) on NODE(3)
   USING( DEVICE '/dev/sdj3' 70000) on NODE(4)
   USING( DEVICE '/dev/sdk3' 70000) on NODE(5)
   USING( DEVICE '/dev/sdm3' 70000) on NODE(6)
   USING( DEVICE '/dev/sdn3' 70000) on NODE(7)
   USING( DEVICE '/dev/sdd3' 70000) on NODE(8)
   USING( DEVICE '/dev/sde3' 70000) on NODE(9)
   USING( DEVICE '/dev/sdg3' 70000) on NODE(10)
   USING( DEVICE '/dev/sdh3' 70000) on NODE(11)
   USING( DEVICE '/dev/sdj3' 70000) on NODE(12)
   USING( DEVICE '/dev/sdk3' 70000) on NODE(13)
   USING( DEVICE '/dev/sdm3' 70000) on NODE(14)
   USING( DEVICE '/dev/sdn3' 70000) on NODE(15)
     bufferpool    BP32K
     EXTENTSIZE 16
     PREFETCHSIZE 256
     overhead 12.67
     transferrate 0.18;

COMMIT WORK;

CREATE TABLESPACE oth_tbsp
PAGESIZE 32K
MANAGED BY DATABASE
   USING( DEVICE '/dev/sdd5' 260000) on NODE(0)
   USING( DEVICE '/dev/sde5' 260000) on NODE(1)
   USING( DEVICE '/dev/sdg5' 260000) on NODE(2)
   USING( DEVICE '/dev/sdh5' 260000) on NODE(3)
   USING( DEVICE '/dev/sdj5' 260000) on NODE(4)
   USING( DEVICE '/dev/sdk5' 260000) on NODE(5)
   USING( DEVICE '/dev/sdm5' 260000) on NODE(6)
   USING( DEVICE '/dev/sdn5' 260000) on NODE(7)
   USING( DEVICE '/dev/sdd5' 260000) on NODE(8)
   USING( DEVICE '/dev/sde5' 260000) on NODE(9)
   USING( DEVICE '/dev/sdg5' 260000) on NODE(10)
   USING( DEVICE '/dev/sdh5' 260000) on NODE(11)
   USING( DEVICE '/dev/sdj5' 260000) on NODE(12)
   USING( DEVICE '/dev/sdk5' 260000) on NODE(13)
   USING( DEVICE '/dev/sdm5' 260000) on NODE(14)
   USING( DEVICE '/dev/sdn5' 260000) on NODE(15)
     bufferpool    BP32K
     EXTENTSIZE 16
     PREFETCHSIZE 256
     overhead 12.67
     transferrate 0.18;

COMMIT WORK;

CREATE TABLESPACE othidx_tbsp
PAGESIZE 32K
MANAGED BY DATABASE
   USING( DEVICE '/dev/sdd6' 55000) on NODE(0)
   USING( DEVICE '/dev/sde6' 55000) on NODE(1)
   USING( DEVICE '/dev/sdg6' 55000) on NODE(2)
   USING( DEVICE '/dev/sdh6' 55000) on NODE(3)
   USING( DEVICE '/dev/sdj6' 55000) on NODE(4)
   USING( DEVICE '/dev/sdk6' 55000) on NODE(5)
   USING( DEVICE '/dev/sdm6' 55000) on NODE(6)
   USING( DEVICE '/dev/sdn6' 55000) on NODE(7)
   USING( DEVICE '/dev/sdd6' 55000) on NODE(8)
   USING( DEVICE '/dev/sde6' 55000) on NODE(9)
   USING( DEVICE '/dev/sdg6' 55000) on NODE(10)
   USING( DEVICE '/dev/sdh6' 55000) on NODE(11)

   USING( DEVICE '/dev/sdj6' 55000) on NODE(12)
   USING( DEVICE '/dev/sdk6' 55000) on NODE(13)
   USING( DEVICE '/dev/sdm6' 55000) on NODE(14)
   USING( DEVICE '/dev/sdn6' 55000) on NODE(15)
     bufferpool    BP32K
     EXTENTSIZE 16
     PREFETCHSIZE 256
     overhead 12.67
     transferrate 0.18;

COMMIT WORK;


CREATE TEMPORARY TABLESPACE TPCDTEMP2
PAGESIZE 4K
MANAGED BY DATABASE
   USING( DEVICE '/dev/sdd7' 4020250) on NODE(0)
   USING( DEVICE '/dev/sde7' 4020250) on NODE(1)
   USING( DEVICE '/dev/sdg7' 4020250) on NODE(2)
   USING( DEVICE '/dev/sdh7' 4020250) on NODE(3)
   USING( DEVICE '/dev/sdj7' 4020250) on NODE(4)
   USING( DEVICE '/dev/sdk7' 4020250) on NODE(5)
   USING( DEVICE '/dev/sdm7' 4020250) on NODE(6)
   USING( DEVICE '/dev/sdn7' 4020250) on NODE(7)
   USING( DEVICE '/dev/sdd7' 4020250) on NODE(8)
   USING( DEVICE '/dev/sde7' 4020250) on NODE(9)
   USING( DEVICE '/dev/sdg7' 4020250) on NODE(10)
   USING( DEVICE '/dev/sdh7' 4020250) on NODE(11)
   USING( DEVICE '/dev/sdj7' 4020250) on NODE(12)
   USING( DEVICE '/dev/sdk7' 4020250) on NODE(13)
   USING( DEVICE '/dev/sdm7' 4020250) on NODE(14)
   USING( DEVICE '/dev/sdn7' 4020250) on NODE(15)
     bufferpool    IBMDEFAULTBP
     EXTENTSIZE 128
```

```
    PREFETCHSIZE 256
    overhead 12.67
    transferrate 0.18;

COMMIT WORK;

DROP TABLESPACE TEMPSPACE1;

COMMIT WORK;
```

B.8 create_UFtables.ddl

```
CREATE TABLE TPCDTEMP.ORDERS_NEW ( APP_ID INTEGER
NOT NULL,
                O_ORDERKEY       INTEGER NOT NULL,
                O_CUSTKEY        INTEGER NOT NULL,
                O_ORDERSTATUS    CHAR(1) NOT NULL,
                O_TOTALPRICE     FLOAT NOT NULL,
                O_ORDERDATE      DATE NOT NULL,
                O_ORDERPRIORITY  CHAR(15) NOT NULL,
                O_CLERK          CHAR(15) NOT NULL,
                O_SHIPPRIORITY   INTEGER NOT NULL,
                O_COMMENT        VARCHAR(79) NOT NULL WITH
DEFAULT)
    IN OTH_TBSP
    INDEX IN OTHIDX_TBSP
    PARTITIONING KEY (O_ORDERKEY);

CREATE TABLE TPCDTEMP.ORDERS_DEL ( APP_ID INTEGER
NOT NULL,
                O_ORDERKEY       INTEGER NOT NULL)
    IN OTH_TBSP
    INDEX IN OTHIDX_TBSP
    PARTITIONING KEY (O_ORDERKEY);

CREATE TABLE TPCDTEMP.LINEITEM_NEW ( APP_ID INTEGER
NOT NULL,
                L_ORDERKEY    INTEGER NOT NULL,
                L_PARTKEY     INTEGER NOT NULL,
                L_SUPPKEY     INTEGER NOT NULL,
                L_LINENUMBER  INTEGER NOT NULL,
                L_QUANTITY    FLOAT NOT NULL,
                L_EXTENDEDPRICE  FLOAT NOT NULL,
                L_DISCOUNT    FLOAT NOT NULL,
                L_TAX         FLOAT NOT NULL,
                L_RETURNFLAG  CHAR(1) NOT NULL,
                L_LINESTATUS  CHAR(1) NOT NULL,
                L_SHIPDATE    DATE NOT NULL,
                L_COMMITDATE  DATE NOT NULL,
                L_RECEIPTDATE DATE NOT NULL,
                L_SHIPINSTRUCT CHAR(25) NOT NULL,
                L_SHIPMODE     CHAR(10) NOT NULL,
                L_COMMENT      VARCHAR(44) NOT NULL WITH
DEFAULT)
    IN LINE_TBSP
    INDEX IN LINEIDX_TBSP
    PARTITIONING KEY (L_ORDERKEY);

CREATE INDEX TPCDTEMP.I_ORDERS_NEW ON
TPCDTEMP.ORDERS_NEW
( APP_ID,
 O_ORDERKEY,
 O_CUSTKEY,
 O_ORDERSTATUS,
 O_TOTALPRICE,
 O_ORDERDATE,
 O_ORDERPRIORITY,
O_COMMENT);

CREATE INDEX TPCDTEMP.I_LINEITEM_NEW ON
TPCDTEMP.LINEITEM_NEW (APP_ID);

CREATE UNIQUE INDEX TPCDTEMP.I_ORDERS_DEL ON
TPCDTEMP.ORDERS_DEL
(APP_ID, O_ORDERKEY);

COMMIT WORK;

ALTER TABLE TPCDTEMP.ORDERS_NEW LOCKSIZE TABLE;
ALTER TABLE TPCDTEMP.ORDERS_DEL LOCKSIZE TABLE;
ALTER TABLE TPCDTEMP.LINEITEM_NEW LOCKSIZE TABLE;

COMMIT WORK;
```

B.9 create_tables.ddl

```
-------------------------------------------------------
-- Create Tables
-------------------------------------------------------
CREATE TABLE TPCD.NATION  ( N_NATIONKEY  INTEGER NOT
NULL,
                N_NAME       CHAR(25) NOT NULL,

                N_REGIONKEY  INTEGER NOT NULL,
                N_COMMENT    VARCHAR(152) NOT NULL)
    IN "SMALL_TABLES";

CREATE TABLE TPCD.REGION  ( R_REGIONKEY  INTEGER NOT
NULL,
                R_NAME       CHAR(25) NOT NULL,
                R_COMMENT    VARCHAR(152) NOT NULL)
    IN "SMALL_TABLES";

CREATE TABLE TPCD.PART  ( P_PARTKEY     INTEGER NOT
NULL,
                P_NAME        VARCHAR(55) NOT NULL,
                P_MFGR        CHAR(25) NOT NULL,
                P_BRAND       CHAR(10) NOT NULL,
                P_TYPE        VARCHAR(25) NOT NULL,
                P_SIZE        INTEGER NOT NULL,
                P_CONTAINER   CHAR(10) NOT NULL,
                P_RETAILPRICE FLOAT NOT NULL,
                P_COMMENT     VARCHAR(23) NOT NULL )
    IN OTH_TBSP
    INDEX IN OTHIDX_TBSP;

CREATE TABLE TPCD.SUPPLIER ( S_SUPPKEY     INTEGER NOT
NULL,
                S_NAME       CHAR(25) NOT NULL,
                S_ADDRESS    VARCHAR(40) NOT NULL,
                S_NATIONKEY  INTEGER NOT NULL,
                S_PHONE      CHAR(15) NOT NULL,
                S_ACCTBAL    FLOAT NOT NULL,
                S_COMMENT    VARCHAR(101) NOT NULL)
    IN OTH_TBSP
    INDEX IN OTHIDX_TBSP;


CREATE TABLE TPCD.PARTSUPP ( PS_PARTKEY     INTEGER
NOT NULL,
                PS_SUPPKEY    INTEGER NOT NULL,
                PS_AVAILQTY   INTEGER NOT NULL,
                PS_SUPPLYCOST  FLOAT   NOT NULL,
                PS_COMMENT    VARCHAR(199) NOT NULL )
```

```
    IN OTH_TBSP
    INDEX IN OTHIDX_TBSP;

CREATE TABLE TPCD.CUSTOMER ( C_CUSTKEY     INTEGER
NOT NULL,
                    C_NAME        VARCHAR(25) NOT NULL,
                    C_ADDRESS     VARCHAR(40) NOT NULL,
                    C_NATIONKEY   INTEGER NOT NULL,
                    C_PHONE       CHAR(15) NOT NULL,
                    C_ACCTBAL     FLOAT    NOT NULL,
                    C_MKTSEGMENT  CHAR(10) NOT NULL,
                    C_COMMENT     VARCHAR(117) NOT NULL)
    IN OTH_TBSP
    INDEX IN OTHIDX_TBSP;

CREATE TABLE TPCD.ORDERS ( O_ORDERKEY       INTEGER
NOT NULL,
                   O_CUSTKEY        INTEGER NOT NULL,
                   O_ORDERSTATUS    CHAR(1) NOT NULL,
                   O_TOTALPRICE     FLOAT NOT NULL,
                   O_ORDERDATE      DATE NOT NULL,
                   O_ORDERPRIORITY  CHAR(15) NOT NULL,
                   O_CLERK          CHAR(15) NOT NULL,
                   O_SHIPPRIORITY   INTEGER NOT NULL,
                   O_COMMENT        VARCHAR(79) NOT NULL)
ORGANIZE BY (O_ORDERDATE)
    IN OTH_TBSP
    INDEX IN OTHIDX_TBSP;

CREATE TABLE TPCD.LINEITEM ( L_ORDERKEY    INTEGER NOT
NULL,
                    L_PARTKEY     INTEGER NOT NULL,
                    L_SUPPKEY     INTEGER NOT NULL,
                    L_LINENUMBER  INTEGER NOT NULL,
                    L_QUANTITY    FLOAT NOT NULL,
                    L_EXTENDEDPRICE  FLOAT NOT NULL,
                    L_DISCOUNT    FLOAT NOT NULL,
                    L_TAX         FLOAT NOT NULL,
                    L_RETURNFLAG  CHAR(1) NOT NULL,
                    L_LINESTATUS  CHAR(1) NOT NULL,
                    L_SHIPDATE    DATE NOT NULL,
                    L_COMMITDATE  DATE NOT NULL,
                    L_RECEIPTDATE DATE NOT NULL,
                    L_SHIPINSTRUCT CHAR(25) NOT NULL,
                    L_SHIPMODE     CHAR(10) NOT NULL,
                    L_COMMENT      VARCHAR(44) NOT NULL)
ORGANIZE BY (L_SHIPDATE)
    IN LINE_TBSP
    INDEX IN LINEIDX_TBSP;

COMMIT WORK;
```

B.10 dss.load.300GB

```
CONNECT TO tpcd;


LOAD FROM
  /vol2/nation.tbl
OF DEL MODIFIED BY COLDEL| FASTPARSE NOHEADER
MESSAGES /home/db2inst1/tpcd/nation.msg
REPLACE INTO tpcd.nation
NONRECOVERABLE;

COMMIT WORK;

LOAD FROM
  /vol2/region.tbl
OF DEL MODIFIED BY COLDEL| FASTPARSE NOHEADER
MESSAGES /home/db2inst1/tpcd/region.msg
REPLACE INTO tpcd.region
NONRECOVERABLE;

COMMIT WORK;

LOAD FROM
  part.tbl.1,
  part.tbl.2
OF DEL MODIFIED BY COLDEL| FASTPARSE
MESSAGES /home/db2inst1/tpcd/part.msg
REPLACE INTO tpcd.part NONRECOVERABLE
PARTITIONED DB CONFIG MODE LOAD_ONLY
part_file_location /vol2;

COMMIT WORK;

LOAD FROM
  supplier.tbl
OF DEL MODIFIED BY COLDEL| FASTPARSE
MESSAGES /home/db2inst1/tpcd/supplier.msg
REPLACE INTO tpcd.supplier NONRECOVERABLE
PARTITIONED DB CONFIG MODE LOAD_ONLY
part_file_location /vol2;

COMMIT WORK;

LOAD FROM
  partsupp.tbl.1,
  partsupp.tbl.2,
  partsupp.tbl.3,
  partsupp.tbl.4,
  partsupp.tbl.5,
  partsupp.tbl.6,
  partsupp.tbl.7,
  partsupp.tbl.8,

  partsupp.tbl.9,
  partsupp.tbl.10
OF DEL MODIFIED BY COLDEL| FASTPARSE
MESSAGES /home/db2inst1/tpcd/partsupp.msg
REPLACE INTO tpcd.partsupp NONRECOVERABLE
PARTITIONED DB CONFIG MODE LOAD_ONLY
part_file_location /vol2;

COMMIT WORK;

LOAD FROM
  customer.tbl.1,
  customer.tbl.2
OF DEL MODIFIED BY COLDEL| FASTPARSE
MESSAGES /home/db2inst1/tpcd/customer.msg
REPLACE INTO tpcd.customer NONRECOVERABLE
PARTITIONED DB CONFIG MODE LOAD_ONLY
part_file_location /vol2;

COMMIT WORK;

LOAD FROM
  orders.tbl.1,
  orders.tbl.2,
  orders.tbl.3,
  orders.tbl.4,
  orders.tbl.5,
  orders.tbl.6,
  orders.tbl.7,
  orders.tbl.8,
  orders.tbl.9,
```

```
  orders.tbl.10,
  orders.tbl.11,
  orders.tbl.12,
  orders.tbl.13,
  orders.tbl.14,
  orders.tbl.15
OF DEL MODIFIED BY COLDEL| FASTPARSE
MESSAGES /home/db2inst1/tpcd/orders.msg
INSERT INTO tpcd.orders NONRECOVERABLE
PARTITIONED DB CONFIG MODE LOAD_ONLY
part_file_location /vol2;

COMMIT WORK;

LOAD FROM
  lineitem.tbl.1,
  lineitem.tbl.2,
  lineitem.tbl.3,
  lineitem.tbl.4,
  lineitem.tbl.5,
  lineitem.tbl.6,
  lineitem.tbl.7,
  lineitem.tbl.8,
  lineitem.tbl.9,
  lineitem.tbl.10,
  lineitem.tbl.11,
  lineitem.tbl.12,
  lineitem.tbl.13,
  lineitem.tbl.14,
  lineitem.tbl.15,
  lineitem.tbl.16,
  lineitem.tbl.17,
  lineitem.tbl.18,
  lineitem.tbl.19,
  lineitem.tbl.20,
  lineitem.tbl.21,
  lineitem.tbl.22,
  lineitem.tbl.23,
  lineitem.tbl.24,
  lineitem.tbl.25,
  lineitem.tbl.26,
  lineitem.tbl.27,
  lineitem.tbl.28,
  lineitem.tbl.29,
  lineitem.tbl.30,
  lineitem.tbl.31,
  lineitem.tbl.32,
  lineitem.tbl.33,
  lineitem.tbl.34,
  lineitem.tbl.35
OF DEL MODIFIED BY COLDEL| FASTPARSE
MESSAGES /home/db2inst1/tpcd/lineitem.msg
INSERT INTO tpcd.lineitem NONRECOVERABLE
PARTITIONED DB CONFIG MODE LOAD_ONLY
part_file_location /vol1;

LOAD FROM
  lineitem.tbl.36,
  lineitem.tbl.37,
  lineitem.tbl.38,
  lineitem.tbl.39,
  lineitem.tbl.40
OF DEL MODIFIED BY COLDEL| FASTPARSE
MESSAGES /home/db2inst1/tpcd/lineitem.msg
INSERT INTO tpcd.lineitem NONRECOVERABLE
PARTITIONED DB CONFIG MODE LOAD_ONLY
part_file_location /vol2;

COMMIT WORK;

CONNECT RESET;
TERMINATE;
```

B.11 create_indexes.ddl

```
-------------------------------------------------------
-- Create Indexes
-------------------------------------------------------
values(current timestamp);
CREATE UNIQUE INDEX TPCD.R_RK ON TPCD.REGION
        (R_REGIONKEY ASC) PCTFREE 0;

commit work;
alter table tpcd.region add primary key (r_regionkey);
commit work;

values(current timestamp);
CREATE UNIQUE INDEX TPCD.N_NK ON TPCD.NATION
        (N_NATIONKEY ASC) PCTFREE 0 ;
commit work;
alter table tpcd.nation add primary key (n_nationkey);
commit work;

values(current timestamp);
CREATE INDEX TPCD.N_RK ON TPCD.NATION
        (N_REGIONKEY ASC) PCTFREE 0 ;
commit work;

values(current timestamp);
CREATE UNIQUE INDEX TPCD.S_SK ON TPCD.SUPPLIER
        (S_SUPPKEY ASC) PCTFREE 0 ;
commit work;
alter table tpcd.supplier add primary key (s_suppkey);
commit work;

values(current timestamp);
CREATE INDEX TPCD.S_NK ON TPCD.SUPPLIER
        (S_NATIONKEY ASC) PCTFREE 0 ;
commit work;

values(current timestamp);
CREATE UNIQUE INDEX TPCD.PS_PKSK ON TPCD.PARTSUPP
        (PS_PARTKEY ASC, PS_SUPPKEY ASC) PCTFREE 0 ;
commit work;
alter table tpcd.partsupp add primary key (ps_partkey, ps_suppkey);
commit work;

values(current timestamp);
CREATE INDEX TPCD.PS_PK ON TPCD.PARTSUPP
        (PS_PARTKEY ASC) PCTFREE 0 ;
commit work;

values(current timestamp);
CREATE UNIQUE INDEX TPCD.PS_SKPK ON TPCD.PARTSUPP
        (PS_SUPPKEY ASC, PS_PARTKEY ASC) PCTFREE 0 ;
commit work;

values(current timestamp);
CREATE INDEX TPCD.PS_SK ON TPCD.PARTSUPP
        (PS_SUPPKEY ASC) PCTFREE 0 ;
commit work;

values(current timestamp);
CREATE UNIQUE INDEX TPCD.P_PK ON TPCD.PART
        (P_PARTKEY ASC) PCTFREE 0 ;
commit work;
alter table tpcd.part add primary key (p_partkey);
commit work;
```

```
values(current timestamp);
CREATE UNIQUE INDEX TPCD.C_CK ON TPCD.CUSTOMER
            (C_CUSTKEY ASC) PCTFREE 0 ;
commit work;
alter table tpcd.customer add primary key (c_custkey);
commit work;

values(current timestamp);
CREATE INDEX TPCD.C_NK ON TPCD.CUSTOMER
            (C_NATIONKEY ASC) PCTFREE 0 ;
commit work;

values(current timestamp);
CREATE UNIQUE INDEX TPCD.O_OK ON TPCD.ORDERS
            (O_ORDERKEY ASC) PCTFREE 3 ;
commit work;
alter table tpcd.orders add primary key (o_orderkey);
commit work;

values(current timestamp);
CREATE INDEX TPCD.O_CK ON TPCD.ORDERS
            (O_CUSTKEY ASC) PCTFREE 3 ;
commit work;

values(current timestamp);
CREATE UNIQUE INDEX TPCD.L_OKLN ON TPCD.LINEITEM
            (L_ORDERKEY ASC, L_LINENUMBER ASC) PCTFREE 3 ;
commit work;
alter table tpcd.lineitem add primary key (l_orderkey, l_linenumber);
commit work;

values(current timestamp);

select substr(tbname,1,10),substr(name,1,18),create_time from
sysibm.sysindexes where tbcreator='TPCD' order by 3;
select substr(tbname,1,10),
substr(name,1,18),indextype,substr(colnames,1,40) from
sysibm.sysindexes where name like 'SQL%' and tbcreator ='TPCD'
order by 1,2;
```

B.12 runstats.ddl

```
RUNSTATS ON TABLE TPCD.NATION    WITH DISTRIBUTION on
all columns

   and columns (
     n_name like statistics,
     n_comment like statistics )
   AND  INDEXES ALL;
commit;

RUNSTATS ON TABLE TPCD.REGION    WITH DISTRIBUTION on
all columns
   and columns (
     r_name like statistics,
     r_comment like statistics )
   AND  INDEXES ALL;
commit;

RUNSTATS ON TABLE TPCD.SUPPLIER  WITH DISTRIBUTION on
all columns
   and columns (
     s_name like statistics,
     s_address like statistics,
     s_phone like statistics,
     s_comment like statistics)
   AND  INDEXES ALL;
commit;

RUNSTATS ON TABLE TPCD.PART      WITH DISTRIBUTION on all
columns
   and columns (
     p_name like statistics,
     p_mfgr like statistics,
     p_brand like statistics,
     p_type like statistics,
     p_container like statistics,
     p_comment like statistics)
   AND  INDEXES ALL;
commit;

RUNSTATS ON TABLE TPCD.PARTSUPP  WITH DISTRIBUTION on
all columns
   and columns (
     ps_comment like statistics)
   AND  INDEXES ALL;
commit;

RUNSTATS ON TABLE TPCD.CUSTOMER  WITH DISTRIBUTION
on all columns
   and columns (
     c_name like statistics,
     c_address like statistics,
     c_phone like statistics,
     c_mktsegment like statistics,
     c_comment like statistics)
   AND  INDEXES ALL;
commit;

RUNSTATS ON TABLE TPCD.ORDERS    WITH DISTRIBUTION on
all columns
   and columns (
     o_orderstatus like statistics,
     o_orderpriority like statistics,
     o_clerk like statistics,
     o_comment like statistics)
   AND  INDEXES ALL;
commit;

RUNSTATS ON TABLE TPCD.LINEITEM  WITH DISTRIBUTION on
all columns
   and columns (
     l_returnflag like statistics,
     l_linestatus like statistics,
     l_shipinstruct like statistics,
     l_shipmode like statistics,
     l_comment like statistics)
   AND  INDEXES ALL;
COMMIT WORK;
```

B.13 run_dbm_cfg.sql

```
update dbm cfg using
MAXAGENTS 200
NUM_POOLAGENTS 64
NUM_INITAGENTS 4
INDEXREC ACCESS
SVCENAME db2cdb2inst1
CPUSPEED 2e-06
COMM_BANDWIDTH 1.000000e+01
DIAGLEVEL 3
NOTIFYLEVEL 3
SHEAPTHRES 140000
MAX_QUERYDEGREE ANY
INTRA_PARALLEL NO
FCM_NUM_BUFFERS 8192
FCM_NUM_RQB 4096
```

```
HEALTH_MON OFF
DFT_MON_TIMESTAMP OFF;
```

B.14 run_db_cfg.sql

```
update db cfg for tpcd using
CATALOGCACHE_SZ 386
APP_CTL_HEAP_SZ 2058
STMTHEAP        20000
APPLHEAPSZ      16000
PCKCACHESZ      640
STAT_HEAP_SZ    10000
DLCHKTIME       5000
MAXLOCKS        5
CHNGPGS_THRESH  60
NUM_IOCLEANERS  6
NUM_IOSERVERS   12
DFT_PREFETCH_SZ 16
MAXAPPLS        40
MAXFILOP        1024
LOGFILSIZ       8192
LOGBUFSZ        256
INDEXREC        ACCESS
DBHEAP          20000
LOCKLIST        20480
DFT_QUERYOPT    7
DFT_DEGREE      1
NUM_FREQVALUES  0
NUM_QUANTILES   300
UTIL_HEAP_SZ    10000
BUFFPAGE        5000
SORTHEAP        5000
LOGPRIMARY      20
LOGSECOND       2
LOGFILSIZ       16384
LOCKTIMEOUT     -1
SOFTMAX         500;
```

B.15 run_set_logfiles.pl

```
#!/usr/bin/perl

system "./set_logfiles_raw_8fastt.sh";
```

B.16 set_logfiles_raw_8fastt.sh

```
db2_all "<<+0< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw42"
db2_all "<<+1< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw46"
db2_all "<<+2< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw50"
db2_all "<<+3< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw54"
db2_all "<<+4< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw58"
db2_all "<<+5< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw62"
db2_all "<<+6< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw66"
db2_all "<<+7< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw70"
db2_all "<<+8< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw42"
db2_all "<<+9< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw46"
db2_all "<<+10< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw50"
db2_all "<<+11< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw54"
db2_all "<<+12< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw58"
db2_all "<<+13< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw62"
db2_all "<<+14< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw66"
db2_all "<<+15< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw70"
```

B.17 run_set_logfiles_qual.pl

```
#!/usr/bin/perl

system "./set_logfiles_raw_qual.sh";
```

B.18 set_logfiles_raw_qual.sh

```
echo " "
echo "Setting db2logs to dedicated raw...."
echo " "
db2_all "<<+0< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw42"
db2_all "<<+1< db2 update db cfg for tpcd using newlogpath
/dev/raw/raw42"
```

B.19 dss.ddl300GB.tbsp.2mln

```
DROP TABLESPACE TPCDTEMP;


create regular tablespace small_tables
in nodegroup catalog_node
managed by system
using ('/vol3/small_tables_mln')
on node(0)
prefetchsize 256
overhead 12.67
transferrate 0.18;

CREATE TEMPORARY TABLESPACE TPCDTEMP
PAGESIZE 32K
MANAGED BY DATABASE
  USING( DEVICE '/dev/sdd1' 1048576) on NODE(0)
  USING( DEVICE '/dev/sdd1' 1048576) on NODE(1)
    bufferpool    BP32KTEMP
    EXTENTSIZE 16
    PREFETCHSIZE 256
    overhead 12.67
    transferrate 0.18;
COMMIT WORK;

CREATE TABLESPACE line_tbsp
PAGESIZE 32K
MANAGED BY DATABASE
  USING( DEVICE '/dev/sdd2' 570000) on NODE(0)
  USING( DEVICE '/dev/sdd2' 570000) on NODE(1)
    bufferpool    BP32K
    EXTENTSIZE 16
    PREFETCHSIZE 256
    overhead 12.67
    transferrate 0.18;

COMMIT WORK;

CREATE TABLESPACE lineidx_tbsp
```

```
PAGESIZE 32K
MANAGED BY DATABASE
  USING( DEVICE '/dev/sdd3' 70000) on NODE(0)
  USING( DEVICE '/dev/sdd3' 70000) on NODE(1)
    bufferpool    BP32K
    EXTENTSIZE 16
    PREFETCHSIZE 256
    overhead 12.67
    transferrate 0.18;

COMMIT WORK;

CREATE TABLESPACE oth_tbsp
PAGESIZE 32K
MANAGED BY DATABASE
  USING( DEVICE '/dev/sdd5' 260000) on NODE(0)
  USING( DEVICE '/dev/sdd5' 260000) on NODE(1)
    bufferpool    BP32K
    EXTENTSIZE 16
    PREFETCHSIZE 256
    overhead 12.67
    transferrate 0.18;

COMMIT WORK;

CREATE TABLESPACE othidx_tbsp
PAGESIZE 32K
MANAGED BY DATABASE
  USING( DEVICE '/dev/sdd6' 55000) on NODE(0)
  USING( DEVICE '/dev/sdd6' 55000) on NODE(1)
    bufferpool    BP32K
    EXTENTSIZE 16
    PREFETCHSIZE 256
    overhead 12.67
    transferrate 0.18;

COMMIT WORK;


CREATE TEMPORARY TABLESPACE TPCDTEMP2
PAGESIZE 4K
MANAGED BY DATABASE
  USING( DEVICE '/dev/sdd7' 4020250) on NODE(0)
  USING( DEVICE '/dev/sdd7' 4020250) on NODE(1)
    bufferpool    IBMDEFAULTBP
    EXTENTSIZE 128
    PREFETCHSIZE 256
    overhead 12.67
    transferrate 0.18;

COMMIT WORK;

DROP TABLESPACE TEMPSPACE1;

COMMIT WORK;
```

B.20 tpcd.setup

```
TPCD_VERSION=2                  # 1 or 2 (Version of tpcd). Default 1
TPCD_PLATFORM=linux                     # aix, nt, sun....
TPCD_DBNAME=TPCD                        # name to create database
under
TPCD_WORKLOAD=H                         # TPC version (R for TPC-R, H
for TPC-H)
TPCD_AUDIT_DIR=/usr/tpcd                # top level directory of tar file for
                                # all the tpcd scripts
TPCD_PRODUCT=v5                         # v5 or pe
                                # Use pe if you really are using pe v1.2!
                                # but I won't guarantee that it will work!
TPCD_MODE=mpp                           # uni/smp/mln/mpp
TPCD_PHYS_NODE=2                        # number of physical nodes
TPCD_LN_PER_PN=8                        # number of logical nodes per
physical node
TPCD_SF=300                             # size of the database (1=1GB,...) to
                                # get test size databases use:
                                #     0.012 = 12MB
                                #     0.1   = 100MB
TPCD_BUILD_STAGE=ALL                    # where to start the build -
currently the
                                # following is possible:
                                #  ALL - do everything (create,load,
                                #        index,stats,config) (Default)
                                #  CRTTBSP - start after create db and
                                #      config setting.  Start righ at
                                #      create tbsp
                                #  LOAD - start from the load of the tables
                                #  INDEX - start from the index creation
                                #       (NOTE if earlyindex is specified,
                                #        then this will do the create index
                                #        followed by the load...)
                                #  RUNSTATS - start from the runstats
                                #       (NOTE Do not use this option if
                                #        distribution stats are gathered
                                #        as part of the load, this will
                                #        start after the load and indices
                                #        have been created.
                                #  CONFIG - start from the setting up of
                                #        the benchmark runs config setup
                                #
TPCD_DBPATH=/usr/tpcd                   # path for database (defaults to
home)
TPCD_DDLPATH=/usr/tpcd/custom           # path for all ddl files and
customized
                                # scripts (load script), config files,etc
TPCD_BUFFERPOOL_DEF=bufferpools.ddl     # name of filp with
bufferpool definitions
                                # and  sizes
TPCD_NODEGROUP_DEF=create_nodegroups    # name of file in
ddlpath with nodegroup
                                # definitions
TPCD_EXPLAIN_DDL=NULL                   #  file with DDL for explains
statments
                                # if this is NULL then uses the default
                                # and puts it in USERSPACE1 across all
                                # nodes...nt 1TB found it was faster if
                                # just in a single node nodegroup
TPCD_TBSP_DDL=dss.ddl300GB.tbsp         # ddl file for tablespaces
TPCD_DDL=create_tables.ddl              # ddl file for tables
TPCD_QUAL_TBSP_DDL=dss.ddl100GB.tbsp.2mln   # ddl file for
tablespaces for qual
TPCD_QUAL_DDL=create_tables.ddl         # ddl file for qualification
database
                                # tblspaces and tables should be identical
                                # to regular ddl except container names
TPCD_INDEXDDL=create_indexes.ddl        # ddl file for indexes
TPCD_EXTRAINDEX=no                      # no = no extra indexes
                                # filename = If you want to create some
                                # indices before
                                # the load, and some indices after, then
                                # use this additional file to specify the
                                # indices to create after the load.
TPCD_ADD_RI=NULL                        # file name that contains any RI
                                # constraints to add after index creation
                                # set to NULL (default) if unused
TPCD_AST=NULL                           # ??
TPCD_RUNSTATS=runstats.ddl              # ddl file for runstats.  If you
have
```

```
                        # created indices before the load (ie
                        # TPCD_EARLYINDEX=yes), have specified
to
                        # gather stats on the load command (either
                        # through your own load script or by using
                        # TPCD_LOADSTATS=yes, AND you have
                        # specified a file for TPCD_EXTRAINDEX
                        # then this runstats file should include
                        # the runstats commands specifically for
                        # the extra indices.
TPCD_RUNSTATSHORT=NULL              # ddl file for short runstats
that are
                        # run in the background while the
                        # TPCD_RUNSTATS are run in the
foreground
                        # of the build.  If this is used, then
                        # TPCD_RUNSTATS should have the
runstats
                        # command for lineitem and
                        # TPCD_RUNSTATSHORT should have
runstats
                        # commands for all other tables.
                        #
TPCD_DBGEN=/usr/tpcd/appendix.v2/dbgen # path name to data
generation code
                        # Parameters used to specify source of
                        # data for load scripts
TPCD_INPUT=NULL                 # NULL - use dbgen generated
data  OR
                        # path name -  to the pre-generated
                        #           flat files
                        # /gwl/dss/12MB - path for pregen'd 12MB
                        # /gwl/dss/100MB - path for pregen'd 100MB
                        #
TPCD_QUAL_INPUT=NULL                # NULL - use dbgen
generated data  OR

                        # path name -  to the pre-generated
                        #           flat files

TPCD_TAILOR_DIR=/usr/tpcd/tailor      # path name for the directory
used to
                        # generate split specific config files
                        # only used for partitioned environment

TPCD_EARLYINDEX=no                  # create indexes before the
load

                        # LOAD specific parameters follow:
TPCD_LOAD_DB2SET_SCRIPT=run_db2set.pl  # Script that contains
the db2set commands
                        # for the load process Use NULL if not
                        # specified
TPCD_LOAD_CONFIGFILE=load_db_cfg.sql   # config file with
specific database config
                        # parms for the load/index/runstats part
                        # of the build.
                        # set to NULL if use defaults
TPCD_LOAD_DBM_CONFIGFILE=load_dbm_cfg.sql   # config file
with specific
                        # database manager config parts for the
                        # load/index/runstats part of the build.
                        # set to NULL if use defaults
TPCD_LOAD_QUALCONFIGFILE=load_db_qual_cfg.sql   # config file
with specific database config
                        # parms for the load/index/runstats part
                        # of the build for qualification db.
                        # set to NULL if use defaults
TPCD_LOAD_DBM_QUALCONFIGFILE=load_dbm_cfg.sql   # config
file with specific
                        # database manager config parts for the
                        # load/index/runstats part of the build.
                        # set to NULL if use defaults

TPCD_LOADSTATS=no                   # gather statistics during load
                        # ignored if EARLYINDEX is not set
                        # due to runstats limitation
#TPCD_TEMP=/usr/db2inst1/sqllib/tmp     # path for LOAD temp files
TPCD_TEMP=/home/db2inst1/sqllib/tmp     # path for LOAD temp files
                        # defaults to /u/<instance>/sqllib/tmp
                        # used in load script only
TPCD_SORTBUF=4096                   # sortbuf size for LOAD
                        # used in load script only
TPCD_LOAD_PARALLELISM=0             # degree of parallelism to
use on load
                        # 0 = use the "intelligent default" that
                        # the load will chose at run time
                        # used in load script only
TPCD_COPY_DIR=/dev/null             # directory where copy image is
created
                        # on load command CURRENTLY UNUSED
                        # used in load script only
TPCD_FASTPARSE=yes                  # use fastparse on load
                        # used in load script only
                        # Backup and logfile specific parameters
follow:
TPCD_BACKUP_DIR=/backup_node          # directory where backup
files are placed
TPCD_LOGPRIMARY=6                   # NULL/value = how many
primary log files
                        # to configure.  If NULL is specified then
                        # the default is not changed.
TPCD_LOGFILSIZ=1024                 # NULL/value = how 4KB pages
to use for
                        # logfilsiz db cfg parameter.  If NULL is
                        # specified then the default is not changed
TPCD_LOGSECOND=5                    # NULL/value = how many
secondary log files
                        # to configure.  If NULL is specified then
                        # the default is not changed.
TPCD_LOG_DIR=/usr/tpcd/vol4/logs       # directory where log files
stored..
                        # NULL leaves them in the dbpath
TPCD_LOG_DIR_SETUP_SCRIPT=run_set_logfiles.pl # executable
script to setup log dir
TPCD_LOG_QUAL_DIR=NULL              # directory where qual log
files stored
                        # NULL leaves them in the dbpath
TPCD_LOG_QUAL_DIR_SETUP_SCRIPT=run_set_logfiles_qual.pl
# executable script to setup QUAL log dir
TPCD_LOG=yes                        # yes/no - whether to turn
LOG_RETAIN on
                        # i.e. are backups needed to be taken



                        # CONFIG specific parameters
TPCD_DB2SET_SCRIPT=run_db2set.pl       # Script that contains the
db2set commands
                        # for the benchmark run.  Use NULL if not
                        # specified
TPCD_CONFIGFILE=run_db_cfg.sql        # name of configuration file
in ddl path
                        # that will be used for the benchmark run
                        # for database configuration parms
TPCD_DBM_CONFIG=run_dbm_cfg.sql       # name of config file for
database manager
                        # config parms
                        #   dbm cfg set up separately from db cfg
TPCD_QUALCONFIGFILE=run_db_qual_cfg.sql    # name of
database cfg file in ddl path
```

```
                        # for qualification database
TPCD_DBM_QUALCONFIG=run_dbm_cfg.sql    # name of config file
for database manager
                        # config parms for qualification database

TPCD_MACHINE=NULL                      # set to NULL if using load
config file
                        # big/medium/small  size of machine used to
                        # determine buffpage, sortheap,sheapthres
                        # and ioservers parms for load, create
                        # index and runstats
                        # NOTE that this parameter is ignored if
                        # a TPCD_LOAD_CONFIGFILE

TPCD_SMPDEGREE=1                       # 1...# of degrees of parallelism
to run
                        # with
TPCD_AGENTPRI=NULL                     # set agentpri to this value
(default
                        # is SYSTEM)

TPCD_ACTIVATE=no                       # activate the database upon
build
                        # completion



                        # run specific parameters
TPCD_AUDIT=no                          # no/yes
                        # no - don't set up qualification db stuff
                        # yes - set up qualification db queries
                        #     - build the update function tables
                        #       and data before we get into the
                        #       timing of the creation of the
                        #       tables and the load.
TPCD_TMP_DIR=/tmp/tpcd                 # place to put temp working files
TPCD_SHARED_TEMP_FULL_PATHNAME=/home/db2inst1/sqllib/tm
p    # where default on AIX is /u/INSTANCE/sqllib/tmp
                        # there is no default on NT systems so
                        # this parameter MUST be set.
                        # NOTE for NT EEE systems this directory
                        #   must have been set up as a shared name.
                        #   it will be referenced by
\\MACHINENAME\sharedtmpdir
                        #   by all nodes (including the catalog node or
origin
                        #   of the shared directory.
                        #   In order to work around the \\ specification
problem
                        #   the pathname must be specified
WITHOUT the \\

TPCD_QUERY_TEMPLATE_DIR=standard.V2    # subdirectory in
AUDIT_DIR/queries
                        # to use as the source of the query
                        # templates.  Currently there are
                        # v2 ones and pe ones.  You can make
                        # your own directory following the same
                        # form as in the v2 directory using
                        # any variant you wish
TPCD_QUAL_DBNAME=TPCD                  # name of qualification
database
TPCD_NUMSTREAM=6                       # number of streams for the
throughput test
TPCD_FLATFILES=/vol1/ufdata_300GB      # where to generate flat
files
                        # for update functions
TPCD_STAGING_TABLE_DDL=create_UFtables.ddl    # file that
contains the ddl for creating
                        # the staging tables if they are used for
                        # the update functions
TPCD_PRELOAD_STAGING_TABLE_SCRIPT=loadSampleUf.sh   #
file that contains the sql for preloading
                        # and gathering stats on sample UF data
                        # Note that the data used is sample data
                        # and is not data from any of the applied
                        # update pairs
TPCD_DELETE_STAGING_TABLE_SQL=NULL     # file that contains
the sql for deleting
                        # the preloaded data from the staging
                        # tables
TPCD_UPDATE_IMPORT=false               # true = use import for the
staging tables
                        #   for UNI/SMP mode only (code change in
                        #   tpcdbatch) (if not uni mode then must
                        #   change load_update)
                        # false = use load for staging tables
                        # The default is false if not set.
                        # NOTE that this parm is only for UNI/SMP
                        # it is not for multi node invocation

TPCD_SPLIT_UPDATES=64                  # number of chunks to split
the update
                        # function into.
TPCD_CONCURRENT_INSERTS=16             # number of insert
chunks that are run
                        # concurrently.  This number should be
                        # evenly divisible by
TPCD_SPLIT_UPDATES
TPCD_CONCURRENT_INSERTS_LOAD=1         # number of insert
chunks that are loaded
                        # concurrently.  This number should be
                        # evenly divisible by
TPCD_SPLIT_UPDATES
                        # this controls the load portion of the
                        # insert routine for partitioned databases
TPCD_SPLIT_DELETES=64                  # number of portions to split
the delete
                        # function into.
                        # this variable is only valid in UNI/SMP
                        # mode.
TPCD_CONCURRENT_DELETES=16             # number of delete
chunks that are run
                        # concurrently.  This number should be
                        # evenly divisible by
TPCD_SPLIT_DELETES
TPCD_GEN_UPDATEPAIRS=28                # number of pairs of
update function data
                        # to generate
                        # if 0 the update data generation and
                        # setup will not be done.  use this if
                        # you don't want to run the update
                        # functions   (Update functions not
                        # fully tested in new env't yet)
TPCD_GENERATE_SEED_FILE=yes            # yes/no  These are the
seed files for
                        # generating the query substitution values
                        # yes - generate a seed file base on
                        #       year/month/day (for audited runs)
                        # no  - use qgen's default seeds
TPCD_RUN_ON_MULTIPLE_NODES=NULL        # pe V1.2 only - will
we be running each
                        # query stream of throughput starting at
                        # different nodes or from same node
TPCD_STATS_INTERVAL=5                  # timing interval for
vmstats/iostats
TPCD_STATS_THRU_INT=60                 # timing interval for
vmstats/iostats for
                        # throughput test
TPCD_GATHER_STATS=off                  # on/off   - only implement for
AIX yet
```

```
                        # on = gather statistics around power
                        #      test run  (vmstat,iostat,netstat)
                        # off = no stats gathered during power run

TPCD_UFTEMP=UFTEMP              # base name of tablespace(s)
where the
                        # staging tables for the update functions
                        # are created
                        # this name will be used as the
                        # basename for the tablespaces...eg
                        # UFTEMP1 UFTEMP2 ....
TPCD_HAVECOMPILER=yes           # rebuild tpcdbatch
executable
                        # yes/no
TPCD_SLEEP=5                # ?
TPCD_INLISTMAX=default          # max num of keys to delete at
a time
                        # for UF2, use "default" for default.
TPCD_LOAD_SCRIPT=dss.load.300GB     # script to run for loading
tables
                        # in TPCD_DDLPATH directory under
mln/mpp
                        # leave as NULL if using default genloaduni
TPCD_LOAD_SCRIPT_QUAL=dss.load.1GB  # script to run for
loading tables in
                        # TPCD_DDLPATH directory under mln/mpp
                        # for QUAL db
TPCD_ROOTPRIV=no                # do you have root privileges to
be able
                        # get values of things like schedtune
                        # and vmtune (currently on AIX only)
TPCD_APPEND_ON=no               # set to no if the cluster
indexes are used
TPCD_DB2LOG=/home/db2inst1/sqllib/db2dump # db2diag.log
directory
```

B.21 buildtpcd

```
#!/usr/bin/perl

# usage   buildtpcd [QUAL]

# ASSUMPTIONS: all ddl files have commits in them!
($myName = $0) =~ s@.*/@@; $usage="
Usage: buildtpcd [QUAL]
   where QUAL is the optional parameter saying to build the qualifica-
tion
            database (sf = .1 = 100MB)\n";

$qual="";
if (@ARGV == 1)
{
   $qual = $ARGV[0];
}

# get TPC-D specific environment variables
require 'getvars';

# Use the macros in here so that they can handle the platform differ-
ences.
# macro.pl should be sourced from cmvc, other people wrote and
maintain it.
require "macro.pl";
require "tpcdmacro.pl";

# Make output unbuffered.
select(STDOUT);
$| = 1 ;

# verify that necessary environment variables for building the database
# are present.  Default those that aren't necessary
require "version";

if (length($ENV{"TPCD_PLATFORM"}) <= 0)
{
  die "TPCD_PLATFORM environment variable not set\n";
}
$platform=$ENV{"TPCD_PLATFORM"};
if ( $platform eq "nt" )
{
   $sep="&";
}
else
{
   $sep=";";
}
if ((length($ENV{"TPCD_BUILD_STAGE"}) <= 0) ||
($ENV{"TPCD_BUILD_STAGE"}) eq "NULL" )
{
   $ENV{"TPCD_BUILD_STAGE"} = "ALL";
}
if (length($ENV{"TPCD_PRODUCT"}) <= 0)
{
   die "Must set TPCD_PRODUCT env't var.\n";
}
if ( length($ENV{"TPCD_DBNAME"}) <= 0 )
{
  die "TPCD_DBNAME environment variable not set\n";
}
if (length($ENV{"TPCD_MODE"}) <= 0)
{
  die "TPCD_MODE environment variable not set - uni/smp/mln \n";
}
if (length($ENV{"TPCD_SF"}) <= 0)
{
  die "TPCD_SF environment variable not set\n";
}
if (length($ENV{"TPCD_DBPATH"}) <= 1)
{
   # if no db pathname specified, build the db in the home directory
   if ( $platform eq "aix" || $platform eq "sun" || $platform eq "ptx" ||
$platform eq "hp" || $platform eq "linux")
   {
      $ENV{"TPCD_DBPATH"} = $ENV{"HOME"};
   }
   elsif ( $platform eq "nt" )
   {
      $ENV{"TPCD_DBPATH"} = $ENV{"HOMEDRIVE"};
   }
   else
   {
      die "platform $platform not supported yet\n";
   }
}
if (length($ENV{"TPCD_DDLPATH"}) <= 0)
{
   # if no db pathname specified, use default
  $ENV{"TPCD_DDLPATH"} =
"/afs/tor/groups/dbp/perf/benchmark/tpcd/ddl/vanilla";
}
if (length($ENV{"TPCD_GENERATE_SEED_FILE"}) <= 0)
{
   # if no db pathname specified, use default
  $ENV{"TPCD_GENERATE_SEED_FILE"} = "no";
}
if (length($ENV{"TPCD_DDL"}) <= 0)
{
   $ENV{"TPCD_DDL"} = "dss.ddl";
```

```
}
if (length($ENV{"TPCD_STAGING_TABLE_DDL"}) <= 0)
{
   $ENV{"TPCD_STAGING_TABLE_DDL"} = "NULL";
}
if (length($ENV{"TPCD_PRELOAD_STAGING_TABLE_SCRIPT"}) <=
0)
{
   $ENV{"TPCD_PRELOAD_STAGING_TABLE_DDL"} = "NULL";
}
if (length($ENV{"TPCD_DELETE_STAGING_TABLE_SQL"}) <= 0)
{
   $ENV{"TPCD_DELETE_STAGING_TABLE_DDL"} = "NULL";
}
if (length($ENV{"TPCD_TBSP_DDL"}) <= 0)
{
   $ENV{"TPCD_TBSP_DDL"} = "dss.tbsp.ddl";
}
if (length($ENV{"TPCD_INDEXDDL"}) <= 0)
{
   $ENV{"TPCD_INDEXDDL"} = "dss.index";
}
if (length($ENV{"TPCD_RUNSTATS"}) <= 0)
{
   $ENV{"TPCD_RUNSTATS"} = "dss.runstats";
}
if (length($ENV{"TPCD_RUNSTATSHORT"}) <= 0)
{
   $ENV{"TPCD_RUNSTATSHORT"} = "NULL";
}
if (length($ENV{"TPCD_ADD_RI"}) <= 0)
{
   $ENV{"TPCD_ADD_RI"} = "NULL";
}
if (length($ENV{"TPCD_AST"}) <= 0)
{
   $ENV{"TPCD_AST"} = "NULL";
}
if ( ($ENV{"TPCD_INPUT"}) eq "NULL" )
{
   if (length($ENV{"TPCD_DBGEN"}) <= 0)
   {
      die "Must set TPCD_DBGEN if pregenerated flatfiles are not pro-
vided (TPCD_INPUT=NULL)\n";
   }
}
if ( $qual eq "QUAL" )
{
   if ( ($ENV{"TPCD_QUAL_INPUT"}) eq "NULL" )
   {
      if (length($ENV{"TPCD_DBGEN"}) <= 0)
      {
         die "Must set TPCD_DBGEN if pregenerated flatfiles are not
provided (TPCD_QUAL_INPUT=NULL)\n";
      }
}
}
if (length($ENV{"TPCD_TEMP"}) <= 1)
{
   $ENV{"TPCD_TEMP"} = "/u/$instance/sqllib/tmp";
}

if (length($ENV{"TPCD_SORTBUF"}) <= 0)
{
   $ENV{"TPCD_SORTBUF"} = 4096;
}
if (length($ENV{"TPCD_LOAD_PARALLELISM"}) <= 0)
{
   $ENV{"TPCD_LOAD_PARALLELISM"} = 0;
}
if (length($ENV{"TPCD_LOADSTATS"}) <= 0)
{
   $ENV{"TPCD_LOADSTATS"} = "no";

}
if (length($ENV{"TPCD_COPY_DIR"}) <= 0)
{
   $ENV{"TPCD_COPY_DIR"} = "${delim}dev${delim}null";
}
if (length($ENV{"TPCD_FASTPARSE"}) <= 0)
{
   $ENV{"TPCD_FASTPARSE"} = "no";
}
if (length($ENV{"TPCD_BACKUP_DIR"}) <= 0)
{
   $ENV{"TPCD_BACKUP_DIR"} = "${delim}dev${delim}null";
}
if (length($ENV{"TPCD_LOG"}) <= 0)
{
   $ENV{"TPCD_LOG"} = "no";
}
if (length($ENV{"TPCD_LOGPRIMARY"}) <= 0)
{
   $ENV{"TPCD_LOGPRIMARY"} = "NULL";
}
if (length($ENV{"TPCD_LOGSECOND"}) <= 0)
{
   $ENV{"TPCD_LOGSECOND"} = "NULL";
}
if (length($ENV{"TPCD_LOGFILSIZ"}) <= 0)
{
   $ENV{"TPCD_LOGFILSIZ"} = "NULL";
}
if (length($ENV{"TPCD_LOG_DIR"}) <= 0)
{
   $ENV{"TPCD_LOG_DIR"} = "NULL";
}
if (length($ENV{"TPCD_LOG_DIR_SETUP_SCRIPT"}) <= 0)
{
   $ENV{"TPCD_LOG_DIR_SETUP_SCRIPT"} = "NULL";
}
if (length($ENV{"TPCD_CONFIGFILE"}) <= 0)
{
   $ENV{"TPCD_CONFIGFILE"} = "dss.dbconfig";
}
if (length($ENV{"TPCD_DBM_CONFIG"}) <= 0)
{
   $ENV{"TPCD_DBM_CONFIG"} = "NULL";
}
if (length($ENV{"TPCD_MACHINE"}) <= 0)
{
   $ENV{"TPCD_MACHINE"} = "medium";
}
if (length($ENV{"TPCD_SMPDEGREE"}) <= 0 ||
$ENV{"TPCD_SMPDEGREE"} eq "NULL")
{
   $ENV{"TPCD_SMPDEGREE"} = 1;
}
if (length($ENV{"TPCD_AGENTPRI"}) <= 0)
{
   $ENV{"TPCD_AGENTPRI"} = NULL;
}
if (length($ENV{"TPCD_ACTIVATE"}) <= 0)
{
   $ENV{"TPCD_ACTIVATE"} = "no";
}
if (length($ENV{"TPCD_AUDIT"}) <= 0)
{
   die "Must set TPCD_AUDIT env't var.  Real audit timing sequence
run if yes\n";
```

```
}
if (length($ENV{"TPCD_AUDIT_DIR"}) <= 0)
{
   die "TPCD_AUDIT_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_LOAD_DB2SET_SCRIPT"}) <= 0)
{
   $ENV{"TPCD_LOAD_DB2SET_SCRIPT"}="NULL"
}
if (length($ENV{"TPCD_DB2SET_SCRIPT"}) <= 0)
{
   $ENV{"TPCD_DB2SET_SCRIPT"}="NULL"
}
if (length($ENV{"TPCD_BUFFERPOOL_DEF"}) <= 0)
{
   $ENV{"TPCD_BUFFERPOOL_DEF"}="NULL"
}
if (length($ENV{"TPCD_EXPLAIN_DDL"}) <= 0)
{
   $ENV{"TPCD_EXPLAIN_DDL"}="NULL"
}
if (length($ENV{"TPCD_NODEGROUP_DEF"}) <= 0)
{
   $ENV{"TPCD_NODEGROUP_DEF"}="NULL"
}
if (length($ENV{"TPCD_PHYS_NODE"}) <= 0)
{
   $ENV{"TPCD_NODEGROUP_DEF"}="NULL"
}
if (length($ENV{"TPCD_APPEND_ON"}) <= 0)
{
   $ENV{"TPCD_APPEND_ON"}="yes"
}
#set up local variables
$tpcdVersion=$ENV{"TPCD_VERSION"};
$buildStage=$ENV{"TPCD_BUILD_STAGE"};
$product=$ENV{"TPCD_PRODUCT"};
$dbname=$ENV{"TPCD_DBNAME"};
$mode=$ENV{"TPCD_MODE"};
$sf=$ENV{"TPCD_SF"};
$sfReal=$sf;          # need a "saved" one for qualification stuff
$dbpath=$ENV{"TPCD_DBPATH"};
$ddlpath=$ENV{"TPCD_DDLPATH"};
$ddl=$ENV{"TPCD_DDL"};
$stagingTbl=$ENV{"TPCD_STAGING_TABLE_DDL"};
$preloadSam-
pleUF=$ENV{"TPCD_PRELOAD_STAGING_TABLE_SCRIPT"};
$deleteSam-
pleUF=$ENV{"TPCD_DELETE_STAGING_TABLE_SQL"};
$buffpooldef=$ENV{"TPCD_BUFFERPOOL_DEF"};
$nodegroupdef=$ENV{"TPCD_NODEGROUP_DEF"};
$tbspddl=$ENV{"TPCD_TBSP_DDL"};
$indexddl=$ENV{"TPCD_INDEXDDL"};
$extraindex=$ENV{"TPCD_EXTRAINDEX"};
$runstats=$ENV{"TPCD_RUNSTATS"};
$runstatShort=$ENV{"TPCD_RUNSTATSHORT"};
$dbgen=$ENV{"TPCD_DBGEN"};
$input=$ENV{"TPCD_INPUT"};
$earlyindex=$ENV{"TPCD_EARLYINDEX"};
$ldtemp=$ENV{"TPCD_TEMP"};
$sortbuf=$ENV{"TPCD_SORTBUF"};
$load_parallelism=$ENV{"TPCD_LOAD_PARALLELISM"};
$loadstats=$ENV{"TPCD_LOADSTATS"};
$copydir=$ENV{"TPCD_COPY_DIR"};
$fparse=$ENV{"TPCD_FASTPARSE"};
$addRI=$ENV{"TPCD_ADD_RI"};
$astFile=$ENV{"TPCD_AST"};
$genSeed=$ENV{"TPCD_GENERATE_SEED_FILE"};
$appendOn=$ENV{"TPCD_APPEND_ON"};

if ( $fparse eq "yes" )
{
   $fastparse="FASTPARSE";
}
else
{
   $fastparse=" ";
}
$backupdir=$ENV{"TPCD_BACKUP_DIR"};
$logprimary=$ENV{"TPCD_LOGPRIMARY"};
$logsecond=$ENV{"TPCD_LOGSECOND"};
$logfilsiz=$ENV{"TPCD_LOGFILSIZ"};
$log=$ENV{"TPCD_LOG"};
$logDir=$ENV{"TPCD_LOG_DIR"};
$logDirScript=$ENV{"TPCD_LOG_DIR_SETUP_SCRIPT"};
$machine=$ENV{"TPCD_MACHINE"};
$configfile=$ENV{"TPCD_CONFIGFILE"};
$dbmconfig=$ENV{"TPCD_DBM_CONFIG"};
$loadconfigfile=$ENV{"TPCD_LOAD_CONFIGFILE"};
$loadDBMconfig=$ENV{"TPCD_LOAD_DBM_CONFIGFILE"};
$smpdegree=$ENV{"TPCD_SMPDEGREE"};
$agentpri=$ENV{"TPCD_AGENTPRI"};
$activate=$ENV{"TPCD_ACTIVATE"};
$RealAudit=$ENV{"TPCD_AUDIT"};
$loadscript=$ENV{"TPCD_LOAD_SCRIPT"};
$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$loadsetScript=$ENV{"TPCD_LOAD_DB2SET_SCRIPT"};
$setScript=$ENV{"TPCD_DB2SET_SCRIPT"};
$explainDDL=$ENV{"TPCD_EXPLAIN_DDL"};
$user=$ENV{"USER"};

# set up override of some parameters to override for qualification da-
tabase
if ( $qual eq "QUAL" )
{
   $loadscript=$ENV{"TPCD_LOAD_SCRIPT_QUAL"};
   if ( length($ENV{"TPCD_QUAL_DBNAME"}) <= 0 )
   {
      die "TPCD_QUAL_DBNAME environment variable not set\n";
   }
   $dbname=$ENV{"TPCD_QUAL_DBNAME"};

   print "DBNAME= $dbname\n\n";

   $sf=$ENV{"TPCD_QUAL_SF"};
   if ( length($ENV{"TPCD_QUAL_DDL"}) <= 0 )
   {
      die "TPCD_QUAL_DDL environment variable not set\n";
   }
   $ddl=$ENV{"TPCD_QUAL_DDL"};
   if ( length($ENV{"TPCD_QUAL_TBSP_DDL"}) <= 0 )
   {
      die "TPCD_QUAL_TBSP_DDL environment variable not set\n";
   }
   $tbspddl=$ENV{"TPCD_QUAL_TBSP_DDL"};
   $input=$ENV{"TPCD_QUAL_INPUT"};
   if ( length($ENV{"TPCD_QUALCONFIGFILE"}) <= 0 )
   {
      die "TPCD_QUALCONFIGFILE environment variable not set\n";
   }
   $configfile=$ENV{"TPCD_QUALCONFIGFILE"};
   if ( length($ENV{"TPCD_DBM_QUALCONFIG"}) <= 0 )
   {
      die "TPCD_DBM_QUALCONFIG environment variable not set\n";
   }
   $dbmconfig=$ENV{"TPCD_DBM_QUALCONFIG"};
   if ( length($ENV{"TPCD_LOAD_QUALCONFIGFILE"}) <= 0 )
   {
      die "TPCD_LOAD_QUALCONFIGFILE environment variable not
set\n";
```

```
    }
    $loadconfigfile=$ENV{"TPCD_LOAD_QUALCONFIGFILE"};
    if ( length($ENV{"TPCD_LOAD_DBM_QUALCONFIGFILE"}) <= 0 )
    {
        die "TPCD_LOAD_DBM_QUALCONFIGFILE environment vari-
able not set\n";
    }
    $loadDBMconfig=$ENV{"TPCD_LOAD_DBM_QUALCONFIGFILE"};
    if (length($ENV{"TPCD_LOG_DIR"}) <= 0)
    {
        $ENV{"TPCD_LOG_DIR"} = "NULL";
    }
    $logDir=$ENV{"TPCD_LOG_QUAL_DIR"};
    if (length($ENV{"TPCD_LOG_QUAL_DIR_SETUP_SCRIPT"}) <= 0)
    {
        die "TPCD_LOG_QUAL_DIR_SETUP_SCRIPT environment
variable not set\n";
    }
    $logDirScript=$ENV{"TPCD_LOG_QUAL_DIR_SETUP_SCRIPT"};
}

if (( $mode eq "uni" ) || ( $mode eq "smp" ))
{
  $all_ln="once";
  $all_pn="once";
  $once="once";
}
else
{
  $all_ln="all_ln";
  $all_pn="all_pn";
  $once="once";
}

###################### set up the config parms for the load, in-
dexes and stats ############################
if ($loadconfigfile eq "NULL")
{
  if ( ($machine eq "NULL") )
  {
   die "Neither a LOAD config file name not a machine size has been
specified!\n";
  }
  $ioclnrs=1;
  $chngpgs=60;

  if ( $machine eq "small" )
  {
     $buffpage = 5000;
     $sortheap = 3000;
     $sheapthres = 8000;
     $util_heap_sz = 5000;
     $ioservers = 6;

  }
  elsif ( $machine eq "medium" )
  {
     $buffpage = 10000;
     $sortheap =  8000;
     $sheapthres = 20000;
     $util_heap_sz = 10000;
     $ioservers = 10;
  }
  elsif ( $machine eq "big" )
  {
     $buffpage = 30000;
     $sortheap = 20000;
     $sheapthres = 50000;
     $util_heap_sz = 30000;
     $ioservers = 20;
  }
  elsif ( $machine eq "sunsmp" )
  {
     $buffpage = 60000;
     $sortheap = 20000;
     $sheapthres = 80000;
     $util_heap_sz = 30000;
     $ioservers = 80;
  }
  elsif ( $machine eq "eastwood" )
  {
     $buffpage = 80000;
     $sortheap = 50000;
     $sheapthres = 81000;
     $ioclnrs = 4;
     $chngpgs = 30;
     $ioservers = 21;
     $util_heap_sz = 50000;
  }
}
###################### echo parameter settings to acknowledge
what is being built ##########################
if ( $buildStage ne "ALL" ){
    print " ***** STARTING the build process at the $buildStage Stage
*****\n";
}
print "Building a TPC-D Version $tpcdVersion $sf GB database on
$dbpath with: \n";
print "   Mode = $mode \n";
print "   Tablespace ddl in $ddlpath${delim}$tbspddl \n";
if ( $nodegroupdef ne "NULL" )
{
   print "   Nodegroup ddl in $ddlpath${delim}$nodegroupdef \n";
}
if ( $buffpooldef ne "NULL" )
{
   print "   Bufferpool ddl in $ddlpath${delim}$buffpooldef \n";
}
print "   Table ddl in $ddlpath${delim}$ddl \n";
print "   Index ddl in              $ddlpath${delim}$indexddl\n";
if ( $extraindex ne "no" )
{
  print " Indices to create after the load $ddlpath${delim}$extraindex\n";
}
if ( $loadscript eq "NULL")
{
    if ( $input eq "NULL" )
    {
        print "   Data generated by DBGEN in $dbgen\n";
    }
    else
    {
        print "   Data loaded from flat files in $input\n";
    }
}
if ( $earlyindex eq "yes" )
{
  print "   Indexes created before loading\n";
}
else
{
  print "   Indexes created after loading\n";
}
if ( $addRI ne "NULL" )
{
    print "   RI being used from $ddlpath${delim}$addRI\n";
}
if ( $astFile ne "NULL" )
{
    print "   AST being used from $ddlpath${delim}$astFile\n";
```

```
}
if ( $loadstats eq "yes" )
{
  if ( $earlyindex eq "yes" )
  {
    print "   Statistics for tables and indexes gathered during load\n";
  }
  else
  {
    if ( $runstatShort eq "NULL" )
    {
      print "   Statistics for tables and indexes gathered after load using
$ddlpath${delim}$runstats \n";
    }
    else
    {
       print "   Statistics for tables and indexes gathered after load using
$ddlpath${delim}$runstats and $ddlpath${delim}$runstatShort\n";
    }
  }
}
else
{
  if ( $runstatShort eq "NULL" )
  {
    print "   Statistics for tables and indexes gathered after load using
$ddlpath${delim}$runstats \n";
  }
  else
  {
    print "   Statistics for tables and indexes gathered after load using
$ddlpath${delim}$runstats and $ddlpath${delim}$runstatShort\n";
  }
}
if ( $loadconfigfile ne "NULL" )
{
  print "   Database Configuration parameters for LOAD taken from
$ddlpath${delim}$loadconfigfile\n";
}
if ( $loadDBMconfig ne "NULL" )
{
  print "   Database manager Configuration parameters for LOAD taken
from $ddlpath${delim}$loadDBMconfig\n";
}
if ( $configfile ne "NULL" )
{
  print "   Database Configuration parameters taken from
$ddlpath${delim}$configfile\n";
}
else
{
  print "   Database Configuration paramters taken from
$ddlpath${delim}dss.dbconfig${sfReal}GB\n";
  $configfile="dss.dbconfig${sfReal}GB";
}
if ( $dbmconfig ne "NULL" )
{
  print "   Database Manager Configuration parameters taken from
$ddlpath${delim}$dbmconfig\n";
}
else
{
  print "   Database Manager Configuration paramters taken from
$ddlpath${delim}dss.dbmconfig${sfReal}GB\n";
  $configfile="dss.dbmconfig${sfReal}GB";
}
#print "   Copy image for load command created in $copydir\n";
if ( $log eq "yes" )
{
  print "   Backup files placed in $backupdir\n";
}
else
{
  print "   No backup will be taken.\n";
}
print "   Log retain set to $log\n";
if ( $logDir eq "NULL" )
{
   print "   Log files remain in database path\n";
}
else
{
   print "   Log file path set to $logDir\n";
}
if ( $logprimary eq "NULL" )
{
   print "   Log Primary left at default\n";
}
else
{
   print "   Log Primary set to $logprimary\n";
}
if ( $logsecond eq "NULL" )
{
   print "   Log Second left at default\n";
}
else
{
   print "   Log second set to $logsecond\n";
}
if ( $logfilsiz eq "NULL" )
{
   print "   Logfilsiz left at default\n";
}
else
{
   print "   Logfilsiz set to $logfilsiz\n";
}

if ($loadconfigfile eq "NULL")
{
  print "   Machine size set to $machine so the following configura-
tion\n";
  print "   parameters are used for load, create index and runstats: \n";
  print "      BUFFPAGE = $buffpage \n";
  print "      SORTHEAP = $sortheap \n";
  print "      SHEAPTHRES = $sheapthres\n";
  print "      NUM_IOSERVERS = $ioservers\n";
  print "      NUM_IOCLEANERS = $ioclnrs\n";
  print "      CHNGPGS_THRESH = $chngpgs\n";
  print "      UTIL_HEAP_SZ = $util_heap_sz\n";
  print "      Degree of parallelism (dft_degree and max_querydegree)
set to $smpdegree\n";
  print "      Parameters for load are: temp file      = $ldtemp\n";
  print "                               sort buf       = $sortbuf\n";
  print "                               ld parallelism = $load_parallelism\n";
  if ( $fparse eq "yes" )
  {
    print "                                FASTPARSE used on load\n";
  }
}
if ( $loadscript ne "NULL")
{
  print "   Load commands  in $ddlpath${delim}$loadscript\n";
}



print "   Degree of parallelism (dft_degree and max_querydegree) set
to $smpdegree\n";
```

```
if ( $agentpri ne "NULL" )
{
   print "   AGENTPRI set to $agentpri\n";
}
if ( $activate eq "yes" )
{
   print "   Database will be activated when build is complete\n";
}
if ( $explainDDL ne "NULL" )
{
   print "   EXPLAIN DDL being used from
$ddlpath${delim}$explainDDL\n";
}
else
{
   print "   EXPLAIN DDL being used from default sqllib directory\n";
}

print "Sleeping for 15 seconds to give you a chance to reconsider...\n";
#sleep 15;
######################################## end echo parame-
ters ##########################################
# don't need separate calls for mln/mpp vs uni since the $all_ln will be
# defined appropriately
if ( $platform eq "nt" )
{
   if (($mode eq "uni") || ($mode eq "smp"))
   {
      #spaces required for NT
      $rc=&dodb_noconn("db2set DB2OPTIONS=\"   -t -v +c\"",$all_ln);
   }
   else
   {
      $rc=&dodb_noconn("db2set DB2OPTIONS=\\\"   -t -v
+c\\\"",$all_ln);
   }
}
else
{
   if (($mode eq "uni") || ($mode eq "smp"))
   {
      $rc=&dodb_noconn("db2set DB2OPTIONS=\"-t -v +c\"",$all_ln);
   }
   else
   {
      $rc=&dodb_noconn("db2set DB2OPTIONS=\\\"-t -v +c\\\"",$all_ln);
   }
}

if ( $rc != 0 )
{
  die "failure setting db2 environment variable : DB2OPTIONS
rc=$rc\n";
}

if ( $platform eq "nt" )
{
   $rc=&dodb_noconn("db2set DB2NTNOCACHE=ON",$all_ln);
}

if ( $rc != 0 )
{
  die "failure setting db2 environment variable : DB2NTNOCACHE\n";
}
# @de980723wlc

# set the db2 env vars for loading, from the
TPCD_LOAD_DB2SET_SCRIPT script
if ( $loadsetScript ne "NULL" )
{
   if ( $platform eq "nt" )
   {
      #### Mike , th, the- for some reason rah on NT doesn't like run-
ning
      #### a fully qualified file name.  Switch to dir, then run
      ## check CHANGE THIS to have single node and multinode exe-
cution
      if (( $mode eq "uni" ) || ( $mode eq "smp" ))
      {
         $ret=system("${ddlpath}${delim}$loadsetScript");
         #$ret=system("cd ${ddlpath} $sep $loadsetScript");
      }
      else
      {
         $ret=system(" rah \" cd ${ddlpath} & $loadsetScript\" ");

      }
   }
   else
   {
#        $ret=system("${ddlpath}${delim}$loadsetScript");
      $ret=system("perl /usr/tpcd/custom/run_db2set.pl");
   }
   if ( $ret != 0 )
   {
      die "failure setting load time db2set parms from $loadsetScript
\n";
   }
}
# stopping and starting db2 before we continue
print "Stopping DB2 ...\n";
$rc=system("db2stop");
if ( $rc < 0 )
{
   die "failure during db2stop rc = $rc \n";
}
print "Starting DB2 ...\n";
$rc=system("db2start");
if ( $rc != 0 )
{
   die "failure during db2start rc = $rc \n";
}

########################################## create database
################################################
if ( $buildStage eq "ALL" )
{
   # build from the beginning
   &outtime("*** Starting to create the database");

   $rc = &dodb_noconn("db2 \"create database $dbname on $dbpath
collate using identity with 'TPC-D $sf GB'\"",$once);
   # remove the update.pair.num file so when setupDir runs, it doesn't
   # hang waiting for an answer on nt

   if($rc != 0){
            die "Failure during create database. rc = $rc\n";
   }


   &rm("$auditDir${delim}$dbname.$user.update.pair.num");

   # reset the db and dbm configuration before we start
   &dodb_noconn("db2 reset database configuration for
$dbname",$all_ln);
   &dodb_conn($dbname,"db2 alter bufferpool ibmdefaultbp size -1
$sep   \
            db2 grant connect on database to public $sep            \
            db2 grant dbadm on database to $dbname $sep             \
            db2 commit",$once);
```

```
    &dodb_noconn("db2 reset database manager configuration",$once);

    # update the log information first
    # set up the log directory before we do any index creation
    if ($logDirScript ne "NULL")
    {
        system ("perl $ddlpath${delim}$logDirScript");
    }
    elsif ( $logDir ne "NULL" )
    {
        &dodb_noconn("db2 update database configuration for $dbname
using newlogpath $logDir",$all_ln);
    }
    $setLogs=0;
    $setLogString="";
    if ( $logprimary ne "NULL" )
    {
        $setLogString.="db2 update db cfg for $dbname using logprimary
$logprimary";
        $setLogs=1;
    }
    if ( $logsecond ne "NULL" )
    {
        if ( $setLogs != 0 )
        {
            $setLogString.=" $sep ";
        }
        $setLogString.="db2 update db cfg for $dbname using logsecond
$logsecond";
        $setLogs=1;
    }
    if ( $logfilsiz ne "NULL" )
    {
        if ( $setLogs != 0 )
        {
            $setLogString.=" $sep ";
        }
        $setLogString.="db2 update db cfg for $dbname using logfilsiz
$logfilsiz";
        $setLogs=1;
    }
    if ( $setLogs != 0 )
    {
        $setLogString.=" $sep ";
    }
    $setLogString.="db2 update db cfg for $dbname using logbufsz
128";
    &dodb_noconn("$setLogString",$all_ln);


    # setup the load configuration
    &loadConfig;
}  # end of "CREATE DATABASE" phase

#die "Ending buildtpcd after create database.... \n";

if (( $buildStage eq "ALL" ) || ( $buildStage eq "CRTTBSP" ) ||
#    ( $buildStage eq "LOAD" ) ||
    (( $buildStage eq "INDEX" ) && ( $earlyindex eq "yes" )) )
{
    # create the nodegroups is there is a file specified
    if ( $nodegroupdef ne "NULL" )
    {
        #run the create bufferpool ddl
        &outtime("*** Creating the nodegroups");
        &dodb2file($dbname,"$ddlpath${delim}$nodegroupdef",$once);

    }
    # create the explain tables - these should go into userspace1 since
no
    # other tablespaces exist
    if ( $explainDDL ne "NULL" )
    {
        $explnPathFile="$ddlpath${delim}$explainDDL";
    }
    else
    {
        if ( $platform eq "ptx" )
        {
            $home=$ENV{"HOME"};
            $sqlpath="$home${delim}sqllib";
        }
        if ( $platform ne "nt" )
        {
            $sqlpath="~${delim}sqllib";
        }
        else
        {
            $sqlpath=$ENV{"DB2PATH"};
            $sqlpath="\"$sqlpath\"";
        }
        $explnPathFile="${sqlpath}${delim}misc${delim}EXPLAIN.DDL";
    }
    &dodb_conn($dbname,
            "db2 -tvf $explnPathFile $sep \
             db2 alter table explain_instance locksize table append on
$sep \
             db2 alter table explain_statement locksize table append on
$sep \
             db2 alter table explain_argument locksize table append on
$sep \
             db2 alter table explain_object locksize table append on $sep
\
             db2 alter table explain_operator locksize table append on
$sep \
             db2 alter table explain_predicate locksize table append on
$sep \
             db2 alter table explain_stream locksize table append on",
            $once);

    if ( $buffpooldef ne "NULL" )
    {
        #run the create bufferpool ddl
        &outtime("*** Creating the bufferpools");
        &dodb2file($dbname,"$ddlpath${delim}$buffpooldef",$once);


    }
    # create the tablespaces
    &outtime("*** Ready to start creating the tablespaces");
    &dodb2file($dbname,"$ddlpath${delim}$tbspddl",$once);
    # if we are in audit mode, then we must create the the tablespaces
and
    # tables for the update functions and we must generate the data for
the
    # update functions before we start timing the load.  (All activity
    # on the database after the table creation is started and before the
performance
    # tests are run must be included in load time
    # NOTE: we do not have to do this if we are building the qualifica-
tion database

    outtime("*** Start of audited Load Time - starting to create tables");

    # create/populate the staging tables
    if ( $stagingTbl ne "NULL" )
    {
        # staging tables must be created for both test and qualification
database
        # but they do not need to be populated for the qualification data-
base
```

```
      &dodb2file($dbname,"$ddlpath${delim}$stagingTbl",$once);
      if ( $qual ne "QUAL" )
      {
         if ( $preloadSampleUF ne "NULL" )
         {
            # preload the sample UF data for statistics gathering
            $rc = system ("perl $ddlpath${delim}$preloadSampleUF");
         }
         if ( $deleteSampleUF ne "NULL" )
         {
            # delete the sample rows now that stats have been gathered

&dodb2file($dbname,"$ddlpath${delim}$deleteSampleUF",$once);
         }
      }
   }
   &dodb2file($dbname,"$ddlpath${delim}$ddl",$once);

   # update the locksize on the non-updated tables to be table level
locking
   # update the tables for appendmode

   if ($appendOn eq "yes"){
      &dodb_conn($dbname,
            "db2 alter table tpcd.nation locksize table $sep \
             db2 alter table tpcd.region locksize table $sep \
             db2 alter table tpcd.customer locksize table $sep \
             db2 alter table tpcd.supplier locksize table $sep \
             db2 alter table tpcd.part locksize table $sep \
             db2 alter table tpcd.partsupp locksize table $sep \
             db2 alter table tpcd.lineitem append on $sep \
             db2 alter table tpcd.orders append on",
            $once);
   }
   else{
      &dodb_conn($dbname,
            "db2 alter table tpcd.nation locksize table $sep \
             db2 alter table tpcd.region locksize table $sep \
             db2 alter table tpcd.customer locksize table $sep \
             db2 alter table tpcd.supplier locksize table $sep \
             db2 alter table tpcd.part locksize table $sep \
             db2 alter table tpcd.partsupp locksize table",
            $once);
   }

   if ( $mode eq "mpp" )
   {
      #need parallel specific
      print "need to figure parallel specific creation of tmp\n";
   }
   mkdir("${delim}tmp${delim}$instance",0777);
}  # end of "CREATE TABLESPACE and TABLES" phase

#die "Ending buildtpcd after create tabspace and tables.... \n";

if (( $buildStage eq "ALL" ) || ( $buildStage eq "CRTTBSP" ) ||
   ( $buildStage eq "LOAD" ) ||
   (( $buildStage eq "INDEX" ) && ( $earlyindex eq "yes" )) )
{
   # do the load stage of the build, or if early index is on do
   # the index creation also
   # setup the load configuration
   if ( $buildStage ne "ALL"){ # we're restarting a build so reapply the
load config
            &loadConfig;
   }

   # if earlyindex requested, create indexes
   if ( $earlyindex eq "yes" )
   {
      &outtime("*** Starting to create indexes");
      &dodb2file($dbname,"$ddlpath${delim}$indexddl",$once);


      &outtime("*** Create index completed");
   }


   # start the dbgen and load.....call the specific mode for loading
(uni,smp,mln)
   if (( $mode eq "uni" ) || ( $mode eq "smp" ))
   {
      &outtime("*** Starting the load");
      # call the appropriate dbgen/load for uni/smp
      if (( $loadscript eq "NULL") || ( $loadscript eq "" ))
      {
         $rc = system("perl genloaduni $qual");
         if ( $rc != 0 )
         {
            die "genloaduni failed rc = $rc\n";
         }
      }
      else
      {
         &dodb2file_noconn("$ddlpath${delim}$loadscript",$once);
      }
   }
   elsif (( $mode eq "mln" ) || ( $mode eq "mpp" ))
   {
      &outtime("*** Starting the load");
      # call the appropriate dbgen/split/(sort)/load for mln/mpp

      if (( $loadscript eq "NULL") || ( $loadscript eq "" ))
      {
         $rc = system("perl genloadmpp $qual");
        if ( $rc != 0 )
        {
           die "genloadmpp failed rc = $rc\n";
        }
      }
      else
      {
         #$rc = &dodb2file_noconn("$ddlpath${delim}$loadscript $sf");
         $rc = &dodb2file_noconn("$ddlpath${delim}$loadscript");
      }

      if ( $rc != 0 )
      {
         die "doload for $dbname failed rc = $rc. Script is
$ddlpath${delim}$loadscript\n";
      }
   }

   else
   {
      die "TPCD_MODE not set to one of uni, smp, mln or mpp\n";
   }

   &outtime("*** Load complete");


   # verify that the audit directory exists
   $filename="$auditDir";
   if (-e $filename)
   {
     # set up the $auditDir/$dbname.$user.update.pair.num file
     # to start at update pair 1
     $filename="$auditDir${delim}$dbname.$user.update.pair.num";
   }
   else
```

```
    {
      mkdir ("$auditDir", 0775) || die "cannot mkdir $auditDir";
    }
    print "setting update pair num to 1\n";
    system("echo 1 > $filename");

}  # end all and load stage (and create index if early index was specified

if (( $buildStage eq "ALL" ) || ( $buildStage eq "CRTTBSP" ) ||
    ( $buildStage eq "LOAD" ) ||
    ( $buildStage eq "INDEX" ))
{
    if ( $buildStage eq "INDEX")
    { # we're restarting a build so reapply the load config
            &loadConfig;
    }
    # if indexes haven't been created, do so now
    if ( $earlyindex ne "yes" )
    {
        &outtime("*** Create index started");
        &dodb2file($dbname,"$ddlpath${delim}$indexddl",$once);

        &outtime("*** Create index completed");
    }
    if ( $extraindex ne "no" )
    {
        # use this additional file for indexes
        &outtime("*** Create index (part 2) started");
        &dodb2file($dbname,"$ddlpath${delim}$extraindex",$once);
        &outtime("*** Create index (part 2) completed");
    }

} # end create/load/index phase of the build

if (( $buildStage eq "ALL" ) ||  ( $buildStage eq "CRTTBSP" ) ||
    ( $buildStage eq "LOAD" ) ||
    ( $buildStage eq "INDEX" ) || ( $buildStage eq "RUNSTATS" ))
{
    if ( $buildStage eq "RUNSTATS")
    { # we're restarting a build so reapply the load config
            &loadConfig;
    }
    # if statistics not gathered on the load, run runstats  (we have to run the
    # stats at the same time as the index creation whether it be both during load,
    # or after load)
    # We need to run the runstats as well if we have specifed an extra index file
    # for "after load" indexes
    if (( $loadstats eq "no" ) || ( $earlyindex eq "no" ) || ( $extraindex ne "no" ))
    {
        # if loadstats not gathered, then index stats not gathered either.
        &outtime("*** Runstats started");

        if ( $runstatShort ne "NULL" )
        {
            # we've specified a second runstats file...This runstats file should do
            # runstats for all table except lineitem. The lineitem runstats command
            # should be left in the main runstats file.
            if ( $platform eq "aix" || $platform eq "sun" || $platform eq "ptx" )
            {
                print "runstats from $ddlpath${delim}$runstatShort running now\n";
                $rc = system("db2 -tvf \"$ddlpath${delim}$runstatShort\" > \"$auditDir${delim}tools${delim}runstatShort.out\"  & ");
                print "rc from runstatshort=$rc\n";
            }
            elsif ( $platform eq "nt" )
            {
                system("start db2 -tvf $ddlpath${delim}$runstatShort");
            }
            else
            {
                print "Don't know how to start in background on $platform platform\n";
                print "therefore running runstats serially\n";

&dodb2file($dbname,"$ddlpath${delim}$runstatShort",$once);
            }
        }
        # run the full runstats, or the remainder of what wasn't put into the short
        # runstats file. You should be sure that this runstats will take longer
        # than the short runstats that is running in the background, otherwise
        # setting the config will happen before this is done.

        &dodb2file($dbname,"$ddlpath${delim}$runstats",$once);
        &outtime("*** Runstats completed");
    }
} # end build phaose all/load/index/runstats

# add the RI
if ( $addRI ne "NULL" )
{
    &outtime("*** Adding RI contraints started");
    &dodb2file($dbname,"$ddlpath${delim}$addRI",$once);
    &outtime("*** Adding RI contraints completed");
}

#add the AST if it has been requested
if ( $astFile ne "NULL" )
{
    &outtime("*** Adding AST started");
    &dodb2file($dbname,"$ddlpath${delim}$astFile",$once);
    &outtime("*** Adding AST completed");
}

# check tbsp info
&dodb_conn($dbname,"db2 list tablespaces show detail",$once);

# set the configuration
&outtime("*** Set Configuration started");
&outtime("*** Setting degree of parallelism");
&dodb_noconn("db2 update database configuration for $dbname using dft_degree $smpdegree",$all_ln);
&dodb_noconn("db2 update database manager configuration using max_querydegree $smpdegree",$once);

&dodb2file_noconn("${ddlpath}${delim}$configfile",$all_ln);
&dodb2file_noconn("${ddlpath}${delim}$dbmconfig",$once);

if ( $agentpri ne "NULL" )
{
    &dodb_noconn("db2 update dbm cfg using AGENTPRI $agentpri",$once);
}

# set the db2 environment variables for running the benchmark
if ( $setScript ne "NULL" )
{
    if ( $platform eq "aix" || $platform eq "sun" || $platform eq "ptx")
```

```
    {
        $ret=system("${ddlpath}${delim}$setScript");
    }
    elsif ( $platform eq "nt" )
    {
        if (($mode eq "uni" ) || ($mode eq "smp" ))
        {
            $ret = system("perl ${ddlpath}${delim}$setScript");
        }
        else
        {
            $ret = system(" rah \" cd ${ddlpath} & $setScript\" ");
        }
    }
    if ( $ret != 0 )
    {
        die "failure setting runtime db2set parms from $setScript \n";
    }
}
# if logging is enabled, we must take a backup of the database
if ( $log eq "yes" )
{
    &dodb_noconn("db2 update database configuration for $dbname
using LOGRETAIN yes",$all_ln);
    print "\n NOTE:  DO NOT RESET THE DATABASE
CONFIGURATION or you will lose logretain\n";
    # force a connection to the database on all nodes to ensure
LOGRETAIN is
    # set in effect.
    # An error message will print to screen if the logretain is set properly
    # i.e. SQL116N A connection to or activation of database <database
name>
    # cannot  be made.
    # This is expected and the lack of this error message should be
seen as an
    # error in the database build.
    &dodb_conn($dbname,"db2 \"select count(*) from
tpcd.region\"",$all_ln);

    if ( $qual eq "QUAL" )
    {
        &outtime("*** Starting the backup");

        if (( $mode eq "mln" ) || ( $mode eq "mpp"))
        {
            # must back up catalog node first...assume node 00
            $rc=system("db2_all \'}]<<+000< db2 \"backup database
$dbname to $backupdir without prompting\" \' ");
            if ( $rc != 0 )
            {
                die "backup of catalog node failed rc = $rc\n";
            }
            # back up remaining nodes
            $rc=system("db2_all \'||}]<<-000< db2 backup database
$dbname to $backupdir without prompting\' ");
            if ( $rc != 0 )
            {
                die "backup of remaining nodes failed rc = $rc\n";
            }
        }
        else
        {
            &dodb_noconn("db2 backup database $dbname to $backup-
dir",$once);
        }
        &outtime("*** Finished the backup");
    }
    else
    {
        # This is the test database. Clause 3.1.4 states that "the test
sponsor is
        # not required to make or have backup copies of the test data-
base; however
        # all other mechanisms that guarantee durability of the qualifica-
tion
        # database must be enabled in the same way for the test data-
base".
        # According to this clause we do need to keep the backup of the
database.
        &dodb_noconn("db2dart $dbname /CHST /WHAT DBBP
OFF",$all_ln);
    }
}

# stop and restart the database to get config parms recognized
$rc=system("db2stop");
if ( $rc != 0 )
{
    die "failure during db2stop rc = $rc \n";
}
$rc=system("db2start");
if ( $rc != 0 )
{
    die "failure during db2start rc = $rc \n";
}

&outtime("*** Set Configuration completed");
&outtime("*** End of audited Load Time");

#create generated seeds
if ( $genSeed ne "no" )
{
  $rc = system("perl createmseedme.pl 1000");
  if ( $rc != 0 )
  {
    warn "createmseedme failed\n";
  }
}

$rc = system("perl buildtpcdbatch $qual");
if ( $rc != 0 )
{
  die "buildtpcdbatch failed rc=$rc\n";
}

if ( $RealAudit eq "yes" )
{
    # if we are in real audit mode then we have to do a number of things
    # set up the audit directory structure and the run directory structure
    # so that once we have completed the buildtpcd, we are ready to
run.
    # first remove any old "update pair number" file so we won't get
prompted
    # doing setupDir
    &rm("$auditDir${delim}tools${delim}tpcd.runsetup");
    system("perl setupRun");
    # before we stop the database for the final time
    # if we are in the real audit mode then run dbtables and dbcheck be-
fore
    # we print out the final notice that we are ready to run the perform-
ance tests
    # if we are building the qualification database then we will bind to
both
    # the dbname database and the qualification database
    if ( $qual eq "QUAL" )
    {
        $verifyType="q";
    }
    else
```

```
    {
        $verifyType="t";
    }
    system("perl tablesdb $verifyType");

&dodb2file($dbname,"$auditDir${delim}tools${delim}first10rows.sql",$o
nce);
}
# stop, restart and activate the database, if necessary

#Create Catalog info
$rc = system("perl catinfo.pl b");

if ( $rc != 0 )
{
  warn "catinfo failed!!!  rc = $rc\n";
}

$rc=system("db2stop");
if ( $rc != 0 )
{
    die "failure during db2stop rc = $rc \n";
}

&outtime("*** Ready to run the performance tests once the dbm has
restarted");

if ( $RealAudit ne "yes" )
{
    # if we are not in a real audit, then we can restart the database
manager
    # if we are in a real audit, then we don't want to do this until the
    # power test starts
    $rc=system("db2start");
    if ( $rc != 0 )
    {
        die "failure during db2start rc = $rc \n";
    }
    if ( $activate eq "yes" )
    {
        &dodb_noconn("activate database $dbname",$once);
    }
}

# finished creating the database
&outtime("*** Finished creating the database");

############################### subroutine for setting the load
configuration ##############################
sub loadConfig{

    &outtime("*** Setting LOAD configuration.");
    if ($loadconfigfile eq "NULL" || $loadDBMconfig eq "NULL")
    {
       &dodb_noconn("db2 update db cfg for $dbname using buffpage
$buffpage $sep \
            db2 update db cfg for $dbname using sortheap $sortheap
$sep \
            db2 update db cfg for $dbname using num_iocleaners $ioclnrs
$sep \
            db2 update db cfg for $dbname using num_ioservers
$ioservers $sep \
            db2 update db cfg for $dbname using util_heap_sz
$util_heap_sz $sep \
            db2 update db cfg for $dbname using chngpgs_thresh
$chngpgs",
            $all_ln);
       &dodb_noconn("db2 update dbm cfg using sheapthres $sheap-
thres",$once);
    }

    else
    {
        &dodb2file_noconn("$ddlpath${delim}$loadconfigfile",$all_ln);
        &dodb2file_noconn("$ddlpath${delim}$loadDBMconfig",$once);
    }
    &dodb_noconn("db2 terminate",$once);
    $rc=system("db2stop");
    if ( $rc != 0 )
    {
        die "failure during db2stop after resetting for load config rc = $rc
\n";
    }
    $rc=system("db2start");
    if ( $rc != 0 )
    {
        die "failure during db2start rc = $rc \n";
    }
    &outtime("*** Setting LOAD configuration DONE.");
}



1;
```

Appendix - C Query Text and Output

C.22 Qualification Query Output

Query 1

```
-- Query 01 - Var_0 Rev_01 - Pricing Summary Report Query

Tag: Q1     Stream: -1    Sequence number: 17

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
tpcd.lineitem
where
l_shipdate <= date ('1998-12-01') - 90 day
group by
l_returnflag,
l_linestatus
order by
l_returnflag,
l_linestatus

L_RETURNFLAG  L_LINESTATUS  SUM_QTY
SUM_BASE_PRICE       SUM_DISC_PRICE       SUM_CHARGE
AVG_QTY              AVG_PRICE            AVG_DISC
COUNT_ORDER
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----
A            F                37734107.000      56586554400.728
53758257134.869      55909065222.829              25.522
38273.130             0.050     1478493
N            F                  991417.000       1487504710.380
1413082168.054       1469649223.194              25.516
38284.468             0.050       38854
N            O                74476040.000      111701729697.737
106118230307.604     110367043872.494              25.502
38249.118             0.050     2920374
R            F                37719753.000      56568041380.900
53741292684.604      55889619119.832              25.506
38250.855             0.050     1478870


Number of rows retrieved is:     4
```

Query 2

```
-- Query 02 - Var_0 Rev_02 - Minimum Cost Supplier Query

Tag: Q2     Stream: -1    Sequence number: 2

select
s_acctbal,
s_name,
n_name,
p_partkey,
p_mfgr,
s_address,
s_phone,
s_comment
from
tpcd.part,
tpcd.supplier,
tpcd.partsupp,
tpcd.nation,
tpcd.region
where
p_partkey = ps_partkey
and s_suppkey = ps_suppkey
and p_size = 15
and p_type like '%BRASS'
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = 'EUROPE'
and ps_supplycost = (
select
min(ps_supplycost)
from
tpcd.partsupp,
tpcd.supplier,
tpcd.nation,
tpcd.region
where
p_partkey = ps_partkey
and s_suppkey = ps_suppkey
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = 'EUROPE'
)
order by
s_acctbal desc,
n_name,
s_name,
p_partkey
fetch first 100 rows only

S_ACCTBAL            S_NAME                    N_NAME
P_PARTKEY    P_MFGR                    S_ADDRESS
S_PHONE         S_COMMENT
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------
          9938.530  Supplier#000005359         UNITED KINGDOM
185358  Manufacturer#4            QKuHYh,vZGiwu2FWEJoLDx04
33-429-790-6131  blithely silent pinto beans are furiously. slyly final
deposits acros
          9937.840  Supplier#000005969         ROMANIA
108438  Manufacturer#1
ANDENSOSmk,miq23Xfb5RWt6dvUcvt6Qa         29-520-692-3537
carefully slow deposits use furiously. slyly ironic platelets above the
ironic
          9936.220  Supplier#000005250         UNITED KINGDOM
249  Manufacturer#4            B3rqp0xbSEim4Mpy2RH J
33-320-228-2957  blithely special packages are. stealthily express
deposits across the closely final instructi

          9923.770  Supplier#000002324         GERMANY
29821  Manufacturer#4            y3OD9UywSTOk
17-779-299-1839  quickly express packages breach quiet pinto beans.
requ
          9871.220  Supplier#000006373         GERMANY
43868  Manufacturer#5            J8fcXWsTqM                17-
813-485-8637  never silent deposits integrate furiously blit
```

9870.780 Supplier#000001286 GERMANY
81285 Manufacturer#2
YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH 17-516-924-4574
final theodolites cajole slyly special,
9870.780 Supplier#000001286 GERMANY
181285 Manufacturer#4
YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH 17-516-924-4574
final theodolites cajole slyly special,
9852.520 Supplier#000008973 RUSSIA
18972 Manufacturer#2 t5L67YdBYYH6o,Vz24jpDyQ9
32-188-594-7038 quickly regular instructions wake-- carefully unusual
braids into the expres
9847.830 Supplier#000008097 RUSSIA
130557 Manufacturer#2 xMe97bpE69NzdwLoX
32-375-640-3593 slyly regular dependencies sleep slyly furiously
express dep
9847.570 Supplier#000006345 FRANCE
86344 Manufacturer#1
VSt3rzk3qG698u6ld8HhOByvrTcSTSvQlDQDag 16-886-766-7945
silent pinto beans should have to snooze carefully along the final
reques
9847.570 Supplier#000006345 FRANCE
173827 Manufacturer#2
VSt3rzk3qG698u6ld8HhOByvrTcSTSvQlDQDag 16-886-766-7945
silent pinto beans should have to snooze carefully along the final
reques
9836.930 Supplier#000007342 RUSSIA
4841 Manufacturer#4 JOlK7C1,7xrEZSSOw
32-399-414-5385 final accounts haggle. bold accounts are furiously
dugouts. furiously silent asymptotes are slyly
9817.100 Supplier#000002352 RUSSIA
124815 Manufacturer#2 4LfoHUZjgjEbAKw
TgdKcgOc4D4uCYw 32-551-831-1437 blithely pending
packages across the ironic accounts grow slyly after the furiously
9817.100 Supplier#000002352 RUSSIA
152351 Manufacturer#3 4LfoHUZjgjEbAKw
TgdKcgOc4D4uCYw 32-551-831-1437 blithely pending
packages across the ironic accounts grow slyly after the furiously
9739.860 Supplier#000003384 FRANCE
138357 Manufacturer#2 o,Z3v4POifevE k9U1b 6J1ucX,I
16-494-913-5925 slyly ironic theodolites hag
9721.950 Supplier#000008757 UNITED KINGDOM
156241 Manufacturer#3 Atg6GnM4dT2 33-
821-407-2995 ironic, even dolphins above the furiously ironic foxes
sleep slyly around the caref
9681.330 Supplier#000008406 RUSSIA
78405 Manufacturer#1 ,qUuXcftUl 32-139-
873-8571 furiously even deposits affix thinly special theodolites. furiou
9643.550 Supplier#000005148 ROMANIA
107617 Manufacturer#1 kT4ciVFslx9z4s79p Js825
29-252-617-4850 doggedly even ideas boost furiously against the
furiously express
9624.820 Supplier#000001816 FRANCE
34306 Manufacturer#3
e7vab91vLJPWxxZnewmnDBpDmxYHrb 16-392-237-6726
blithely regular accounts cajole furiously. regular
9624.780 Supplier#000009658 ROMANIA
189657 Manufacturer#1 oE9uBgEfSS4opIcepXyAYM,x
29-748-876-2014 regular deposits haggle. furiously express asympto
9612.940 Supplier#000003228 ROMANIA
120715 Manufacturer#2
KDdpNKN3cWu7ZSrbdqp7AfSLxx,qWB 29-325-784-8187
carefully pending accounts serve. furiously close deposits boost slyly.
q
9612.940 Supplier#000003228 ROMANIA
198189 Manufacturer#4
KDdpNKN3cWu7ZSrbdqp7AfSLxx,qWB 29-325-784-8187
carefully pending accounts serve. furiously close deposits boost slyly.
q
9571.830 Supplier#000004305 ROMANIA
179270 Manufacturer#2 qNHZ7WmCzygwMPRDO9Ps
29-973-481-1831 furiously final deposits
9558.100 Supplier#000003532 UNITED KINGDOM
88515 Manufacturer#4
EOeuiiOn21OVpTlGguufFDFsbN1p0lhpxHp 33-152-301-2164
daring, sly accounts breach about th
9492.790 Supplier#000005975 GERMANY
25974 Manufacturer#5 S6mIiCTx82z7lV 17-
992-579-4839 always pending packages boost slyly.
9461.050 Supplier#000002536 UNITED KINGDOM
20033 Manufacturer#1 8mmGbyzaU
7ZS2wJumTibypncu9pNkDc4FYA 33-556-973-5522 even foxes
are quickly furiously express requests. packages
9453.010 Supplier#000000802 ROMANIA
175767 Manufacturer#1 ,6HYXb4uaHITmtMBj4Ak57Pd
29-342-882-6463 final, regular packages across the slowly regular
packag
9408.650 Supplier#000007772 UNITED KINGDOM
117771 Manufacturer#4 AiC5YAH,gdu0i7
33-152-491-1126 blithely final ideas sleep carefully. requests are
9359.610 Supplier#000004856 ROMANIA
62349 Manufacturer#5 HYogcF3Jb yh1 29-
334-870-9731 carefully unusual packages sleep carefully even ideas.
dogged accoun
9357.450 Supplier#000006188 UNITED KINGDOM
138648 Manufacturer#1 g801,ssP8wpTk4Hm
33-583-607-1633 carefully regular deposits wake carefully furiously
even i
9352.040 Supplier#000003439 GERMANY
170921 Manufacturer#4 qYPDgoiBGhCYxjgC
17-128-996-4650 fluffily regular pinto beans wake. unusual, final ideas
c
9312.970 Supplier#000007807 RUSSIA
90279 Manufacturer#5 oGYMPCk9XHGB2PBfKRnHA
32-673-872-5854 unusual asymptotes above the
9312.970 Supplier#000007807 RUSSIA
100276 Manufacturer#5 oGYMPCk9XHGB2PBfKRnHA
32-673-872-5854 unusual asymptotes above the
9280.270 Supplier#000007194 ROMANIA
47193 Manufacturer#3 zhRUQkBSrFYxIAXTfInj vyGRQjeK
29-318-454-2133 slyly ironic requests despite the unusual ins
9274.800 Supplier#000008854 RUSSIA
76346 Manufacturer#3 1xhLoOUM7I3mZ1mKnerw
OSqdbb4QbGa 32-524-148-5221 ruthlessly ironic instructions
along the regular, furious requests integrate car

9249.350 Supplier#000003973 FRANCE
26466 Manufacturer#1
d18GiDsL6Wm2IsGXM,RZf1jCsgZAOjNYVThTRP4 16-722-866-
1658 quickly ironic sauternes use b
9249.350 Supplier#000003973 FRANCE
33972 Manufacturer#1
d18GiDsL6Wm2IsGXM,RZf1jCsgZAOjNYVThTRP4 16-722-866-
1658 quickly ironic sauternes use b
9208.700 Supplier#000007769 ROMANIA
40256 Manufacturer#5 rsimdze 5o9P Ht7xS 29-
964-424-9649 furiously ruthless epitaphs among the furiously regular
accounts use slowly fluffily ev
9201.470 Supplier#000009690 UNITED KINGDOM
67183 Manufacturer#5 CB BnUTlmi5zdeEl7R7
33-121-267-9529 blithely unusual accounts integrate slyly. platelets
9192.100 Supplier#000000115 UNITED KINGDOM
85098 Manufacturer#3 nJ 2t0f7Ve,wL1,6WzGBJLNBUCKlsV
33-597-248-1220 slyly bold pinto beans boost across the furiously
regular packages. carefully regu
9189.980 Supplier#000001226 GERMANY
21225 Manufacturer#4 qsLCqSvLyZfuXlpjz 17-
725-903-1381 final, express instruction

9128.970 Supplier#000004311 RUSSIA
146768 Manufacturer#5 I8IjnXd7NSJRs594RxsRR0
32-155-440-7120 regular pinto beans sleep ca
9104.830 Supplier#000008520 GERMANY
150974 Manufacturer#4 RqRVDgD0ER J9 b41vR2,3
17-728-804-1793 deposits sleep carefully e
9101.000 Supplier#000005791 ROMANIA
128254 Manufacturer#5 zub2zCV,jhHPPQqi,P2INAjE1zI
n66cOEoXFG 29-549-251-5384 carefully ironic packages after the
9094.570 Supplier#000004582 RUSSIA
39575 Manufacturer#1 WB0XkCSG3r,mnQ
n,h9VIxjjr9ARHFvKgMDf 32-587-577-1351 asymptotes above the
slyly even requests haggle furiously about the regular accounts
8996.870 Supplier#000004702 FRANCE
102191 Manufacturer#5 8XVcQK23akp 16-
811-269-8946 stealthy requests haggle c
8996.140 Supplier#000009814 ROMANIA
139813 Manufacturer#2
af0O5pg83lPU4IDVmEylXZVqYZQzSDlYLAmR 29-995-571-8781
ironic theodolites are evenly unusual requests-- pending pinto beans
across the in
8968.420 Supplier#000010000 ROMANIA
119999 Manufacturer#5 aTGLEusCiL4F
PDBdv665XBJhPyCOB0i 29-578-432-2146 furiously final ideas
believe furiously. furiously final ideas
8936.820 Supplier#000007043 UNITED KINGDOM
109512 Manufacturer#1
FVajceZInZdbJE6Z9XsRUxrUEpiwHDrOXi,1Rz 33-784-177-8208
furiously regular excuses wake after the blithely special pinto beans?
even instructions sl
8929.420 Supplier#000008770 FRANCE
173735 Manufacturer#4 R7cG26TtXrHAP9 HckhfRi
16-242-746-9248 final accounts sleep furiously. blithely ironic foxes
wake boldly across the furiously s
8920.590 Supplier#000003967 ROMANIA
26460 Manufacturer#1 eHoAXe62SY9 29-
194-731-3944 quickly even requests should have to affix blithely-- fur
8920.590 Supplier#000003967 ROMANIA
173966 Manufacturer#2 eHoAXe62SY9
29-194-731-3944 quickly even requests should have to affix blithely--
fur
8913.960 Supplier#000004603 UNITED KINGDOM
137063 Manufacturer#2
OUzlvMUr7n,utLxmPNeYKSf3T24OXskxB5 33-789-255-7342
slyly ironic packages detect furious accounts. ironic de
8877.820 Supplier#000007967 FRANCE
167966 Manufacturer#5 A3pi1BARM4nx6R,qrwFoRPU
16-442-147-9345 final deposits after the silent deposits ha
8862.240 Supplier#000003323 ROMANIA
73322 Manufacturer#3 W9 lYcsC9FwBqk3ItL
29-736-951-3710 unusual, pending theodolites integrate furiously slyly
even pinto beans. unusual sheaves sleep befor
8841.590 Supplier#000005750 ROMANIA
100729 Manufacturer#5
Erx3lAgu0g62iaHF9x50uMH4EgeN9hEG 29-344-502-5481
excuses after the blithely regular packages mold carefully deposits.
regular a
8781.710 Supplier#000003121 ROMANIA
13120 Manufacturer#5 wNqTogx238ZYCamFb,50v,bj
4IbNFW9Bvw1xP 29-707-291-5144 packages are quickly after the
final, even packages. furiously regular
8754.240 Supplier#000009407 UNITED KINGDOM
179406 Manufacturer#4 CHRCbkaWcf5B
33-903-970-9604 regular dependencies haggle across the carefully
bold
8691.060 Supplier#000004429 UNITED KINGDOM
126892 Manufacturer#2 k,BQms5UhoAF1B2Asi,fLib
33-964-337-5038 quickly special foxes against the furiously silent
platelets wake quickly after t
8655.990 Supplier#000006330 RUSSIA
193810 Manufacturer#2 UozlaENr0ytKe2w6CeIEWFWn
iO3S8Rae7Ou 32-561-198-3705 blithely even packages alongside
8638.360 Supplier#000002920 RUSSIA
75398 Manufacturer#1 Je2a8bszf3L 32-
122-621-7549 express deposits wake. furiously silent requests wake
carefully silent instru
8638.360 Supplier#000002920 RUSSIA
170402 Manufacturer#3 Je2a8bszf3L 32-
122-621-7549 express deposits wake. furiously silent requests wake
carefully silent instru
8607.690 Supplier#000006003 UNITED KINGDOM
76002 Manufacturer#2
EH9wADcEiuenM0NR08zDwMidw,52Y2RyILEiA 33-416-807-5206
always special foxes wake slyly bold, ironic accounts. ironic
instructions affix carefull
8569.520 Supplier#000005936 RUSSIA
5935 Manufacturer#5 jXaNZ6vwnEWJ2ksLZJpjtgt0bY2a3AU
32-644-251-7916 packages sleep furiously. special requests about
the fluffily even accounts detect
8564.120 Supplier#000000033 GERMANY
110032 Manufacturer#1
gfeKpYw3400L0SDywXA6Ya1Qmq1w6YB9f3R 17-138-897-9374
ironic instructions are. special pearls above
8553.820 Supplier#000003979 ROMANIA
143978 Manufacturer#4 BfmVhCAnCMY3jzpjUMy4CNWs9
HzpdQR7INJU 29-124-646-4897 express, ironic pinto beans cajole
around the express, even packages. qu
8517.230 Supplier#000009529 RUSSIA
37025 Manufacturer#5 e44R8o7JAIS9iMcr
32-565-297-8775 furiously silent requests cajole furiously furiously
ironic foxes. slyly express p

8517.230 Supplier#000009529 RUSSIA
59528 Manufacturer#2 e44R8o7JAIS9iMcr
32-565-297-8775 furiously silent requests cajole furiously furiously
ironic foxes. slyly express p
8503.700 Supplier#000006830 RUSSIA
44325 Manufacturer#4 BC4WFCYRUZyaIgchU 4S
32-147-878-5069 quickly regular excuses detect evenly around
8457.090 Supplier#000009456 UNITED KINGDOM
19455 Manufacturer#1 7SBhZs8gP1cJjT0Qf433YBk
33-858-440-4349 carefully final accounts sleep blithely special foxes.
slyly regular pinto beans alon
8441.400 Supplier#000003817 FRANCE
141302 Manufacturer#2 hU3fz3xL78 16-
339-356-5115 blithely blithe ideas are
8432.890 Supplier#000003990 RUSSIA
191470 Manufacturer#1
wehBBp1RQbfxAYDASS75MsywmsKHRVdkrvNe6m 32-839-509-
9301 final requests along the blithely ironic packages kindle against
the carefully fina
8431.400 Supplier#000002675 ROMANIA
5174 Manufacturer#1
HJFStOu9R5NGPOegKhgbzBdyvrG2yh8w 29-474-643-1443
express, final deposits cajole carefully. stealthily unusual requests
8407.040 Supplier#000005406 RUSSIA
162889 Manufacturer#4 j7 gYF5RW8DC5UrjKC
32-626-152-4621 quickly final sheaves boost. car
8386.080 Supplier#000008518 FRANCE
36014 Manufacturer#3
2jqzqqAVe9crMVGP,n9nTsQXulNLTUYoJjEDcqWV 16-618-780-7481
slyly ironic theodolites are slyly. dogged, pendin
8376.520 Supplier#000005306 UNITED KINGDOM
190267 Manufacturer#5 9t8Y8
QqSIsoADPt6NLdk,TP5zyRx41oBUlgoGc9 33-632-514-7931
furiously even instructions integrate during the furiously regular re
8348.740 Supplier#000008851 FRANCE
66344 Manufacturer#4 nWxi7GwEbjhw1 16-

796-240-2472 ironic instructions nag slyly against the slyly even theodolites. requests alongside of
8338.580 Supplier#000007269 FRANCE
17268 Manufacturer#4 ZwhJSwABUoiB04,3
16-267-277-4365 ruthlessly regular asymptotes a
8328.460 Supplier#000001744 ROMANIA
69237 Manufacturer#5 oLo3fV64q2,FKHa3p,qHnS7Yzv,ps8
29-330-728-5873 blithely silent excuses are slyly above the furiously even courts
8307.930 Supplier#000003142 GERMANY
18139 Manufacturer#1 dqblvV8dCNAorGlJ
17-595-447-6026 theodolites sleep blithely carefully regular warhorses. slyly regular ins
8231.610 Supplier#000009558 RUSSIA
192000 Manufacturer#2 mcdgen,yT1iJDHDS5fV
32-762-137-5858 slyly regular theodolites sleep fluffily express depos
8152.610 Supplier#000002731 ROMANIA
15227 Manufacturer#4 nluXJCuY1tu 29-805-463-2030 gifts use. slyly silent ideas are carefully beneath the silent instructions. slyly sil
8109.090 Supplier#000009186 FRANCE
99185 Manufacturer#1 wgfosrVPexl9pEXWywaqlBMDYYf
16-668-570-1402 quickly pending requests are blithely along the ironic, final requests; instr
8102.620 Supplier#000003347 UNITED KINGDOM
18344 Manufacturer#5 m CtXS2S16i 33-454-274-8532 packages grow special orbits. regular theodolites about the carefully pe
8046.070 Supplier#000008780 FRANCE
191222 Manufacturer#3 AczzuE0UK9osj ,Lx0Jmh
16-473-215-6395 regular epitaphs integrate slyly.
8042.090 Supplier#000003245 RUSSIA
135705 Manufacturer#4
Dh8Ikg39onrbOL4DyTfGw8a9oKUX3d9Y 32-836-132-8872 carefully regular instructions integrate blithely silent foxes. furiously express instructions haggl
8042.090 Supplier#000003245 RUSSIA
150729 Manufacturer#1
Dh8Ikg39onrbOL4DyTfGw8a9oKUX3d9Y 32-836-132-8872 carefully regular instructions integrate blithely silent foxes. furiously express instructions haggl
7992.400 Supplier#000006108 FRANCE
118574 Manufacturer#1 8tBydnTDwUqfBfFV4l3
16-974-998-8937 regular pinto beans are after
7980.650 Supplier#000001288 FRANCE
13784 Manufacturer#4 zE,7HgVPrCn 16-646-464-8247 unusual pinto beans cajole furiously according t
7950.370 Supplier#000008101 GERMANY
33094 Manufacturer#5 kkYvL6IuvojJgTNG
IKkaXQDYgx8ILohj 17-627-663-8014 quickly regular requests are furiously. pending deposits wake
7937.930 Supplier#000009012 ROMANIA
83995 Manufacturer#2 iUiTziH,Ek3i4lwSgunXMgrcTzwdb
29-250-925-9690 blithely bold ideas haggle quickly final, regular request
7914.450 Supplier#000001013 RUSSIA
125988 Manufacturer#2 riRcntps4KEDtYScjpMIWeYF6mNnR
32-194-698-3365 final, ironic theodolites alongside of the ironic
7912.910 Supplier#000004211 GERMANY
159180 Manufacturer#5
2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG 17-266-947-7315 final requests integrate slyly above the silent, even
7912.910 Supplier#000004211 GERMANY
184210 Manufacturer#4
2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG 17-266-947-7315 final requests integrate slyly above the silent, even
7894.560 Supplier#000007981 GERMANY
85472 Manufacturer#4 NSJ96vMROAbeXP
17-963-404-3760 regular, even theodolites integrate carefully. bold, special theodolites are slyly fluffily iron
7887.080 Supplier#000009792 GERMANY
164759 Manufacturer#3 Y28ITVeYriT3kIGdV2K8fSZ
V2UqT5H1Otz 17-988-938-4296 pending, ironic packages sleep among the carefully ironic accounts. quickly final accounts
7871.500 Supplier#000007206 RUSSIA
104695 Manufacturer#1 3w fNCnrVmvJjE95sgWZzvW
32-432-452-7731 furiously dogged pinto beans cajole. bold, express notornis until the slyly pending
7852.450 Supplier#000005864 RUSSIA
8363 Manufacturer#4 WCNfBPZeSXh3h,c
32-454-883-3821 blithely regular deposits
7850.660 Supplier#000001518 UNITED KINGDOM
86501 Manufacturer#1 ONda3YJiHKJOC
33-730-383-3892 furiously final accounts wake carefully idle requests. even dolphins wake acc
7843.520 Supplier#000006683 FRANCE
11680 Manufacturer#4 2Z0JGkiv01Y00oCFwUGfvilbhzCdy
16-464-517-8943 carefully bold accounts doub

Number of rows retrieved is: 100

Query 3

-- Query 03 - Var_0 Rev_01 - Shipping Priority Query

Tag: Q3 Stream: -1 Sequence number: 11

```
select
l_orderkey,
sum(l_extendedprice * (1 - l_discount)) as revenue,
o_orderdate,
o_shippriority
from
tpcd.customer,
tpcd.orders,
tpcd.lineitem
where
c_mktsegment = 'BUILDING'
and c_custkey = o_custkey
and l_orderkey = o_orderkey
and o_orderdate < date ('1995-03-15')
and l_shipdate > date ('1995-03-15')
group by
l_orderkey,
o_orderdate,
o_shippriority
order by
revenue desc,
o_orderdate
fetch first 10 rows only
```

L_ORDERKEY	REVENUE	O_ORDERDATE	O_SHIPPRIORITY
2456423	406181.011	1995-03-05	0
3459808	405838.699	1995-03-04	0
492164	390324.061	1995-02-19	0
1188320	384537.936	1995-03-09	0
2435712	378673.056	1995-02-26	0
4878020	378376.795	1995-03-12	0
5521732	375153.922	1995-03-13	0
2628192	373133.309	1995-02-22	0
993600	371407.459	1995-03-05	0
2300070	367371.145	1995-03-13	0

Number of rows retrieved is: 10

Query 4

```
-- Query 04 - Var_0 Rev_01 - Order Priority Checking Query

Tag: Q4  Stream: -1  Sequence number: 14

select
o_orderpriority,
count(*) as order_count
from
tpcd.orders
where
o_orderdate >= date ('1993-07-01')
and o_orderdate < date ('1993-07-01') + 3 month
and exists (
select
*
from
tpcd.lineitem
where
l_orderkey = o_orderkey
and l_commitdate < l_receiptdate
)
group by
o_orderpriority
order by
o_orderpriority

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

Number of rows retrieved is:  5
```

Query 5

```
-- Query 05 - Var_0 Rev_02 Local Supplier Volume Query

Tag: Q5  Stream: -1  Sequence number: 20

select
n_name,
sum(l_extendedprice * (1 - l_discount)) as revenue
from
tpcd.customer,
tpcd.orders,
tpcd.lineitem,
tpcd.supplier,
tpcd.nation,
tpcd.region
where
c_custkey = o_custkey
and o_orderkey = l_orderkey
and l_suppkey = s_suppkey
and c_nationkey = s_nationkey
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = 'ASIA'
and o_orderdate >= date ('1994-01-01')
and o_orderdate < date ('1994-01-01') + 1 year
group by
n_name
order by
revenue desc

N_NAME                    REVENUE
-----------------------------------------------------
INDONESIA                  55502041.170
VIETNAM                    55295086.997
CHINA                      53724494.257
INDIA                      52035512.000
JAPAN                      45410175.695

Number of rows retrieved is:  5
```

Query 6

```
-- Query 06 - Var_0 Rev_01 - Forecasting Revenue Change Query

Tag: Q6  Stream: -1  Sequence number: 5

select
sum(l_extendedprice * l_discount) as revenue
from
tpcd.lineitem
where
l_shipdate >= date ('1994-01-01')
and l_shipdate < date ('1994-01-01') + 1 year
and l_discount between .06 - 0.01 and .06 + 0.01
and l_quantity < 24

REVENUE
----------------------
     123141078.228

Number of rows retrieved is:  1
```

Query 7

```
-- Query 07 - Var_0 Rev_01 - Volume Shipping Query

Tag: Q7  Stream: -1  Sequence number: 21

select
supp_nation,
cust_nation,
l_year,
sum(volume) as revenue
from
(
select
n1.n_name as supp_nation,
n2.n_name as cust_nation,
year (l_shipdate) as l_year,
l_extendedprice * (1 - l_discount) as volume
from
tpcd.supplier,
tpcd.lineitem,
tpcd.orders,
tpcd.customer,
tpcd.nation n1,
tpcd.nation n2

where
s_suppkey = l_suppkey
and o_orderkey = l_orderkey
```

```
and c_custkey = o_custkey
and s_nationkey = n1.n_nationkey
and c_nationkey = n2.n_nationkey
and (
(n1.n_name = 'FRANCE' and n2.n_name = 'GERMANY')
or (n1.n_name = 'GERMANY' and n2.n_name = 'FRANCE')
)
and l_shipdate between date('1995-01-01') and date('1996-12-31')
) as shipping
group by
supp_nation,
cust_nation,
l_year
order by
supp_nation,
cust_nation,
l_year
```

```
SUPP_NATION          CUST_NATION          L_YEAR
REVENUE
-------------------------------------------------------------------------------
FRANCE               GERMANY              1995
54639732.734
FRANCE               GERMANY              1996
54633083.308
GERMANY              FRANCE               1995
52531746.670
GERMANY              FRANCE               1996
52520549.022
```

Number of rows retrieved is: 4

Query 8

```
-- Query 08 - Var_0 Rev_01 - National Market Share Query

Tag: Q8  Stream: -1  Sequence number: 8

select
o_year,
sum(case
when nation = 'BRAZIL' then volume
else 0
end) / sum(volume) as mkt_share
from
(
select
year(o_orderdate) as o_year,
l_extendedprice * (1 - l_discount) as volume,
n2.n_name as nation
from
tpcd.part,
tpcd.supplier,
tpcd.lineitem,
tpcd.orders,
tpcd.customer,
tpcd.nation n1,
tpcd.nation n2,
tpcd.region
where
p_partkey = l_partkey
and s_suppkey = l_suppkey
and l_orderkey = o_orderkey
and o_custkey = c_custkey
and c_nationkey = n1.n_nationkey
and n1.n_regionkey = r_regionkey
and r_name = 'AMERICA'
and s_nationkey = n2.n_nationkey
and o_orderdate between date('1995-01-01') and date ('1996-12-31')
and p_type = 'ECONOMY ANODIZED STEEL'
) as all_nations
group by
o_year
order by
o_year
```

```
O_YEAR      MKT_SHARE
------------------------------------
      1995           0.034
      1996           0.041
```

Number of rows retrieved is: 2

Query 9

```
-- Query 09 - Var_0 Rev_01 - Product Type Profit Measure Query

Tag: Q9  Stream: -1  Sequence number: 3

select
nation,
o_year,
sum(amount) as sum_profit
from
(
select
n_name as nation,
year(o_orderdate) as o_year,
l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity as
amount
from
tpcd.part,
tpcd.supplier,
tpcd.lineitem,
tpcd.partsupp,
tpcd.orders,
tpcd.nation
where
s_suppkey = l_suppkey
and ps_suppkey = l_suppkey
and ps_partkey = l_partkey
and p_partkey = l_partkey
and o_orderkey = l_orderkey
and s_nationkey = n_nationkey
and p_name like '%green%'
) as profit
group by
nation,
o_year
order by
nation,
o_year desc
```

```
NATION               O_YEAR      SUM_PROFIT
-------------------------------------------------------------
ALGERIA                 1998       31342867.234
ALGERIA                 1997       57138193.023
ALGERIA                 1996       56140140.133
ALGERIA                 1995       53051469.653
ALGERIA                 1994       53867582.129
ALGERIA                 1993       54942718.132
ALGERIA                 1992       54628034.713
ARGENTINA                 1998       30211185.708
ARGENTINA                 1997       50805741.752
```

ARGENTINA	1996	51923746.575
ARGENTINA	1995	49298625.767
ARGENTINA	1994	50835610.109
ARGENTINA	1993	51646079.177
ARGENTINA	1992	50410314.995
BRAZIL	1998	27217924.383
BRAZIL	1997	48378669.199
BRAZIL	1996	50482870.357
BRAZIL	1995	47623383.635
BRAZIL	1994	47840165.726
BRAZIL	1993	49054694.035
BRAZIL	1992	48667639.084
CANADA	1998	30379833.768
CANADA	1997	50465052.311
CANADA	1996	52560501.390
CANADA	1995	52375332.809
CANADA	1994	52600364.659
CANADA	1993	52644504.073
CANADA	1992	53932871.697
CHINA	1998	31075466.165
CHINA	1997	50551874.450
CHINA	1996	51039293.875
CHINA	1995	49287534.617
CHINA	1994	50851090.067
CHINA	1993	54229629.833
CHINA	1992	52400529.372
EGYPT	1998	29054433.386
EGYPT	1997	50627611.452
EGYPT	1996	49542212.845
EGYPT	1995	48311550.321
EGYPT	1994	49790644.736
EGYPT	1993	48904292.969
EGYPT	1992	49434932.619
ETHIOPIA	1998	28040717.267
ETHIOPIA	1997	47455009.866
ETHIOPIA	1996	46491097.573
ETHIOPIA	1995	46804449.301
ETHIOPIA	1994	48516143.917
ETHIOPIA	1993	46551891.563
ETHIOPIA	1992	44934648.643
FRANCE	1998	32226407.839
FRANCE	1997	47121485.860
FRANCE	1996	47263135.496
FRANCE	1995	47275997.571
FRANCE	1994	47067209.332
FRANCE	1993	51163370.106
FRANCE	1992	47846235.331
GERMANY	1998	28624942.660
GERMANY	1997	49309074.877
GERMANY	1996	49918683.168
GERMANY	1995	52650718.724
GERMANY	1994	50346900.423
GERMANY	1993	50991895.806
GERMANY	1992	48274126.099
INDIA	1998	29943144.354
INDIA	1997	50665453.231
INDIA	1996	50283092.291
INDIA	1995	50006774.645
INDIA	1994	48995190.756
INDIA	1993	50286902.853
INDIA	1992	50850329.402
INDONESIA	1998	27672339.996
INDONESIA	1997	50512145.726
INDONESIA	1996	51653060.117
INDONESIA	1995	51508779.594
INDONESIA	1994	52817950.322
INDONESIA	1993	47959994.955
INDONESIA	1992	51776605.032
IRAN	1998	29065736.238
IRAN	1997	50042063.054
IRAN	1996	50926653.188
IRAN	1995	51249667.648
IRAN	1994	50337085.865
IRAN	1993	51730763.490
IRAN	1992	49955856.563
IRAQ	1998	31624551.002
IRAQ	1997	55121749.019
IRAQ	1996	55897663.794
IRAQ	1995	54815472.517
IRAQ	1994	54408516.127
IRAQ	1993	53633167.977
IRAQ	1992	55891939.340
JAPAN	1998	27934179.670
JAPAN	1997	44517162.546
JAPAN	1996	42545606.120
JAPAN	1995	43749356.400
JAPAN	1994	44840243.070
JAPAN	1993	44660015.533
JAPAN	1992	45410249.122
JORDAN	1998	26901488.578
JORDAN	1997	45471878.410
JORDAN	1996	46794325.792
JORDAN	1995	45178828.576
JORDAN	1994	45333636.508
JORDAN	1993	47971496.098
JORDAN	1992	44717239.177
KENYA	1998	28597614.337
KENYA	1997	47949733.727
KENYA	1996	46886924.623
KENYA	1995	46072338.755
KENYA	1994	45772061.171
KENYA	1993	46308728.234
KENYA	1992	47257780.841
MOROCCO	1998	26732115.580
MOROCCO	1997	45637304.250
MOROCCO	1996	45558221.745
MOROCCO	1995	47851318.887
MOROCCO	1994	46272172.944
MOROCCO	1993	46764326.182
MOROCCO	1992	48122783.583
MOZAMBIQUE	1998	30712392.011
MOZAMBIQUE	1997	50316528.763
MOZAMBIQUE	1996	51640320.251
MOZAMBIQUE	1995	50693774.506
MOZAMBIQUE	1994	49253277.626
MOZAMBIQUE	1993	49153016.537
MOZAMBIQUE	1992	48247551.850
PERU	1998	29326102.320
PERU	1997	49753780.395
PERU	1996	50935170.293
PERU	1995	53309883.407
PERU	1994	50643531.797
PERU	1993	51584622.002
PERU	1992	47523899.055
ROMANIA	1998	30368667.400
ROMANIA	1997	50365683.853
ROMANIA	1996	49598999.015
ROMANIA	1995	47537642.870
ROMANIA	1994	51455283.009
ROMANIA	1993	50407136.892
ROMANIA	1992	48185385.129
RUSSIA	1998	28322384.027
RUSSIA	1997	50106685.182
RUSSIA	1996	51753342.430
RUSSIA	1995	49215820.365
RUSSIA	1994	52205666.441
RUSSIA	1993	51860230.034
RUSSIA	1992	53251677.153
SAUDI ARABIA	1998	31541259.810
SAUDI ARABIA	1997	52438750.808

```
SAUDI ARABIA                1996        52543737.820
SAUDI ARABIA                1995        52938696.533
SAUDI ARABIA                1994        51389601.967
SAUDI ARABIA                1993        52937508.882
SAUDI ARABIA                1992        54843459.641
UNITED KINGDOM              1998        28494874.004
UNITED KINGDOM              1997        49381810.899
UNITED KINGDOM              1996        51386853.960
UNITED KINGDOM              1995        51509586.788
UNITED KINGDOM              1994        48086499.712
UNITED KINGDOM              1993        49166827.224
UNITED KINGDOM              1992        49349122.082
UNITED STATES               1998        25126238.946
UNITED STATES               1997        50077306.419
UNITED STATES               1996        48048649.470
UNITED STATES               1995        48809032.423
UNITED STATES               1994        49296747.183
UNITED STATES               1993        48029946.801
UNITED STATES               1992        48671944.498
VIETNAM                     1998        30442736.059
VIETNAM                     1997        50309179.794
VIETNAM                     1996        50488161.410
VIETNAM                     1995        49658284.613
VIETNAM                     1994        50596057.261
VIETNAM                     1993        50953919.152
VIETNAM                     1992        49613838.315
```

Number of rows retrieved is: 175

Query 10

```
-- Query 10 - Var_0 Rev_01 - Returned Item Reporting Query

Tag: Q10    Stream: -1    Sequence number: 18

select
c_custkey,
c_name,
sum(l_extendedprice * (1 - l_discount)) as revenue,
c_acctbal,
n_name,
c_address,
c_phone,
c_comment
from
tpcd.customer,
tpcd.orders,
tpcd.lineitem,
tpcd.nation
where
c_custkey = o_custkey
and l_orderkey = o_orderkey
and o_orderdate >= date ('1993-10-01')
and o_orderdate < date ('1993-10-01') + 3 month
and l_returnflag = 'R'
and c_nationkey = n_nationkey
group by
c_custkey,
c_name,
c_acctbal,
c_phone,
n_name,
c_address,
c_comment
order by
revenue desc
fetch first 20 rows only
```

```
C_CUSTKEY    C_NAME                  REVENUE
C_ACCTBAL          N_NAME                  C_ADDRESS
C_PHONE         C_COMMENT
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
----
     57040  Customer#000057040               734235.246
632.870  JAPAN                    Eioyzjf4pp                      22-895-
641-3466  requests sleep blithely about the furiously i
    143347  Customer#000143347               721002.695
2557.470  EGYPT                    1aReFYv,Kw4                     14-
742-935-3718  fluffily bold excuses haggle finally after the u
     60838  Customer#000060838               679127.308
2454.770  BRAZIL                   64EaJ5vMAHWJlBOxJklpNc2RJiWE
12-913-494-9813  furiously even pinto beans integrate under the
ruthless foxes; ironic, even dolphins across the slyl
    101998  Customer#000101998               637029.567
3790.890  UNITED KINGDOM           01c9CILnNtfOQYmZj
33-593-865-6378  accounts doze blithely! enticing, final deposits sleep
blithely special accounts. slyly express accounts pla
    125341  Customer#000125341               633508.086
4983.510  GERMANY                  S29ODD6bceU8QSuuEJznkNaK
17-582-695-5962  quickly express requests wake quickly blithely
     25501  Customer#000025501               620269.785
7725.040  ETHIOPIA
W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ       15-874-808-6793
quickly special requests sleep evenly among the special deposits.
special deposi

    115831  Customer#000115831               596423.867
5098.100  FRANCE                   rFeBbEEyk dl
ne7zV5fDrmiq1oK09wV7pxqCgIc  16-715-386-3788  carefully bold
excuses sleep alongside of the thinly idle
     84223  Customer#000084223               594998.024
528.650  UNITED KINGDOM           nAVZCs6BaWap rrM27N
2qBnzc5WBauxbA       33-442-824-8191  pending, final ideas haggle
final requests. unusual, regular asymptotes affix according to the even
foxes.
     54289  Customer#000054289               585603.392
5583.020  IRAN                     vXCxoCsU0Bad5JQI ,oobkZ
20-834-292-4707  express requests sublate blithely regular requests.
regular, even ideas solve.
     39922  Customer#000039922               584878.113
7321.110  GERMANY
Zgy4s50l2GKN4pLDPBU8m342gIw6R            17-147-757-8036  even
pinto beans haggle. slyly bold accounts inte
      6226  Customer#000006226               576783.761
2230.090  UNITED KINGDOM
8gPu8,NPGkfyQQ0hcIYUGPIBWc,ybP5g,        33-657-701-3391
quickly final requests against the regular instructions wake blithely final
instructions. pa
       922  Customer#000000922               576767.533
3869.250  GERMANY
Az9RFaut7NkPnc5zSD2PwHgVwr4jRzq          17-945-916-9648
boldly final requests cajole blith
    147946  Customer#000147946               576455.132
2030.130  ALGERIA                  iANyZHjqhyy7Ajah0pTrYyhJ
10-886-956-3143  furiously even accounts are blithely above the
furiousl
    115640  Customer#000115640               569341.193
6436.100  ARGENTINA                Vtgfia9qI 7EpHgecU1X
11-411-543-4901  final instructions are slyly according to the
     73606  Customer#000073606               568656.858
1785.670  JAPAN                    xuR0Tro5yChDfOCrjkd2ol
22-437-653-6966  furiously bold orbits about the furiously busy
requests wake across the furiously quiet theodolites. d
```

110246 Customer#000110246 566842.981
7763.350 VIETNAM 7KzflgX MDOq7sOkI
31-943-426-9837 dolphins sleep blithely among the slyly final
142549 Customer#000142549 563537.237
5085.990 INDONESIA
ChqEoK43OysjdHbtKCp6dKqjNyvvi9 19-955-562-2398 regular,
unusual dependencies boost slyly; ironic attainments nag fluffily into
the unusual packages?
146149 Customer#000146149 557254.986
1791.550 ROMANIA s87fvzFQpU 29-
744-164-6487 silent, unusual requests detect quickly slyly regul
52528 Customer#000052528 556397.351
551.790 ARGENTINA NFztyTOR10UOJ
11-208-192-3205 unusual requests detect. slyly dogged theodolites
use slyly. deposit
23431 Customer#000023431 554269.536
3381.860 ROMANIA HgiV0phqhaIa9aydNoIlb
29-915-458-2654 instructions nag quickly. furiously bold accounts
cajol

Number of rows retrieved is: 20

Query 11

-- Query 11 - Var_0 Rev_01 - Important Stock Identification Query

Tag: Q11 Stream: -1 Sequence number: 15

```
select
ps_partkey,
sum(ps_supplycost * ps_availqty) as value
from
tpcd.partsupp,
tpcd.supplier,
tpcd.nation
where
ps_suppkey = s_suppkey
and s_nationkey = n_nationkey
and n_name = 'GERMANY'
group by
ps_partkey having
sum(ps_supplycost * ps_availqty) > (
select
sum(ps_supplycost * ps_availqty) * 0.0001000000
from
tpcd.partsupp,
tpcd.supplier,
tpcd.nation
where
ps_suppkey = s_suppkey
and s_nationkey = n_nationkey
and n_name = 'GERMANY'
)
order by
value desc
```

PS_PARTKEY	VALUE
129760	17538456.860
166726	16503353.920
191287	16474801.970
161758	16101755.540
34452	15983844.720
139035	15907078.340
9403	15451755.620
154358	15212937.880
38823	15064802.860
85606	15053957.150
33354	14408297.400
154747	14407580.680
82865	14235489.780
76094	14094247.040
222	13937777.740
121271	13908336.000
55221	13716120.470
22819	13666434.280
76281	13646853.680
85298	13581154.930
85158	13554904.000
139684	13535538.720
31034	13498025.250
87305	13482847.040
10181	13445148.750
62323	13411824.300
26489	13377256.380
96493	13339057.830
56548	13329014.970
55576	13306843.350
159751	13306614.480
92406	13287414.500
182636	13223726.740
199969	13135288.210
62865	13001926.940
7284	12945298.190
197867	12944510.520
11562	12931575.510
75165	12916918.120
97175	12911283.500
140840	12896562.230
65241	12890600.460
166120	12876927.220
9035	12863828.700
144616	12853549.300
176723	12832309.740
170884	12792136.580
29790	12723300.330
95213	12555483.730
183873	12550533.050
171235	12476538.300
21533	12437821.320
17290	12432159.500
156397	12260623.500
122611	12222812.980
139155	12220319.250
146316	12215800.610
171381	12199734.520
198633	12078226.950
167417	12046637.620
59512	12043468.760
31688	12034893.640
159586	12001505.840
8993	11963814.300
120302	11857707.550
43536	11779340.520
9552	11776909.160
86223	11772205.080
53776	11758669.650
131285	11616953.740
91628	11611114.830
169644	11567959.720
182299	11567462.050
33107	11453818.760
104184	11436657.440
67027	11419127.140
176869	11371451.710
30885	11369674.790

```
     54420      11345076.880
     72240      11313951.050
    178708       11294635.170
     81298      11273686.130
    158324       11243442.720
    117095       11242535.240
    176793       11237733.380
     86091      11177793.790
    116033       11145434.360
    129058       11119112.200
    193714       11104706.390
    117195       11077217.960
     49851      11043701.780
     19791      11030662.620
     75800      11012401.620
    161562       10996371.690
     10119      10980015.750
     39185      10970042.560
     47223      10950022.130
    175594       10942923.050
    111295       10893675.610
    155446       10852764.570
    156391       10839810.380
     40884      10837234.190
    141288       10837130.210
    152388       10830977.820
     33449      10830858.720
    149035       10826130.020
    162620       10814275.680
    118324       10791788.100
     38932      10777541.750
[rows deleted]
    139680        7939242.360
     31134       7938318.300
     45636       7937240.850
     56694       7936015.950
      8114      7933921.880
     71518       7932261.690
     72922       7930400.640
    146699        7929167.400
     92387       7928972.670
    186289        7928786.190
     95952       7927972.780
    196514        7927180.700
      4403      7925729.040
      2267      7925649.370
     45924       7925047.680
     11493       7916722.230
    104478        7916253.600
    166794        7913842.000
    161995        7910874.270
     23538       7909752.060
     41093       7909579.920
    112073        7908617.570
     92814       7908262.500
     88919       7907992.500
     79753       7907933.880
    108765        7905338.980
    146530        7905336.600
     71475       7903367.580
     36289       7901946.500
     61739       7900794.000
     52338       7898638.080
    194299        7898421.240
    105235        7897829.940
     77207       7897752.720
     96712       7897575.270
     10157       7897046.250
    171154        7896814.500
     79373       7896186.000
    113808        7893353.880
     27901       7892952.000
    128820        7892882.720
     25891       7890511.200
    122819        7888881.020
    154731        7888301.330
    101674        7879324.600
     51968       7879102.210
     72073       7877736.110
      5182      7874521.730

Number of rows retrieved is:   1048
```

Query 12

```
-- Query 12 - Var_0 Rev_02 - Shipping Modes and Order Priority
Query

Tag: Q12    Stream: -1   Sequence number: 22

select
l_shipmode,
sum(case
when o_orderpriority = '1-URGENT'
or o_orderpriority = '2-HIGH'
then 1
else 0
end) as high_line_count,
sum(case
when o_orderpriority <> '1-URGENT'
and o_orderpriority <> '2-HIGH'
then 1
else 0
end) as low_line_count
from
tpcd.orders,
tpcd.lineitem
where
o_orderkey = l_orderkey
and l_shipmode in ('MAIL', 'SHIP')
and l_commitdate < l_receiptdate
and l_shipdate < l_commitdate
and l_receiptdate >= date ('1994-01-01')
and l_receiptdate < date ('1994-01-01') + 1 year
group by
l_shipmode
order by
l_shipmode

L_SHIPMODE  HIGH_LINE_COUNT  LOW_LINE_COUNT
----------------------------------------------
MAIL                6202          9324
SHIP                6200          9262


Number of rows retrieved is:      2
```

Query 13

```
-- Query 13 - Var_0 Rev_01 - Customer Distribution Query

Tag: Q13    Stream: -1   Sequence number: 10

select
c_count,
count(*) as custdist
```

```
from
(
select
c_custkey,
count(o_orderkey)
from
tpcd.customer left outer join tpcd.orders on
c_custkey = o_custkey
and o_comment not like '%special%requests%'
group by
c_custkey
) as c_orders (c_custkey, c_count)
group by
c_count
order by
custdist desc,
c_count desc
```

```
C_COUNT      CUSTDIST
-----------------------
          0     50004
          9      6641
         10      6566
         11      6058
          8      5949
         12      5553
         13      4989
         19      4748
          7      4707
         18      4625
         15      4552
         17      4530
         14      4484
         20      4461
         16      4323
         21      4217
         22      3730
          6      3334
         23      3129
         24      2622
         25      2079
          5      1972
         26      1593
         27      1185
          4      1033
         28       869
         29       559
          3       398
         30       373
         31       235
          2       144
         32       128
         33        71
         34        48
         35        33
          1        23
         36        17
         37         7
         40         4
         38         4
         39         2
         41         1
```

Number of rows retrieved is: 42

Query 14

```
-- Query 14 - Var_0 Rev_01 - Promotion Effect Query

Tag: Q14    Stream: -1    Sequence number: 1

select
100.00 * sum(case

when p_type like 'PROMO%'
then l_extendedprice * (1 - l_discount)
else 0
end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
tpcd.lineitem,
tpcd.part
where
l_partkey = p_partkey
and l_shipdate >= date ('1995-09-01')
and l_shipdate < date ('1995-09-01') + 1 month
```

```
PROMO_REVENUE
---------------------
           16.381
```

Number of rows retrieved is: 1

Query 15

```
-- Query 15 - Var_a Rev_01 - Top Supplier Query

Tag: Q15a    Stream: -1    Sequence number: 16

with revenue (supplier_no, total_revenue) as (
select
l_suppkey,
sum(l_extendedprice * (1-l_discount))
from
tpcd.lineitem
where
l_shipdate >= date ('1996-01-01')
and l_shipdate < date ('1996-01-01') + 3 month
group by
l_suppkey
)
select
s_suppkey,
s_name,
s_address,
s_phone,
total_revenue
from
tpcd.supplier,
revenue
where
s_suppkey = supplier_no
and total_revenue = (
select
max(total_revenue)
from
revenue
)
order by
s_suppkey
```

```
S_SUPPKEY    S_NAME                    S_ADDRESS
S_PHONE         TOTAL_REVENUE
-------------------------------------------------------------------------------------------
-------------------------
```

8449 Supplier#000008449 Wp34zim9qYFbVctdW
20-469-856-8873 1772627.209

Number of rows retrieved is: 1

Query 16

-- Query 16 - Var_0 Rev_01 - Parts/Supplier Relationship Query

Tag: Q16 Stream: -1 Sequence number: 13

```
select
p_brand,
p_type,
p_size,
count(distinct ps_suppkey) as supplier_cnt
from
tpcd.partsupp,
tpcd.part
where
p_partkey = ps_partkey
and p_brand <> 'Brand#45'
and p_type not like 'MEDIUM POLISHED%'
and p_size in (49, 14, 23, 45, 19, 3, 36, 9)
and ps_suppkey not in (
select
s_suppkey
from
tpcd.supplier
where
s_comment like '%Customer%Complaints%'
)
group by
p_brand,
p_type,
p_size
order by
supplier_cnt desc,
p_brand,
p_type,
p_size
```

P_BRAND	P_TYPE	P_SIZE	SUPPLIER_CNT
Brand#41	MEDIUM BRUSHED TIN	3	28
Brand#54	STANDARD BRUSHED COPPER	14	27
Brand#11	STANDARD BRUSHED TIN	23	24
Brand#11	STANDARD BURNISHED BRASS	36	24
Brand#15	MEDIUM ANODIZED NICKEL	3	24
Brand#15	SMALL ANODIZED BRASS	45	24
Brand#15	SMALL BURNISHED NICKEL	19	24
Brand#21	MEDIUM ANODIZED COPPER	3	24
Brand#22	SMALL BRUSHED NICKEL	3	24
Brand#22	SMALL BURNISHED BRASS	19	24
Brand#25	MEDIUM BURNISHED COPPER	36	24
Brand#31	PROMO POLISHED COPPER	36	24
Brand#33	LARGE POLISHED TIN	23	24
Brand#33	PROMO POLISHED STEEL	14	24
Brand#35	PROMO BRUSHED NICKEL	14	24
Brand#41	ECONOMY BRUSHED STEEL	9	24
Brand#41	ECONOMY POLISHED TIN	19	24
Brand#41	LARGE PLATED COPPER	36	24
Brand#42	ECONOMY PLATED BRASS	3	24
Brand#42	STANDARD POLISHED TIN	49	24
Brand#43	PROMO BRUSHED TIN	3	24
Brand#43	SMALL ANODIZED COPPER	36	24
Brand#44	STANDARD POLISHED NICKEL	3	24
Brand#52	ECONOMY PLATED TIN	14	24
Brand#52	STANDARD BURNISHED NICKEL	3	24
Brand#53	MEDIUM ANODIZED STEEL	14	24
Brand#14	PROMO ANODIZED NICKEL	45	23
Brand#32	ECONOMY PLATED BRASS	9	23
Brand#52	SMALL ANODIZED COPPER	3	23
Brand#11	ECONOMY BRUSHED COPPER	45	20
Brand#11	ECONOMY PLATED BRASS	23	20
Brand#11	LARGE BRUSHED COPPER	49	20
Brand#11	LARGE POLISHED COPPER	49	20
Brand#12	STANDARD ANODIZED TIN	49	20
Brand#12	STANDARD PLATED BRASS	19	20
Brand#13	ECONOMY BRUSHED BRASS	9	20
Brand#13	ECONOMY BURNISHED STEEL	14	20
Brand#13	LARGE BURNISHED NICKEL	19	20
Brand#13	MEDIUM BURNISHED COPPER	36	20
Brand#13	SMALL BRUSHED TIN	45	20
Brand#13	STANDARD ANODIZED COPPER	3	20
Brand#13	STANDARD PLATED NICKEL	23	20
Brand#14	ECONOMY ANODIZED COPPER	14	20
Brand#14	ECONOMY PLATED TIN	36	20
Brand#14	ECONOMY POLISHED NICKEL	3	20
Brand#14	MEDIUM ANODIZED NICKEL	3	20
Brand#14	SMALL POLISHED TIN	14	20
Brand#15	MEDIUM ANODIZED COPPER	9	20
Brand#15	MEDIUM PLATED TIN	23	20
Brand#15	PROMO PLATED BRASS	14	20
Brand#15	SMALL ANODIZED COPPER	45	20
Brand#15	SMALL PLATED COPPER	49	20
Brand#15	STANDARD PLATED TIN	3	20
Brand#21	LARGE ANODIZED COPPER	36	20
Brand#21	LARGE BRUSHED TIN	3	20
Brand#21	MEDIUM ANODIZED COPPER	14	20
Brand#21	PROMO BRUSHED TIN	36	20
Brand#21	PROMO POLISHED NICKEL	45	20
Brand#21	SMALL ANODIZED COPPER	9	20
Brand#21	SMALL POLISHED NICKEL	23	20
Brand#22	LARGE ANODIZED COPPER	36	20
Brand#22	LARGE BRUSHED COPPER	49	20
Brand#22	PROMO ANODIZED TIN	49	20
Brand#22	PROMO POLISHED BRASS	45	20
Brand#22	SMALL BURNISHED STEEL	45	20
Brand#23	MEDIUM ANODIZED STEEL	45	20
Brand#23	PROMO POLISHED STEEL	23	20
Brand#23	STANDARD BRUSHED TIN	14	20
Brand#23	STANDARD PLATED NICKEL	36	20
Brand#24	PROMO PLATED COPPER	49	20
Brand#24	PROMO PLATED STEEL	49	20
Brand#24	PROMO POLISHED STEEL	9	20
Brand#24	STANDARD BRUSHED TIN	36	20
Brand#25	LARGE ANODIZED BRASS	3	20
Brand#25	PROMO BURNISHED TIN	3	20
Brand#31	ECONOMY POLISHED NICKEL	3	20
Brand#31	MEDIUM PLATED TIN	45	20
Brand#31	SMALL ANODIZED STEEL	14	20
Brand#32	ECONOMY ANODIZED COPPER	36	20
Brand#32	ECONOMY BRUSHED NICKEL	49	20
Brand#32	LARGE ANODIZED TIN	19	20
Brand#32	MEDIUM BURNISHED COPPER	19	20
Brand#32	SMALL ANODIZED STEEL	45	20
Brand#33	ECONOMY POLISHED COPPER	19	20
Brand#33	PROMO PLATED NICKEL	14	20
Brand#33	SMALL POLISHED TIN	9	20
Brand#33	STANDARD ANODIZED BRASS	49	20
Brand#33	STANDARD BURNISHED BRASS	45	20
Brand#34	ECONOMY BRUSHED NICKEL	49	20
Brand#34	LARGE BRUSHED BRASS	19	20
Brand#34	SMALL BRUSHED TIN	3	20
Brand#34	STANDARD PLATED COPPER	9	20

```
Brand#35    LARGE ANODIZED NICKEL          3          20
Brand#35    MEDIUM ANODIZED BRASS          45         20
Brand#35    MEDIUM ANODIZED STEEL          23         20
Brand#35    PROMO ANODIZED COPPER          49         20
Brand#35    SMALL POLISHED COPPER          14         20
Brand#41    LARGE ANODIZED STEEL           3          20
Brand#41    LARGE BRUSHED NICKEL           23         20
Brand#41    LARGE BURNISHED COPPER         3          20
Brand#41    MEDIUM PLATED STEEL            19         20
Brand#41    SMALL BURNISHED COPPER         23         20
Brand#42    MEDIUM BURNISHED BRASS         14         20
Brand#42    SMALL BURNISHED COPPER         3          20
Brand#43    ECONOMY POLISHED COPPER        9          20
Brand#43    SMALL PLATED STEEL             3          20
Brand#43    STANDARD BURNISHED TIN         23         20
Brand#44    LARGE ANODIZED STEEL           23         20
[rows deleted]
Brand#55    STANDARD BURNISHED TIN         36         4
Brand#55    STANDARD BURNISHED TIN         49         4
Brand#55    STANDARD PLATED BRASS          9          4
Brand#55    STANDARD PLATED BRASS          45         4
Brand#55    STANDARD PLATED BRASS          49         4
Brand#55    STANDARD PLATED COPPER         9          4
Brand#55    STANDARD PLATED COPPER         45         4
Brand#55    STANDARD PLATED NICKEL         3          4
Brand#55    STANDARD PLATED NICKEL         19         4
Brand#55    STANDARD PLATED NICKEL         45         4
Brand#55    STANDARD PLATED STEEL          14         4
Brand#55    STANDARD PLATED STEEL          23         4
Brand#55    STANDARD PLATED STEEL          49         4
Brand#55    STANDARD PLATED TIN            9          4
Brand#55    STANDARD PLATED TIN            14         4
Brand#55    STANDARD PLATED TIN            36         4
Brand#55    STANDARD POLISHED BRASS        3          4
Brand#55    STANDARD POLISHED BRASS        9          4
Brand#55    STANDARD POLISHED BRASS        23         4
Brand#55    STANDARD POLISHED COPPER       3          4
Brand#55    STANDARD POLISHED COPPER       23         4
Brand#55    STANDARD POLISHED COPPER       45         4
Brand#55    STANDARD POLISHED NICKEL       3          4
Brand#55    STANDARD POLISHED NICKEL       23         4
Brand#55    STANDARD POLISHED NICKEL       36         4
Brand#55    STANDARD POLISHED NICKEL       45         4
Brand#55    STANDARD POLISHED NICKEL       49         4
Brand#55    STANDARD POLISHED STEEL        14         4
Brand#55    STANDARD POLISHED STEEL        23         4
Brand#55    STANDARD POLISHED TIN          9          4
Brand#55    STANDARD POLISHED TIN          19         4
Brand#55    STANDARD POLISHED TIN          36         4
Brand#11    SMALL BRUSHED TIN              19         3
Brand#15    LARGE PLATED NICKEL            45         3
Brand#15    LARGE POLISHED NICKEL          9          3
Brand#21    PROMO BURNISHED STEEL          45         3
Brand#22    STANDARD PLATED STEEL          23         3
Brand#25    LARGE PLATED STEEL             19         3
Brand#32    STANDARD ANODIZED COPPER       23         3
Brand#33    SMALL ANODIZED BRASS           9          3
Brand#35    MEDIUM ANODIZED TIN            19         3
Brand#51    SMALL PLATED BRASS             23         3
Brand#52    MEDIUM BRUSHED BRASS           45         3
Brand#53    MEDIUM BRUSHED TIN             45         3
Brand#54    ECONOMY POLISHED BRASS         9          3
Brand#55    PROMO PLATED BRASS             19         3
Brand#55    STANDARD PLATED TIN            49         3
```

Number of rows retrieved is: 18314

Query 17

```
-- Query 17 - Var_0 Rev_01 - Small-Quantity-Order Revenue Query

Tag: Q17    Stream: -1    Sequence number: 6

select
sum(l_extendedprice) / 7.0 as avg_yearly
from
tpcd.lineitem,
tpcd.part
where
p_partkey = l_partkey
and p_brand = 'Brand#23'
and p_container = 'MED BOX'
and l_quantity < (
select
0.2 * avg(l_quantity)
from
tpcd.lineitem
where
l_partkey = p_partkey
)

AVG_YEARLY
----------
        348406.054


Number of rows retrieved is:      1
```

Query 18

```
-- Query 18 - Var_0 Rev_01 - Large Volume Customer Query

Tag: Q18    Stream: -1    Sequence number: 7

select
c_name,
c_custkey,
o_orderkey,
o_orderdate,

o_totalprice,
sum(l_quantity)
from
tpcd.customer,
tpcd.orders,
tpcd.lineitem
where
o_orderkey in (
select
l_orderkey
from
tpcd.lineitem
group by
l_orderkey having
sum(l_quantity) > 300
)
and c_custkey = o_custkey
and o_orderkey = l_orderkey
group by
c_name,
c_custkey,
o_orderkey,
o_orderdate,
o_totalprice
order by
o_totalprice desc,
```

```
o_orderdate
fetch first 100 rows only

C_NAME                  C_CUSTKEY    O_ORDERKEY
O_ORDERDATE  O_TOTALPRICE        6
-------------------------------------------------------------------------------------------------
---------------
Customer#000128120           128120     4722021  1994-04-07
544089.090           323.000
Customer#000144617           144617     3043270  1997-02-12
530604.440           317.000
Customer#000013940            13940     2232932  1997-04-13
522720.610           304.000
Customer#000066790            66790     2199712  1996-09-30
515531.820           327.000
Customer#000046435            46435     4745607  1997-07-03
508047.990           309.000
Customer#000015272            15272     3883783  1993-07-28
500241.330           302.000
Customer#000146608           146608     3342468  1994-06-12
499794.580           303.000
Customer#000096103            96103     5984582  1992-03-16
494398.790           312.000
Customer#000024341            24341     1474818  1992-11-15
491348.260           302.000
Customer#000137446           137446     5489475  1997-05-23
487763.250           311.000
Customer#000107590           107590     4267751  1994-11-04
485141.380           301.000
Customer#000050008            50008     2366755  1996-12-09
483891.260           302.000
Customer#000015619            15619     3767271  1996-08-07
480083.960           318.000
Customer#000077260            77260     1436544  1992-09-12
479499.430           307.000
Customer#000109379           109379     5746311  1996-10-10
478064.110           302.000
Customer#000054602            54602     5832321  1997-02-09
471220.080           307.000
Customer#000105995           105995     2096705  1994-07-03
469692.580           307.000
Customer#000148885           148885     2942469  1992-05-31
469630.440           313.000
Customer#000114586           114586      551136  1993-05-19
469605.590           308.000
Customer#000105260           105260     5296167  1996-09-06
469360.570           303.000
Customer#000147197           147197     1263015  1997-02-02
467149.670           320.000
Customer#000064483            64483     2745894  1996-07-04
466991.350           304.000
Customer#000136573           136573     2761378  1996-05-31
461282.730           301.000
Customer#000016384            16384      502886  1994-04-12
458378.920           312.000
Customer#000117919           117919     2869152  1996-06-20
456815.920           317.000
Customer#000012251            12251      735366  1993-11-24
455107.260           309.000
Customer#000120098           120098     1971680  1995-06-14
453451.230           308.000
Customer#000066098            66098     5007490  1992-08-07
453436.160           304.000
Customer#000117076           117076     4290656  1997-02-05
449545.850           301.000
Customer#000129379           129379     4720454  1997-06-07
448665.790           303.000
Customer#000126865           126865     4702759  1994-11-07
447606.650           320.000
Customer#000088876            88876      983201  1993-12-30
446717.460           304.000
Customer#000036619            36619     4806726  1995-01-17
446704.090           328.000
Customer#000141823           141823     2806245  1996-12-29
446269.120           310.000
Customer#000053029            53029     2662214  1993-08-13
446144.490           302.000
Customer#000018188            18188     3037414  1995-01-25
443807.220           308.000
Customer#000066533            66533       29158  1995-10-21
443576.500           305.000
Customer#000037729            37729     4134341  1995-06-29
441082.970           309.000
Customer#000003566             3566     2329187  1998-01-04
439803.360           304.000
Customer#000045538            45538     4527553  1994-05-22
436275.310           305.000
Customer#000081581            81581     4739650  1995-11-04
435405.900           305.000
Customer#000119989           119989     1544643  1997-09-20
434568.250           320.000
Customer#000003680             3680     3861123  1998-07-03
433525.970           301.000
Customer#000113131           113131      967334  1995-12-15
432957.750           301.000
Customer#000141098           141098      565574  1995-09-24
430986.690           301.000
Customer#000093392            93392     5200102  1997-01-22
425487.510           304.000
Customer#000015631            15631     1845057  1994-05-12
419879.590           302.000
Customer#000112987           112987     4439686  1996-09-17
418161.490           305.000
Customer#000012599            12599     4259524  1998-02-12
415200.610           304.000
Customer#000105410           105410     4478371  1996-03-05
412754.510           302.000
Customer#000149842           149842     5156581  1994-05-30
411329.350           302.000
Customer#000010129            10129     5849444  1994-03-21
409129.850           309.000
Customer#000069904            69904     1742403  1996-10-19
408513.000           305.000
Customer#000017746            17746        6882  1997-04-09
408446.930           303.000
Customer#000013072            13072     1481925  1998-03-15
399195.470           301.000
Customer#000082441            82441      857959  1994-02-07
382579.740           305.000
Customer#000088703            88703     2995076  1994-01-30
363812.120           302.000


Number of rows retrieved is:     57
```

Query 19

```
-- Query 19 - Var_0 Rev_01 - Discounted Revenue Query

Tag: Q19    Stream: -1   Sequence number: 19

select
sum(l_extendedprice* (1 - l_discount)) as revenue
from
tpcd.lineitem,
tpcd.part
where
```

```
(
p_partkey = l_partkey
and p_brand = 'Brand#12'
and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
and l_quantity >= 1 and l_quantity <= 1 + 10
and p_size between 1 and 5
and l_shipmode in ('AIR', 'AIR REG')
and l_shipinstruct = 'DELIVER IN PERSON'
)
or
(
p_partkey = l_partkey
and p_brand = 'Brand#23'
and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED
PACK')
and l_quantity >= 10 and l_quantity <= 10 + 10
and p_size between 1 and 10

and l_shipmode in ('AIR', 'AIR REG')
and l_shipinstruct = 'DELIVER IN PERSON'
)
or
(
p_partkey = l_partkey
and p_brand = 'Brand#34'
and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
and l_quantity >= 20 and l_quantity <= 20 + 10
and p_size between 1 and 15
and l_shipmode in ('AIR', 'AIR REG')
and l_shipinstruct = 'DELIVER IN PERSON'
)

REVENUE
---------------------
        3083843.058


Number of rows retrieved is:      1
```

Query 20

```
-- Query 20 - Var_0 Rev_01 - Potential Part Promotion Query

Tag: Q20    Stream: -1    Sequence number: 4

select
s_name,
s_address
from
tpcd.supplier,
tpcd.nation
where
s_suppkey in (
select
ps_suppkey
from
tpcd.partsupp
where
ps_partkey in (
select
p_partkey
from
tpcd.part
where
p_name like 'forest%'
)
and ps_availqty > (
select
0.5 * sum(l_quantity)
from
tpcd.lineitem
where
l_partkey = ps_partkey
and l_suppkey = ps_suppkey
and l_shipdate >= date ('1994-01-01')
and l_shipdate < date ('1994-01-01') + 1 year
)
)
and s_nationkey = n_nationkey
and n_name = 'CANADA'
order by
s_name

S_NAME                    S_ADDRESS
------------------------------------------------------------------------
Supplier#000000020        iybAE,RmTymrZVYaFZva2SH,j
Supplier#000000091
YV45D7TkfdQanOOZ7q9QxkyGUapU1oOWU6q3
Supplier#000000197        YC2Acon6kjY3zj3Fbxs2k4Vdf7X0cd2F
Supplier#000000226        83qOdU2EYRdPQAQhEtn GRZEd
Supplier#000000285        Br7e1nnt1yxrw6ImgpJ7YdhFDjuBf
Supplier#000000378        FfbhyCxWvcPrO8ltp9
Supplier#000000402        i9Sw4DoyMhzhKXCH9By,AYSgmD
Supplier#000000530        0qwCMwobKY OcmLyfRXlagA8ukENJv,
Supplier#000000688        D fw5ocppmZpYBBIPI718hCihLDZ5KhKX
Supplier#000000710        f19YPvOyb QoYwjKC,oPycpGfieBAcwKJo
Supplier#000000736
l6i2nMwVuovfKnuVgaSGK2rDy65DlAFLegiL7
Supplier#000000761
zlSLelQUj2XrvTTFnv7WAcYZGvvMTx882d4
Supplier#000000884        bmhEShejaS
Supplier#000000887        urEaTejH5POADP2ARrf
Supplier#000000935        ij98czM 2KzWe7dDTOxB8sq0UfCdvrX
Supplier#000000975        ,AC e,tBpNwKb5xMUzeohxlRn,
hdZJo73gFQF8y
Supplier#000001263        rQWr6nf8ZhB2TAiIDIvo5Io
Supplier#000001399        LmrocnIMSyYOWuANx7
Supplier#000001446        Ich9HMNU1R7a0LIybsUodVknk6
Supplier#000001454        TOpimgu2TVXljhiL93h,
Supplier#000001500        wDmF5xLxtQch9ctVu,
Supplier#000001602        uKNWIeafaM644
Supplier#000001626        UhxNRzUu1dtFmp0
Supplier#000001682        pXTkGxrTQVyH1Rr
Supplier#000001699        Q9C4rfJ26oijVPqqcqVXeRI
Supplier#000001700        7hMlCof1Y5zLFg
Supplier#000001726        TeRY7TtTH24sEword7yAaSkjx8
Supplier#000001730        Rc8e,1Pybn r6zo0VJIEiD0UD vhk
Supplier#000001746
qWsendlOekQG1aW4uq06uQaCm51se8lirv7 hBRd
Supplier#000001752        Fra7outx41THYJaRThdOGiBk
Supplier#000001856
jXcRgzYF0ah05iR8p6w5SbJJLcUGyYiURPvFwUWM
Supplier#000001931        FpJbMU2h6ZR2eBv8I9NIxF
Supplier#000001939        Nrk,JA4bfReUs
Supplier#000001990
DSDJkCgBJzuPg1yuM,CUdLnsRliOxkkHezTCA
Supplier#000002020        jB6r1d7MxP6co
Supplier#000002022        dwebGX7Id2pc25YvY33
Supplier#000002036        20ytTtVObjKUUI2WCB0A
Supplier#000002204        uYmlr46C06udCqanj0KiRsoTQakZsEyssL
Supplier#000002243        nSOEV3JeOU79
Supplier#000002245        hz2qWXWVjOyKhqPYMoEwz6zFkrTaDM
Supplier#000002282        ES21K9dxoW1I1TzWCj7ekdlNwSWnv1Z
6mQ,BKn
Supplier#000002303        nCoWfpB6YOymbgOht7ltfklpkHl
Supplier#000002373        RzHSxOTQmElCjxIBiVA52Z JB58rJhPRylR
Supplier#000002419        qydBQd14l5l5mVXa4fYY
Supplier#000002481        nLKHUOn2Ml9TOA06Znq9GEMcIlMO2
Supplier#000002571        JZUugz04c iJFLrlGsz9O N,W 1rVHNIReyq
```

Supplier#000002585 CsPoKpw2QuTY4AV1NkWuttneIa4SN
Supplier#000002630 ZIQAvjNUY9KH5ive zm7k VlPiDl7CCo21
Supplier#000002719 4nnzQI2CbqREQUuIsXTBVUkaP4mNS3
Supplier#000002721 HVdFAN2JHMQSpKm
Supplier#000002730 lIFxR4fzm31C6,muzJwl84z
Supplier#000002775 yDclaDaBD4ihH
Supplier#000002853 rTNAOltXka
Supplier#000002875 6JgMi 9Qt6VmwL3Ltt1SRlKww0keLQ,RAZa
Supplier#000002934 m,trBENywSArwg3DhB
Supplier#000002941 Naddba 8YTEKekZyP0
Supplier#000002960 KCPCEsRGGo6vx8TygHh60nAYf9rStQT2T
Supplier#000002980 B9k9yVsyaXvWktOSHezqHiAEp9id0SKzkw
Supplier#000003062 LSQNgqY1xnOzz9zBCapy7HwOZQ
Supplier#000003087 ANwe8QsZ4rgj1HSqVz991eWQ
Supplier#000003089 s5b VCIZqMSZVa r g7LTdcg29GbTE7rI1x
Supplier#000003095 HxON3jJhUi3zjt,r mTD
Supplier#000003201 E87yws6I,t0qNs4QW7UzExKiJnJDZWue
Supplier#000003213 pxrRP4irQ1VoyfQ,dTf3
Supplier#000003241 j06SU,LS9O3mwjAMOViANeIhb
Supplier#000003275 9xO4nyJ2QJcX6vGf
Supplier#000003288 EDdfNt7E5Uc,xLTupoIgYL4yY7ujh,
Supplier#000003313
El2I7we,049SPrvomUm4hZwJoOhZkvLxLJXgVH
Supplier#000003314
jnisU8MzqO4iUB3zsPcrysMw3DDUojS4q7LD
Supplier#000003380 jPv0V,pszouuFT3YsAqlP,kxT3u,gTFiEbRt,x
Supplier#000003403 e3X2o ,KCG9tsHji8A XXCxiF2hZWBw
Supplier#000003421 Sh3dt9W5oeofFWovnFhrg,
Supplier#000003441 zvFJIzS,oUuShHjpcX
Supplier#000003590 sy79CMLxqb,Cbo
Supplier#000003607 lNqFHQYjwSAkf
Supplier#000003625 qY588W0Yk5iaUy1RXTgNrEKrMAjBYHcKs
Supplier#000003656 eEYmmO2gmD JdfG32XtDgJV,db56
Supplier#000003782 iVsPZg7bk06TqNMwi0LKbLUrC1zmrg
Supplier#000003918 meRvRCsJoAbfqd0Re4
Supplier#000003941 Pmb05mQfBMS618O7WKqZJ 9vyv
Supplier#000003994 W00LZp3NjK0
Supplier#000004005 V723F1wCy2eA4OgIu8TjBtOVUHp
Supplier#000004033 ncsAhv9Je,kFXTNjfb2
Supplier#000004140 0hL7DJyYjcHL
Supplier#000004165 wTJ2dZNQA8P2oi99N6DT47ndHy,XKD2
Supplier#000004207 tF64pwiOM4IkWjN3mS,e06WuAjLx
Supplier#000004236
dl,HPtJmGipxYsSqn9wmqkuWjst,mCeJ8O6T
Supplier#000004246 Xha aXQF7u4qU3LsHD
Supplier#000004278 bBddbpBxIVp Di9
Supplier#000004343 GK3sbopqrQEkWLMvVBFCG
Supplier#000004346 S3076LEOwo
Supplier#000004388 VfZ l1J,mwp4aS
Supplier#000004406
Ah0ZaLu6VwufPWUz,7kbXgYZhauEaHqGIg
Supplier#000004430 yvSsKNSTL5HLXBET4IuOsPNLxKzAMk
Supplier#000004522 xXtCKwsZDArxIBGDfzX2PgobGZsBg
Supplier#000004527 p pVXCnxgcklWF6A1o3OHY3qW6
Supplier#000004542 NJSbLJDroYG2y1r3rDiKg
Supplier#000004574 1HvGwnVueZ5CIndc
Supplier#000004655 67NqBc4 t3PG3F8aO IsqWNq4kGaPowYL
Supplier#000004701 6jX4u47URzIMHf
Supplier#000004711 bEzjp1QdQu ls2ERMxv0km vn6bu2zXlL1
Supplier#000004987 UFx1upJ8MvOvgFjA8
Supplier#000005000 DeX804 w0H8FrCUvahgy ilbuzBX3NK
Supplier#000005100 OfvYPs3Io,wEvvLHNaLuCX
Supplier#000005192 JDp4rhXiDw0kf6RH
Supplier#000005195 Woi3b2ZaicPh ZSfu1EfXhE
Supplier#000005283 5fxYXxwXy,TQX,MqDC2hxzyQ
Supplier#000005300 gXG28YqpxU
Supplier#000005386 Ub6AAfHpWLWP
Supplier#000005426 9Dz2OVT1q sb4BK71ljQ1XjPBYRPvO
Supplier#000005484 saFdOR
qW7AFY,3asPqiiAa11Mo22pCoN0BtPrKo
Supplier#000005505 d2sbjG43KwMPX
Supplier#000005506 On f5ypzoWgB
Supplier#000005516
XsN99Ks9wEvcohU6jRD2MeebQFf76mD8vovuY
Supplier#000005536 Nzo9tGkpgbHT,EZ4D,77MYKl4ah1C
Supplier#000005605 7Vj6Eil0mThqkM
Supplier#000005631 14TVrjlzo2SJEBYCDgpMwTlvwSqC
Supplier#000005730 5rkb0PSews HvxkL8JaD41UpnSF2cg8H1
Supplier#000005736 2dq XTYhtYWSfp
Supplier#000005737 dmEWcS32C3kx,d,B95 OmYn48
Supplier#000005797 ,o,OebwRbSDmVl9gN9fpWPCiqB UogvlSR
Supplier#000005836 tx3SjPD2ZuWGFBRH,
Supplier#000005875
lK,sYiGzB94hSyHy9xvSZFbVQNCZe2LXZuGbS
Supplier#000005974 REhR5jE,lLusQXvf54SwYySgsSSVFhu
Supplier#000005989 rjFY,5kgLpBu7c
Supplier#000006059 4m0cv8MwJ9yX2vlwI Z
Supplier#000006065 UiI2Cy3W4Tu5sLk LuvXLRy6KihlGv
Supplier#000006070 TaIC5m0pDrO6DZbngfmGmqe
Supplier#000006109 rY5gbfh3dKHnylcQUTPGCwnbe
Supplier#000006121 S92ycWwEzYYw4GspCBJN1WMuHhoZ
Supplier#000006215 j2iEbTsl,5PWdqWZ7k1yIlSb7qtiiZljDIPEo
Supplier#000006217 RVN23SYT9jenUeaWGXUd
Supplier#000006274 S3yTZWqxTKUq g QQgcW9
AqhCkNZsW51hHuwU
Supplier#000006435 xIgE69XszYbnO4Eon7cHHO8y
Supplier#000006463 7 wkdj2EO49iotley2kmIM
ADpLSszGV3RNWj
Supplier#000006493 ojV f,sNaB6Hm7r,fknDVTL63raJgAjZK
Supplier#000006521 b9 2zjHzxR
Supplier#000006607 3F 2e2gqD5u5B
Supplier#000006706 Ak4ga,ePu1QZ6C3qkrqjosaX0gxvqS9vkbe
Supplier#000006761 n4jhxGMqB5prD1HhpLvwrWStOLlla
Supplier#000006808 HGd2Xo 9nEcHJhZvXjXxWKIpApT
Supplier#000006858 fnlINT885vBBhsWwTGiZ0o22thwGY16h
GHJj21
Supplier#000006872 XIDPiA7PLXCWK6SeEcld
Supplier#000006949 mLxYUJhsGcLtKe ,GFirNu183AvT
Supplier#000006985 PrUUiboQpy,OtgJ01Z4BxJQUyrw9c3l
Supplier#000007072 2tRyX9M1a 4Rcm57s779F1ANG9jlpK
Supplier#000007098
G3j8g0KC4OcbAu2OVoPHrXQWMCUdjq8wgCHOExu
Supplier#000007135 ls DoKV7V5ulfQy9V
Supplier#000007160
TqDGBULB3cTqIT6FKDvm9BS4e4v,zwYiQPb
Supplier#000007169 tEc95D2moN9S84nd55O,dlnW
Supplier#000007322 wr7dgte5q MAjiY0uwmi3MyDkSMX1
Supplier#000007365 51xhROLvQMJ05DndtZWt
Supplier#000007398 V8eE6oZ00OFNU,
Supplier#000007402 4UVv58ery1rjmqSR5
Supplier#000007448 yhhpWiJi7EJ6Q5VCaQ
Supplier#000007477 9m9j0wfhWzCvVHxkU,PpAxwSH0h
Supplier#000007509 q8,V6LJRoHJjHcOuSG7aLTMg
Supplier#000007561 rMcFg2530VC
Supplier#000007789 rQ7cUcPrtudOyO3svNSkimqH6qrfWT2Sz
Supplier#000007801 69fi,U1r6enUb
Supplier#000007818 yhhc2CQec Jrvc8zqBi83
Supplier#000007885 u3sicchh5ZpyTUpN1cJKNcAoabIWgY
Supplier#000007918 r,v9mBQ6LoEYyj1
Supplier#000007926 ErzCF80K9Uy
Supplier#000007957 ELwnio14ssoU1 dRyZIL OK3Vtzb
Supplier#000007965 F7Un5lJ7p5hhj
Supplier#000007968
DsF9UlZ2Fo6HXN9aErvyg1ikHoD582HSGZpP
Supplier#000007998 LnASFBfYRFOo9d6d,asBvVq9Lo2P
Supplier#000008168 aOa82a8ZbKCnfDLX
Supplier#000008231 IK7eGw Yj90sTdpsP,vcqWxLB
Supplier#000008243 2AyePMkDqmzVzjGTizXthFLo8h
EiudCMxOmIIG
Supplier#000008275 BlbNDfWg,gpXKQlLN

```
Supplier#000008323      75I18sZmASwm
POeheRMdj9tmpyeQ,BfCXN5BIAb
Supplier#000008366
h778cEj14BuW9OEKlvPTWq4iwASR6EBBXN7zeS8
Supplier#000008423      RQhKnkAhR0DAr3lx4Q1weMMn00hNe Kq
Supplier#000008480      4sSDA4ACReklNjEm5T6b
Supplier#000008532      Uc29q4,5xVdDOF87UZrxhr4xWS0ihEUXuh
Supplier#000008595      MH0iB73GQ3z UW3O DbCbqmc
Supplier#000008610      SgVgP90vP452sUNTgzL9zKwXHXAzV6tV
Supplier#000008705      aE,trRNdPx,4yinTD9O3DebDIp
Supplier#000008742      HmPlQEzKCPEcTUL14,kKq
Supplier#000008841      I 85Lu1sekbg2xrSIzm0
Supplier#000008895      2cH4okfaLSZTTg8sKRbbJQxkmeFu2Esj
Supplier#000008967      2kwEHyMG
7FwozNImAUE6mH0hYtqYculJM
Supplier#000008972      w2vF6 D5YZO3visPXsqVfLADTK
Supplier#000009032      qK,trB6Sdy4Dz1BRUFNy
Supplier#000009147      rOAuryHxpZ9eOvx
Supplier#000009252      F7cZaPUHwh1 ZKyj3xmAVWC1XdP
ue1p5m,i
Supplier#000009278      RqYTzgxj93CLX 0mcYfCENOefD
Supplier#000009327      uoqMdf7e7Gj9dbQ53
Supplier#000009430      igRqmneFt
Supplier#000009567      r4Wfx4c3xsEAjcGj71HHZByornl D9vrztXlv4
Supplier#000009601      51m637bO,Rw5DnHWFUvLacRx9
Supplier#000009709      rRnCbHYgDgl9PZYnyWKVYSUW0vKg
Supplier#000009753      wLhVEcRmd7PkJF4FBnGK7Z
Supplier#000009796      z,y4Idmr15DOvPUqYG
Supplier#000009799      4wNjXGa4OKWl
Supplier#000009811      E3iuyq7UnZxU7oPZIe2Gu6
Supplier#000009812      APFRMy3lCbgFga53n5t9DxzFPQPgnjrGt32
Supplier#000009862      rJzweWeN58
Supplier#000009868      ROjGgx5gvtkmnUUoeyy7v
Supplier#000009869
ucLqxzrpBTRMewGSM29t0rNTM30g1Tu3Xgg3mKag
Supplier#000009899      7XdpAHrzr1t,UQFZE
Supplier#000009974      7wJ,J5DKcxSU4Kp1cQLpbcAvB5AsvKT


Number of rows retrieved is:    204
```

Query 21

```
-- Query 21 - Var_0 Rev_01 - Suppliers Who Kept Orders Waiting
Query


Tag: Q21    Stream: -1    Sequence number: 9

select
s_name,
count(*) as numwait
from

tpcd.supplier,
tpcd.lineitem l1,
tpcd.orders,
tpcd.nation
where
s_suppkey = l1.l_suppkey
and o_orderkey = l1.l_orderkey
and o_orderstatus = 'F'
and l1.l_receiptdate > l1.l_commitdate
and exists (
select
*
from
tpcd.lineitem l2
where
l2.l_orderkey = l1.l_orderkey
and l2.l_suppkey <> l1.l_suppkey
)
and not exists (
select
*
from
tpcd.lineitem l3
where
l3.l_orderkey = l1.l_orderkey
and l3.l_suppkey <> l1.l_suppkey
and l3.l_receiptdate > l3.l_commitdate
)
and s_nationkey = n_nationkey
and n_name = 'SAUDI ARABIA'
group by
s_name
order by
numwait desc,
s_name
fetch first 100 rows only

S_NAME                    NUMWAIT
----------------------------------------
Supplier#000002829              20
Supplier#000005808              18
Supplier#000000262              17
Supplier#000000496              17
Supplier#000002160              17
Supplier#000002301              17
Supplier#000002540              17
Supplier#000003063              17
Supplier#000005178              17
Supplier#000008331              17
Supplier#000002005              16
Supplier#000002095              16
Supplier#000005799              16
Supplier#000005842              16
Supplier#000006450              16
Supplier#000006939              16
Supplier#000009200              16
Supplier#000009727              16
Supplier#000000486              15
Supplier#000000565              15
Supplier#000001046              15
Supplier#000001047              15
Supplier#000001161              15
Supplier#000001336              15
Supplier#000001435              15
Supplier#000003075              15
Supplier#000003335              15
Supplier#000005649              15
Supplier#000006027              15
Supplier#000006795              15
Supplier#000006800              15
Supplier#000006824              15
Supplier#000007131              15
Supplier#000007382              15
Supplier#000008913              15
Supplier#000009787              15
Supplier#000000633              14
Supplier#000001960              14

Supplier#000002323              14
Supplier#000002490              14
Supplier#000002993              14
Supplier#000003101              14
Supplier#000004489              14
Supplier#000005435              14
```

```
Supplier#000005583              14
Supplier#000005774              14
Supplier#000007579              14
Supplier#000008180              14
Supplier#000008695              14
Supplier#000009224              14
Supplier#000000357              13
Supplier#000000436              13
Supplier#000000610              13
Supplier#000000788              13
Supplier#000000889              13
Supplier#000001062              13
Supplier#000001498              13
Supplier#000002056              13
Supplier#000002312              13
Supplier#000002344              13
Supplier#000002596              13
Supplier#000002615              13
Supplier#000002978              13
Supplier#000003048              13
Supplier#000003234              13
Supplier#000003727              13
Supplier#000003806              13
Supplier#000004472              13
Supplier#000005236              13
Supplier#000005906              13
Supplier#000006241              13
Supplier#000006326              13
Supplier#000006384              13
Supplier#000006394              13
Supplier#000006624              13
Supplier#000006629              13
Supplier#000006682              13
Supplier#000006737              13
Supplier#000006825              13
Supplier#000007021              13
Supplier#000007417              13
Supplier#000007497              13
Supplier#000007602              13
Supplier#000008134              13
Supplier#000008234              13
Supplier#000009435              13
Supplier#000009436              13
Supplier#000009564              13
Supplier#000009896              13
Supplier#000000379              12
Supplier#000000673              12
Supplier#000000762              12
Supplier#000000811              12
Supplier#000000821              12
Supplier#000001337              12
Supplier#000001916              12
Supplier#000001925              12
Supplier#000002039              12
Supplier#000002357              12
Supplier#000002483              12
```

Number of rows retrieved is: 100

Query 22

```
-- Query 22 - Var_0 Rev_01 - Global Sales Opportunity Query

Tag: Q22   Stream: -1   Sequence number: 12

select
cntrycode,
count(*) as numcust,
sum(c_acctbal) as totacctbal
from
(
select
substr(c_phone, 1, 2) as cntrycode,
c_acctbal
from
tpcd.customer
where
substr(c_phone, 1, 2) in
('13', '31', '23', '29', '30', '18', '17')
and c_acctbal > (
select
avg(c_acctbal)
from
tpcd.customer
where
c_acctbal > 0.00
and substr(c_phone, 1, 2) in
('13', '31', '23', '29', '30', '18', '17')
)
and not exists (
select
*
from
tpcd.orders
where
o_custkey = c_custkey
)
) as custsale
group by
cntrycode
order by
cntrycode

CNTRYCODE  NUMCUST      TOTACCTBAL
-------------------------------------------
13             888         6737713.990
17             861         6460573.720
18             964         7236687.400
23             892         6701457.950
29             948         7158866.630
30             909         6808436.130
31             922         6806670.180


Number of rows retrieved is:      7
```

C.23 First 10 rows of Test Database Tables

```
SELECT * FROM TPCD.REGION FETCH FIRST 10 ROWS ONLY

R_REGIONKEY R_NAME                 R_COMMENT
----------- ------------------------ -------------------------------------------------------
-------------------------------------------------------------------------------
          0 AFRICA                   special Tiresias about the furiously even
dolphins are furi
          1 AMERICA                  even, ironic theodolites according to
the bold platelets wa
          2 ASIA                     silent, bold requests sleep slyly across the
quickly sly dependencies. furiously silent instructions alongside
          3 EUROPE                   special, bold deposits haggle foxes.
platelet
```

```
         4 MIDDLE EAST               furiously unusual packages use
carefully above the unusual, exp

  5 record(s) selected.


SELECT * FROM TPCD.NATION FETCH FIRST 10 ROWS ONLY

N_NATIONKEY N_NAME                 N_REGIONKEY N_COMMENT
----------- ------------------------ ----------- ------------------------------------------------------------------------------------
------------
         0 ALGERIA                             0 final accounts wake quickly.
special reques
         5 ETHIOPIA                            0 fluffily ruthless requests inte-
grate fluffily. pending ideas wake blithely acco
        14 KENYA                               0 ironic requests boost. quickly
pending pinto beans cajole slyly slyly even deposits. ironic packages
        15 MOROCCO                             0 ideas according to the fluff-
ily final pinto beans sleep furiously
        16 MOZAMBIQUE                          0 ironic courts wake fluffily
even, bold deposi
         1 ARGENTINA                           1 idly final instructions cajole
stealthily. regular instructions wake carefully blithely express accounts.
fluffi
         2 BRAZIL                              1 always pending pinto beans
sleep sil
         3 CANADA                              1 foxes among the bold re-
quests
        17 PERU                                1 final, final accounts sleep slyly
across the requests.
        24 UNITED STATES                       1 blithely regular deposits
serve furiously blithely regular warthogs! slyly fi

  10 record(s) selected.


SELECT * FROM TPCD.PART FETCH FIRST 10 ROWS ONLY

P_PARTKEY   P_NAME                                        P_MFGR
P_BRAND    P_TYPE                 P_SIZE      P_CONTAINER
P_RETAILPRICE           P_COMMENT
----------- ---------------------------------------------- ------------------------- -
--------- ------------------------- ----------- ----------- ------------------------ -----------
------------
        43 medium khaki chocolate rosy blush                  Manufac-
turer#4           Brand#44   PROMO POLISHED STEEL              5
WRAP CASE   9.43040000000000e+02 carefully iro
        47 sky firebrick red linen dim                        Manufacturer#4
Brand#45   LARGE BURNISHED BRASS             14 JUMBO PACK
9.47040000000000e+02 bold, unusual a
        89 ghost khaki lawn pale dim                          Manufac-
turer#5           Brand#53   STANDARD BURNISHED STEEL          7
MED JAR     9.89080000000000e+02 quickly ironi
        98 frosted goldenrod chartreuse dark honeydew         Manu-
facturer#5           Brand#54   STANDARD ANODIZED BRASS
22 MED JAR     9.98090000000000e+02 furiou
       119 olive black navajo burlywood ghost                 Manufac-
turer#4           Brand#43   LARGE POLISHED STEEL             30
WRAP CASE   1.01911000000000e+03 blithely pendin
       122 ghost royal chocolate peach ivory                  Manufac-
turer#2           Brand#21   MEDIUM ANODIZED TIN               8 LG
DRUM     1.02212000000000e+03 pinto beans w
       125 metallic indian magenta navy medium                Manu-
facturer#1           Brand#12   STANDARD BRUSHED BRASS
17 WRAP BAG    1.02512000000000e+03 carefully fina
       126 burnished navy indian medium deep                  Manufac-
turer#4           Brand#45   MEDIUM BRUSHED NICKEL             4
LG BAG      1.02612000000000e+03 slowly bold pin
       144 wheat brown orange almond aquamarine               Manu-
facturer#1           Brand#14   SMALL ANODIZED TIN               26
SM BOX      1.04414000000000e+03 blithely bold r
       169 tan lime sky almond midnight                       Manufac-
turer#5           Brand#55   STANDARD POLISHED BRASS          10
JUMBO CASE  1.06916000000000e+03 blithely

  10 record(s) selected.


SELECT * FROM TPCD.SUPPLIER FETCH FIRST 10 ROWS ONLY

S_SUPPKEY   S_NAME                 S_ADDRESS
S_NATIONKEY S_PHONE         S_ACCTBAL
S_COMMENT
----------- ------------------------ ---------------------------------------- ----------- -----
--------- ------------------------ ----------------------------------------------------------
----------------------------------------


        43 Supplier#000000043
Z5mLuAoTUEeKY5v22VnnA4D87Ao6jF2LvMYnlX8h          12 22-
421-568-4862 7.77341000000000e+03 slyly final accounts wake
blithely slyly regular requests. sl
        47 Supplier#000000047
3XM1x,Pcxqw,HK4XNlgbnZMbLhBHLA                    14 24-810-354-
4471 2.95809000000000e+03 regular deposits engage slyly with the
        89 Supplier#000000089        fhtzZcSorhud1
9 19-259-876-1014 1.63802000000000e+03 bold, even instructions
print carefully even courts. ironic,
        98 Supplier#000000098        ogHn8dpXB5Q
21 31-914-775-1978 5.87307000000000e+03 slyly regular requests
mold slyly regular depo
       119 Supplier#000000119
4CxBrM0o4yt6LYFxZIyZ89Xnf8LZNn6KcYc               18 28-558-264-
1202 2.06013000000000e+03 blithely express theodolites wake furi-
ously within the b
       122 Supplier#000000122        2RUSHHspScCVTWC6z vw2XVR
16 26-432-258-4986 2.73295000000000e+03 fluffily silent asymptotes
along the accounts ha
       125 Supplier#000000125        XG
eO4Xb4TSF7rj4R6WRQ1v2seTlyga3tvFZaC               2 12-419-430-
3983 5.15725000000000e+03 quickly final excuses wake furiously
against the furiously ex
       126 Supplier#000000126        CaO4YuZ oSkzemn
14 24-728-670-3468 6.82986000000000e+03 carefully pending pack-
ages sleep slyly.
       144 Supplier#000000144        f8tddEKps816HHqNwsKdn3
20 30-726-423-7363 9.80629000000000e+03 even, fluffy somas ca-
jole ironically. even instructions are after the bold deposits. silent ac
       169 Supplier#000000169        ycymrfB5JV1vU,swPXggAt
13 23-698-509-1073 -9.27500000000000e+02 even, ironic packages
boost regularly at the carefully unusual inst

  10 record(s) selected.


SELECT * FROM TPCD.PARTSUPP FETCH FIRST 10 ROWS ONLY

PS_PARTKEY  PS_SUPPKEY  PS_AVAILQTY PS_SUPPLYCOST
PS_COMMENT
----------- ----------- ----------- ------------------------- ---------------------------------
----------------------------------------------------------------------------------------------
----------------------------------------------------------------------
         43          44        3211 8.05780000000000e+02 final, express
dependencies sleep according to the express requests. bold, regular
accounts detect outside the slyly
         43      750044        6770 4.93190000000000e+02 furiously spe-
cial pinto beans cajole. ironic decoys across the
```

```
        43      1500044         9506 4.93650000000000e+02 carefully
fluffy accounts across the blithely final accounts hang slyly according
to the furiously special platelets. sil
        43      2250044         3232 3.07120000000000e+02 bold pack-
ages wake blithely above the furiously bold
        47      48              6989 2.92520000000000e+02 slyly regular
foxes are furiously. ironic accounts engage quickly ironic, close pinto
beans. blithely ironic packages wake blithely ironic epitaphs. carefully
even
        47      750048          4458 5.39470000000000e+02 carefully
ironic asymptotes against the permanently bold deposits boost furi-
ously ruthless theodolites. carefully final requests are fluffily regula
        47      1500048         2896 7.45400000000000e+01 ironic re-
quests across the always ironic pinto beans haggle furiously at the
even frets. regular foxes along the unusual, final instructions sleep
against the carefully fi
        47      2250048         5873 2.96630000000000e+02 express ac-
counts haggle carefully. fluffily pending packages alongside of the fur
        89      90              3430 7.44870000000000e+02 boldly silent
theodolites grow carefully furiously special platelets. packages cajole
fluffily foxes. slyly special ideas against the slyly unusual packages
        89      750090          8599 7.76530000000000e+02 blithely even
deposits integrate blithely about the furiously regular pains. special ac-
coun

  10 record(s) selected.


SELECT * FROM TPCD.CUSTOMER FETCH FIRST 10 ROWS
ONLY

C_CUSTKEY   C_NAME                  C_ADDRESS
C_NATIONKEY C_PHONE         C_ACCTBAL
C_MKTSEGMENT C_COMMENT
----------- ------------------------ ---------------------------------------- ----------- -----
--------- ------------------------- ------------ ----------------------------------------------
-------------------------------------------------------------------------
        26 Customer#000000026       8ljrc5ZeMl7UciP
22 32-363-455-4837 5.18205000000000e+03 AUTOMOBILE   bold, fi-
nal ideas sleep carefully never express foxes.
        30 Customer#000000030       nJDsEL-
GAavU63Jl0c5NKsKfL8rIJQQkQnYL2QJY           1 11-764-165-5076
9.32101000000000e+03 BUILDING     regular deposits hinder slyly
ironic foxes
        57 Customer#000000057       97XYbsuOPRXPWU
21 31-835-306-1650 4.15193000000000e+03 AUTOMOBILE   ac-
counts are. quickly regular pinto beans subla
        88 Customer#000000088
wtkjBN9eyrFuENSMmMFlJ3e7jE5KXcg             16 26-516-273-
2566 8.03144000000000e+03 AUTOMOBILE   ironic packages ought
to sleep packages. enticingly e
        94 Customer#000000094       IfV-
NIN9KtkScJ9dUjK3Pg5gY1aFeaXewwf             9 19-953-499-8833
5.50011000000000e+03 HOUSEHOLD    furiously even requests nag
slyly ironic accounts. blithely bold packages across the final,
       100 Customer#000000100       fptUABXcmkC5Wx
20 30-749-445-4907 9.88989000000000e+03 FURNITURE    final re-
quests solve slyly regular instructions. slyly regular excuses solve
above the carefully express accounts
       107 Customer#000000107
Zwg64UZ,q7GRqo3zm7P1tZIRshBDz               15 25-336-529-
9919 2.51415000000000e+03 AUTOMOBILE   packages haggle ac-
cording to the dependencies. blithely even deposits shall nag? slyly
regular courts nag bli
       109 Customer#000000109       OOOkYBgCMzgMQXUmko-
coLb56rfrdWp2NE2c           16 26-992-422-8153 -
7.16100000000000e+02 BUILDING     quickly bold asymptotes haggle
dolphins. fu
       123 Customer#000000123       YsOnaaER8MkvK5cpf4VSlq
5 15-817-151-1168 5.89783000000000e+03 BUILDING     slyly regular
accounts use blithely. silent instructions haggle blithely silent pack-
ages. ironic ac
       127 Customer#000000127
Xyge4DX2rXKxXyye1Z47LeLVEYMLf4Bfcj          21 31-101-672-
2951 9.28071000000000e+03 MACHINERY    slyly pending deposits
sleep according to the furiously regular packages. slyly ironic pack-
ages s

  10 record(s) selected.


SELECT * FROM TPCD.ORDERS FETCH FIRST 10 ROWS ONLY

O_ORDERKEY  O_CUSTKEY   O_ORDERSTATUS O_TOTALPRICE
O_ORDERDATE O_ORDERPRIORITY O_CLERK
O_SHIPPRIORITY O_COMMENT
----------- ----------- ------------ ------------------------ ----------- --------------- -----
--------- -------------- ----------------------------------------------------------------------
----------
      27137     16232609 F            2.46557720000000e+05 01/01/1992
2-HIGH          Clerk#000264258               0 ironic, pending excuses
kindle along the
     179335     24706957 F            7.25575100000000e+04
01/01/1992  3-MEDIUM        Clerk#000051659               0 ironic re-
quests across the requests doze furiously ironic instruction
     436836     18970130 F            1.64781850000000e+05
01/01/1992  3-MEDIUM        Clerk#000185968               0 slyly slow re-
quests around the slyly silent deposi
     494854      8386792 F            1.92162030000000e+05 01/01/1992
2-HIGH          Clerk#000243771               0 regular requests are blithely
bold, even frets.
     862278      7837630 F            1.80929630000000e+05 01/01/1992
1-URGENT        Clerk#000107382               0 fluffily ironic requests a
     902276      6452023 F            7.64533200000000e+04 01/01/1992
5-LOW           Clerk#000199580               0 final courts cajole furiously
express, ironic frays. express accounts wake.
    1211207     18698104 F            1.55174650000000e+05
01/01/1992  4-NOT SPECIFIED Clerk#000145531               0 final re-
quests among the regular, even excuses haggle furiously pendin
    1373925      1748192 F            3.19194110000000e+05
01/01/1992  1-URGENT        Clerk#000091974               0 final pinto
beans cajole ruthlessly regular excuses. even, ironic platelet
    1508679     23314243 F            2.94180390000000e+05
01/01/1992  4-NOT SPECIFIED Clerk#000233632               0 carefully
fluffy theodolites against the slyly bold hock
    1656837     18999214 F            2.22567040000000e+05
01/01/1992  3-MEDIUM        Clerk#000063027               0 even, pend-
ing packages nag according to th

  10 record(s) selected.


SELECT * FROM TPCD.LINEITEM FETCH FIRST 10 ROWS ONLY

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
----------- ----------- ----------- ----------- ------------------------ ----------------------
-- ------------------------ ------------------------ ------------ ------------ ---------- -----
------ -------------- -------------------------- ---------- ----------------------------------------
------
    1054181     4865078     1865079           1
4.50000000000000e+01 4.69273500000000e+04
3.00000000000000e-02 8.00000000000000e-02 R          F
01/02/1992 02/05/1992   01/15/1992    NONE                 MAIL
even instructions kindle furio
    5018977     24611600     611601           1
2.00000000000000e+01 3.02074000000000e+04
```

```
0.00000000000000e+00 0.00000000000000e+00 A          F
01/02/1992 03/19/1992   01/15/1992    NONE                 SHIP
quickly ironic excu
   19547653     37197566     1947579          1
4.80000000000000e+01 7.97620800000000e+04
0.00000000000000e+00 7.00000000000000e-02 R          F
01/02/1992 02/08/1992   01/26/1992    NONE                 MAIL
regular accounts haggle. h
   19918976     6334057     2584064          6
3.10000000000000e+01 3.38129400000000e+04
5.00000000000000e-02 6.00000000000000e-02 R          F
01/02/1992 02/23/1992   01/03/1992    DELIVER IN PERSON
REG AIR    carefully fluffy requests
   27011394     1996025     2746026          4
4.20000000000000e+01 4.70790600000000e+04
4.00000000000000e-02 5.00000000000000e-02 A          F
01/02/1992 03/21/1992   01/20/1992    NONE                 REG AIR
carefully express packa
   35459264     33148881     2398915          2
1.30000000000000e+01 2.50669900000000e+04
1.00000000000000e-01 1.00000000000000e-02 A          F
01/02/1992 02/02/1992   01/07/1992    COLLECT COD          AIR
even packages abo
   35584422     24142067     2392092          2
2.00000000000000e+00 2.21572000000000e+03
0.00000000000000e+00 1.00000000000000e-02 R          F
01/02/1992 03/03/1992   01/28/1992    NONE                 SHIP
furiously dogged s
   37766533     7675859     1675860          5
1.60000000000000e+01 2.93515200000000e+04
4.00000000000000e-02 2.00000000000000e-02 A          F
01/02/1992 02/23/1992   01/07/1992    COLLECT COD          FOB
deposits sleep among the quickl
   45828802     3935061     935062          7
4.50000000000000e+01 4.93141500000000e+04
3.00000000000000e-02 4.00000000000000e-02 A          F
01/02/1992 03/29/1992   01/14/1992    TAKE BACK RETURN
FOB        carefully special fo
   47726080     17850144     2850145          1
2.40000000000000e+01 2.62380000000000e+04
2.00000000000000e-02 2.00000000000000e-02 A          F
01/02/1992 02/23/1992   01/16/1992    DELIVER IN PERSON
RAIL       ironic excuses ab

 10 record(s) selected.
```

C.24 Query Substitution Parameters

```
Power stream    Seed = 1206223349

-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 1206223349 as a seed to the RNG
Q1  DELTA       80
Q2  SIZE        14
   TYPE        BRASS
   REGION      AMERICA
Q3  SEGMENT     AUTOMOBILE
   DATE        1995-03-11
Q4  DATE        1994-10-01
Q5  REGION      AFRICA
   DATE        1995-01-01
Q6  DATE        1995-01-01
   DISCOUNT    0.04
   QUANTITY    25
Q7  NATION1     GERMANY
   NATION2     MOROCCO
Q8  NATION      MOROCCO
   REGION      AFRICA
   TYPE        SMALL BURNISHED COPPER
Q9  COLOR       cyan
Q10 DATE        1993-08-01
Q11 NATION      VIETNAM
   FRACTION    0.0000003333
Q12 SHIPMODE1   MAIL
   SHIPMODE2   SHIP
   DATE        1995-01-01
Q13 WORD1       express
   WORD2       requests
Q14 DATE        1995-05-01
Q15 DATE        1993-11-01
Q16 BRAND       Brand#45
   TYPE        SMALL BRUSHED
   SIZE1       24
   SIZE2       13
   SIZE3       6
   SIZE4       18
   SIZE5       38
   SIZE6       11
   SIZE7       40
   SIZE8       8
Q17 BRAND       Brand#52
   CONTAINER   MED DRUM
Q18 QUANTITY    313
Q19 BRAND1      Brand#32
   BRAND2      Brand#14
   BRAND3      Brand#43
   QUANTITY1   8
   QUANTITY2   15
   QUANTITY3   27
Q20 COLOUR      dodger
   DATE        1993-01-01
   NATION      MOROCCO
Q21 NATION      BRAZIL
Q22 I1          12
   I2          24
   I3          17
   I4          28
   I5          32
   I6          26
   I7          31

Throughput Stream = 1    Seed = 1206223350
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 1206223350 as a seed to the RNG
Q1  DELTA       88
Q2  SIZE        1
   TYPE        TIN
   REGION      MIDDLE EAST
Q3  SEGMENT     HOUSEHOLD
   DATE        1995-03-28
Q4  DATE        1997-05-01
Q5  REGION      AMERICA
   DATE        1995-01-01
Q6  DATE        1995-01-01
   DISCOUNT    0.09
   QUANTITY    24
Q7  NATION1     UNITED STATES
   NATION2     GERMANY
Q8  NATION      GERMANY
   REGION      EUROPE
   TYPE        STANDARD BRUSHED COPPER
Q9  COLOR       chiffon
Q10 DATE        1994-05-01
Q11 NATION      INDONESIA
   FRACTION    0.0000003333
Q12 SHIPMODE1   TRUCK
```

```
   SHIPMODE2    SHIP
   DATE         1995-01-01
Q13 WORD1       express
   WORD2        accounts
Q14 DATE        1995-09-01
Q15 DATE        1996-06-01
Q16 BRAND       Brand#35
   TYPE         ECONOMY ANODIZED
   SIZE1        25
   SIZE2        35
   SIZE3        45
   SIZE4        4
   SIZE5        48
   SIZE6        39
   SIZE7        16
   SIZE8        20
Q17 BRAND       Brand#54
   CONTAINER    JUMBO BAG
Q18 QUANTITY    314
Q19 BRAND1      Brand#44
   BRAND2       Brand#42
   BRAND3       Brand#42
   QUANTITY1    4
   QUANTITY2    16
   QUANTITY3    23
Q20 COLOUR      peru
   DATE         1996-01-01
   NATION       ETHIOPIA
Q21 NATION      ROMANIA
Q22 I1          31
   I2           25
   I3           18
   I4           26
   I5           21
   I6           33
   I7           30

Throughput Stream = 2    Seed = 1206223351
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 1206223351 as a seed to the RNG
Q1  DELTA       96
Q2  SIZE        39
   TYPE         COPPER
   REGION       ASIA
Q3  SEGMENT     AUTOMOBILE
   DATE         1995-03-13
Q4  DATE        1995-02-01
Q5  REGION      ASIA
   DATE         1996-01-01
Q6  DATE        1996-01-01
   DISCOUNT     0.07
   QUANTITY     25
Q7  NATION1     MOZAMBIQUE
   NATION2      UNITED STATES
Q8  NATION      UNITED STATES
   REGION       AMERICA
   TYPE         STANDARD PLATED COPPER
Q9  COLOR       blue
Q10 DATE        1993-02-01
Q11 NATION      RUSSIA
   FRACTION     0.0000003333
Q12 SHIPMODE1   RAIL
   SHIPMODE2    REG AIR
   DATE         1995-01-01
Q13 WORD1       special
   WORD2        accounts
Q14 DATE        1995-12-01
Q15 DATE        1994-03-01
Q16 BRAND       Brand#15
   TYPE         STANDARD PLATED
   SIZE1        14
   SIZE2        6
   SIZE3        38
   SIZE4        41
   SIZE5        19
   SIZE6        9
   SIZE7        4
   SIZE8        20
Q17 BRAND       Brand#51
   CONTAINER    JUMBO PKG

Q18 QUANTITY    312
Q19 BRAND1      Brand#41
   BRAND2       Brand#35
   BRAND3       Brand#31
   QUANTITY1    9
   QUANTITY2    17
   QUANTITY3    30
Q20 COLOUR      blush
   DATE         1995-01-01
   NATION       SAUDI ARABIA
Q21 NATION      IRAQ
Q22 I1          30
   I2           24
   I3           18
   I4           26
   I5           27
   I6           16
   I7           15

Throughput Stream = 3    Seed = 1206223352
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 1206223352 as a seed to the RNG
Q1  DELTA       104
Q2  SIZE        27
   TYPE         STEEL
   REGION       MIDDLE EAST
Q3  SEGMENT     FURNITURE
   DATE         1995-03-30
Q4  DATE        1997-08-01
Q5  REGION      EUROPE

   DATE         1996-01-01
Q6  DATE        1996-01-01
   DISCOUNT     0.04
   QUANTITY     25
Q7  NATION1     INDIA
   NATION2      MOZAMBIQUE
Q8  NATION      MOZAMBIQUE
   REGION       AFRICA
   TYPE         STANDARD ANODIZED COPPER
Q9  COLOR       aquamarine
Q10 DATE        1993-11-01
Q11 NATION      INDONESIA
   FRACTION     0.0000003333
Q12 SHIPMODE1   AIR
   SHIPMODE2    REG AIR
   DATE         1996-01-01
Q13 WORD1       special
   WORD2        accounts
Q14 DATE        1996-03-01
Q15 DATE        1996-10-01
Q16 BRAND       Brand#45
   TYPE         MEDIUM POLISHED
   SIZE1        21
   SIZE2        8
   SIZE3        9
   SIZE4        20
   SIZE5        25
   SIZE6        27
```

```
   SIZE7       26
   SIZE8       1
Q17 BRAND      Brand#53
   CONTAINER   JUMBO DRUM
Q18 QUANTITY   313
Q19 BRAND1     Brand#43
   BRAND2      Brand#12
   BRAND3      Brand#31
   QUANTITY1   4
   QUANTITY2   18
   QUANTITY3   27
Q20 COLOUR     maroon
   DATE        1993-01-01
   NATION      INDONESIA
Q21 NATION     EGYPT
Q22 I1         27
   I2          22
   I3          19
   I4          32
   I5          11
   I6          17
   I7          15

Throughput Stream = 4   Seed = 1206223353
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 1206223353 as a seed to the RNG
Q1  DELTA      112
Q2  SIZE       15
   TYPE        BRASS
   REGION      ASIA
Q3  SEGMENT    MACHINERY
   DATE        1995-03-15
Q4  DATE       1995-05-01
Q5  REGION     MIDDLE EAST
   DATE        1996-01-01
Q6  DATE       1996-01-01
   DISCOUNT    0.02
   QUANTITY    24
Q7  NATION1    ALGERIA
   NATION2     INDIA
Q8  NATION     INDIA
   REGION      ASIA
   TYPE        PROMO BRUSHED TIN
Q9  COLOR      violet
Q10 DATE       1994-09-01
Q11 NATION     RUSSIA
   FRACTION    0.0000003333
Q12 SHIPMODE1  REG AIR
   SHIPMODE2   MAIL
   DATE        1994-01-01
Q13 WORD1      special
   WORD2       accounts
Q14 DATE       1996-06-01
Q15 DATE       1994-06-01
Q16 BRAND      Brand#35
   TYPE        PROMO ANODIZED
   SIZE1       8
   SIZE2       38
   SIZE3       29
   SIZE4       13
   SIZE5       22
   SIZE6       1
   SIZE7       20
   SIZE8       7
Q17 BRAND      Brand#55
   CONTAINER   WRAP BAG
Q18 QUANTITY   315
Q19 BRAND1     Brand#55
   BRAND2      Brand#55
   BRAND3      Brand#35
   QUANTITY1   9
   QUANTITY2   19
   QUANTITY3   23
Q20 COLOUR     tomato
   DATE        1997-01-01
   NATION      UNITED STATES
Q21 NATION     VIETNAM
Q22 I1         11
   I2          19
   I3          31
   I4          26
   I5          13
   I6          17
   I7          18

Throughput Stream = 5   Seed = 1206223354
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 1206223354 as a seed to the RNG
Q1  DELTA      120
Q2  SIZE       2
   TYPE        NICKEL
   REGION      AFRICA
Q3  SEGMENT    FURNITURE
   DATE        1995-03-01
Q4  DATE       1993-02-01
Q5  REGION     AFRICA
   DATE        1996-01-01
Q6  DATE       1996-01-01
   DISCOUNT    0.07
   QUANTITY    24
Q7  NATION1    PERU
   NATION2     ALGERIA
Q8  NATION     ALGERIA
   REGION      AFRICA
   TYPE        PROMO PLATED TIN
Q9  COLOR      spring
Q10 DATE       1993-06-01
Q11 NATION     IRAN
   FRACTION    0.0000003333
Q12 SHIPMODE1  FOB
   SHIPMODE2   REG AIR
   DATE        1996-01-01
Q13 WORD1      special
   WORD2       accounts
Q14 DATE       1996-10-01
Q15 DATE       1997-01-01
Q16 BRAND      Brand#15
   TYPE        SMALL BURNISHED
   SIZE1       47
   SIZE2       44
   SIZE3       27
   SIZE4       29
   SIZE5       2
   SIZE6       20
   SIZE7       21
   SIZE8       35
Q17 BRAND      Brand#52
   CONTAINER   WRAP PKG
Q18 QUANTITY   312
Q19 BRAND1     Brand#52
   BRAND2      Brand#33
   BRAND3      Brand#24
   QUANTITY1   5
   QUANTITY2   20
   QUANTITY3   30
Q20 COLOUR     goldenrod
   DATE        1995-01-01
   NATION      KENYA
Q21 NATION     JORDAN
Q22 I1         11
```

```
   I2         18
   I3         16
   I4         12
   I5         14
   I6         10
   I7         22

Throughput Stream = 6    Seed = 1206223355
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 1206223355 as a seed to the RNG
Q1  DELTA       67
Q2  SIZE        40
   TYPE        COPPER
   REGION      ASIA
Q3  SEGMENT     MACHINERY
   DATE        1995-03-17
Q4  DATE        1995-09-01
Q5  REGION      AMERICA
   DATE        1997-01-01
Q6  DATE        1997-01-01
   DISCOUNT    0.04
   QUANTITY    25
Q7  NATION1     INDONESIA
   NATION2     PERU
Q8  NATION      PERU
   REGION      AMERICA
   TYPE        PROMO ANODIZED TIN
Q9  COLOR       seashell
Q10 DATE        1994-03-01
Q11 NATION      UNITED KINGDOM
   FRACTION    0.0000003333

Q12 SHIPMODE1   MAIL
   SHIPMODE2   AIR
   DATE        1996-01-01
Q13 WORD1       special
   WORD2       deposits
Q14 DATE        1997-01-01
Q15 DATE        1994-10-01
Q16 BRAND       Brand#45
   TYPE        LARGE POLISHED
   SIZE1       38
   SIZE2       39
   SIZE3       27
   SIZE4       8
   SIZE5       26
   SIZE6       21
   SIZE7       34
   SIZE8       29
Q17 BRAND       Brand#54
   CONTAINER   WRAP DRUM
Q18 QUANTITY    314
Q19 BRAND1      Brand#54
   BRAND2      Brand#21
   BRAND3      Brand#23
   QUANTITY1   10
   QUANTITY2   10
   QUANTITY3   26
Q20 COLOUR      rosy
   DATE        1994-01-01
   NATION      EGYPT
Q21 NATION      ETHIOPIA
Q22 I1          14
   I2         10
   I3         16
   I4         30
   I5         17
   I6         25
   I7         11
```

Appendix - D
Driver Source Code

D.1 tpcdbatch.h

```
/** Necessary header files **/

/** System header files **/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <fcntl.h>              /* SUN bbe */

#include <time.h>
#include <ctype.h>

#if (defined(SQLAIX) || defined(SQLPTX) || defined(LINUX) || de-
fined(SQLHP))
#include <unistd.h>             /* SUN */
#include <sys/stat.h>           /* SUN */
#endif
#if ((defined(SQLAIX) || defined(SQLPTX)) && !defined(LINUX))
#include <sys/vnode.h>          /* SUN */
#endif
#ifndef SQLWINT
#include <sys/time.h>                           /*@d33143aha*/
#include <sys/ipc.h>
#include <sys/sem.h>
#if (!defined(SQLPTX) && !defined(LINUX)&& !defined(SQLHP))
#include <sys/mode.h>
#endif
#include <sys/timeb.h>
#include <sys/types.h>
#else
#include <windows.h>
#include <sys\timeb.h>
#endif
#include <errno.h>


/** External header files **/
#include "sqlda.h"
#include "sqlenv.h"
#include "sql.h"
#include "sqlmon.h"
#include "sqlca.h"
#include "sqlutil.h"
#include "sqlcodes.h"

/** Internal header files **/
/** #ifdef __cplusplus **/
/** #include "sqlz.h" **/
/** #include "sqlzcopy.h" **/
/** #endif **/


/*****************************************************************/
/* Define synonyms here                                          */
/*****************************************************************/
#define TPCDBATCH_VERSION "5.6"

#define TPCDBATCH_NONSQL      10                        /* @d23684
tjg */
#define TPCDBATCH_SELECT      20
#define TPCDBATCH_NONSELECT   30
#define TPCDBATCH_EOBLOCK     40                        /*
@d30369 tjg */
#define TPCDBATCH_INSERT      50
#define TPCDBATCH_DELETE      60

#define TPCDBATCH_MAX_COLS    100                       /*
@d30369 tjg */

#define TPCDBATCH_CHAR char

#define TPCDBATCH_PRINT_FLOAT_WIDTH  20
  /* kmw - allow 15 whole digit for %#.3f format */
  /*     - note: use > 18, size of long indentifier so that it will */
  /*       be larger than any column heading                        */
#define TPCDBATCH_PRINT_FLOAT_MAX    1e15  /* kmw */
/* #define TPCD_PREPARETIME 1 */      /* for separate prep/exec on
uf jen 1106 */

#ifdef SQLWINT
#define PATH_DELIM '\\'
#define sleep(a) Sleep((a)*1000)
#else
#define PATH_DELIM '/'
#endif

#define PARALLEL_UPDATES 1

#ifdef PARALLEL_UPDATES
#define UF1OUTSTREAMPATTERN "%s%cuf1.%02d.%d.out"
#ifdef TPCD_NONPARTITIONED
#define UF2OUTSTREAMPATTERN "%s%cuf2.%02d.%d.out"
#else
/* kelly add same as NONPART. */
#define UF2OUTSTREAMPATTERN "%s%cuf2.%02d.%d.out"
/*  kelly ... take this out ... should be same name as for non-paritioned
#define UF2OUTSTREAMPATTERN "%s%cuf2.%02d.%d.%d.out"   */
/*DELjen add delchunk*/
#endif
#define BUFSIZE 1024
#endif


#define T_STAMP_FORM_1 1
#define T_STAMP_FORM_2 2
/* jen TIME_ACC start */
#define T_STAMP_FORM_3 3
#define T_STAMP_1LEN 17
#if defined (SQLUNIX) || defined (SQLAIX)
#define T_STAMP_3LEN 24
#elif (defined (SQLOS2) || defined(SQLWINT) || defined (SQLWIN) ||
defined(SQLDOS))
#define T_STAMP_3LEN 21    /* WIN NT timestamp fix bbe */
#else
#error Unknown operating system
#endif
/* jen TIME_ACC start */


#define BLANKS      "\0"
#define READMODE    "r\0"
#define WRITEMODE   "w\0"
#define APPENDMODE  "a\0"
#define mem_error(xx)                                          \
{ fprintf(stderr,"\n--Out of memory when %s.\n",xx); }
/* Display out-of-memory and end */

#define     TPCDBATCH_MIN(x,y)        ((x) < (y) ? (x) : (y))
/** Returns the smaller of both x and y **/
#define     TPCDBATCH_MAX(x,y)        ((x) > (y) ? (x) : (y))  /*
@d22817 tjg */
```

```
/** Returns the larger of both x and y **/



/** Defines needed for decimal conversion **/
#define SQLZ_DYNLINK
#define TRUE 1
#define LEFT 1
#define RIGHT 0
#define FALSE 0
#define sqlrx_get_left_nibble(byte) (((unsigned char)(byte)) >> 4)

#define sqlrx_get_right_nibble(byte) ((unsigned char) (byte & '\x0f'))
#define SQL_MAXDECIMAL        31
#define  SQLRX_PREFERRED_PLUS  0x0c

/** Timer-necessary defines for portability **/
#if (defined (SQLOS2) || defined(SQLWINT)) || defined(SQLWIN) || de-
fined(SQLDOS)
typedef struct timeb Timer_struct;
#elif (defined (SQLUNIX) || defined(SQLAIX))                /*TIMER jen*/
typedef struct timeval Timer_struct;
#else
#error Unknown operating system
#endif

/* sleep time between starting subsequent tpcdbatches running UF1
and UF2 */
#define UF1_SLEEP 1
#define UF2_SLEEP 1
#define UF_DEADLOCK_SLEEP 1 /* sleep between deadlock retries
in UF1,UF2 */

#define MAXWAIT 50 /* maximum retries for deadlock encounters */

#define DEBUG 0    /* to be set to 1 for diagnostic purposes if needed
*/
/* #define UF1DEBUG 1   */
/* #define UF2DEBUG 1   */
```

D.2 tpcdbatch.sqc

```
/* included in tpcdbatch.sqc and tpcdUF.sqc */

#include "tpcdbatch.h"

/*****************************************************************************/
/* global structure containing elements passed between different func-
tions */
/*****************************************************************************/
struct global_struct
{
  struct stmt_info     *s_info_ptr;          /* ptr to stmt_info list       */
  struct stmt_info     *s_info_stop_ptr;     /* ptr to last struct in list  */
  struct comm_line_opt *c_l_opt;             /* ptr to comm_line_opt
struct */
  struct ctrl_flags    *c_flags;             /* ptr to ctrl_flags struct    */
  Timer_struct         stream_start_time;    /* start time for stream
TIME_ACC */
  Timer_struct         stream_end_time;      /* end time for stream
TIME_ACC  */
  char                 file_time_stamp[50];  /* time stamp for output files */
  double               scale_factor;         /* scale factor of database    */
  char                 run_dir[150];         /* directory for output files  */
  int                  copy_on_load;         /* indication of whether or not */
                                             /* to do use a copy directory   */
                                             /* (equiv to COPY YES) on load  */
                                             /* default is FALSE */
  long                 lSeed;                /* seed used to generate the    */
                                             /* queries for this particular  */
                                             /* run.                         */
  FILE                 *stream_list;         /* ptr to query list file      */
  char                 update_num_file[150];  /* name of file that keeps
track */
                                             /* of which update pairs have run*/
  char                 sem_file[150];        /* semaphore name */
  char                 sem_file2[150];       /* semaphore name bbe */
  FILE                 *stream_report_file;  /* file to report start stop */
                                             /* progress of the stream */
};


/*****************************************************************************/
/* New type declaration to store details about SQL statement          */
/*****************************************************************************/

struct stmt_info
{
  long                 max_rows_fetch;
  long                 max_rows_out;
  int                  query_block;                    /* @d30369 tjg */
  unsigned int         stmt_num;                        /* @d24993 tjg */
  double               elapse_time;                     /* @d24993 tjg */
  double               adjusted_time;
  char                 start_stamp[50];      /* start time stamp for block  */
  char                 end_stamp[50];        /* end time stamp for block    */
  char                 tag[50];              /* block tag                   */
  char                 qry_description[100];
  struct stmt_info     *next;                          /* @d24993 tjg */

};


/*****************************************************************************/
/* Structure containing command line options                          */
/*****************************************************************************/
struct comm_line_opt
{                                                 /* @d22275 tjg */

  char                 str_file_name[256];    /* output filename    */
  char                 infile[256];     /* input filename     */
  int                  intStreamNum;    /* integer version of stream number
*/
  int                  a_commit;        /* auto-commit flag   */
  int                  short_time;      /* time interval flag */
  int                  update;
  int                  outfile;
};


/*****************************************************************************/
/* Structure used to hold precision for decimal numbers               */
/*****************************************************************************/
struct declen
{/* kmw */
  unsigned char m;        /* # of digits left of decimal */
  unsigned char n;        /* # of digits right of decimal */
};



/*****************************************************************************/
/* Structure containing control flags passed between functions        */
/*****************************************************************************/
struct ctrl_flags
{                                                 /* @d25594 tjg */
  int eo_infile;
  int time_stamp;
```

```
   int eo_block;                                /* @d30369 tjg */
   int select_status;
};


/*********************************************************************/
/* Function Prototypes                                               */
/*********************************************************************/
int SleepSome( int amount );
int get_env_vars(void);
int Get_SQL_stmt(struct global_struct *g_struct);

void print_headings (struct sqlda *sqlda, int *col_lengths);  /* @d22817
tjg */
void echo_sqlda(struct sqlda *sqlda, int *col_lengths);
void allocate_sqlda(struct sqlda *sqlda);

void get_start_time(Timer_struct *start_time);
double get_elapsed_time (Timer_struct *start_time);

long error_check(void);                              /* @d28763 tjg */
void dumpCa(struct sqlca*);       /*kmw*/

void display_usage(void);
char *uppercase(char *string);
char *lowercase(char *string);
void comm_line_parse(int agrc, char *argv[], struct global_struct
*g_struct);
int sqlrxd2a(char *decptr,char *asciiptr,short prec,short scal);
void init_setup(int argc, char *argv[], struct global_struct *g_struct);
void runUF1( struct global_struct *g_struct, int updatePair );
void runUF2( struct global_struct *g_struct, int updatePair );

/* These need to be extern because they're in another SQC file.   aph
981205 */
/*extern void runUF1_fn( int updatePair, int i );*/              /* aph
981205 */
/*extern void runUF2_fn( int updatePair, int i, int numChunks );*/ /* aph
981205 */
/* Added four new arguments because SQL host vars can't be global.
aph 981205 */
extern void runUF1_fn ( int updatePair, int i, char *dbname, char
*userid, char *passwd );
extern void runUF2_fn ( int updatePair, int thisConcurrentDelete, int
numChunks, char *dbname, char *userid, char *passwd );

int sem_op (int semid, int semnum, int value);

char *get_time_stamp(int form, Timer_struct *timer_pointer);      /*
TIME_ACC jen */
void summary_table (struct global_struct *g_struct);
void free_sqlda (struct sqlda *sqlda, int select_status);     /* @d30369
tjg */
void output_file(struct global_struct *g_struct);
int PreSQLprocess(struct global_struct *g_struct, Timer_struct
*start_time);
void SQLprocess(struct global_struct *g_struct);
int PostSQLprocess(struct global_struct *g_struct, Timer_struct
*start_time);
int cleanup(struct global_struct *g_struct);

/* Semaphore control functions */
void create_semaphores(struct global_struct *g_struct);
void throughput_wait(struct global_struct *g_struct);
void runpower_wait(struct global_struct *g_struct, int sem_num);
void release_semaphore(struct global_struct *g_struct, int sem_num);
#ifdef SQLWINT
HANDLE open_semaphore(struct global_struct *g_struct, int num);
#else
int open_semaphore(struct global_struct *g_struct);
#endif


EXEC SQL INCLUDE SQLCA;

/*********************************************************************/
/* Declare the SQL host variables.                                   */
/*********************************************************************/
EXEC SQL BEGIN DECLARE SECTION;

char    stmt_str1[4000] = "\0";      /* Assume max SQL statment
                                    of 4000 char  */
struct {                            /* jen LONG */
       short len;
       char data[32700];
      } stmt_str;                   /* jen LONG */
char    dbname[9] = "\0";
char    userid[9] = "\0";
char    passwd[9] = "\0";
char    sourcefile[256];            /* used for semaphores and table func-
tions?*/
sqlint32 chunk = 0;                 /* jenCI counter for within the set of
chunks*/

EXEC SQL END DECLARE SECTION;


/*********************************************************************/
/* Declare the global variables.                                     */
/*********************************************************************/
struct sqlda   *sqlda;                  /* SQL Descriptor area  */

/* Global environment variables (sks May 25 98)*/
char env_tpcd_dbname[100];
char env_user[100];
char env_tpcd_audit_dir[150];
char env_tpcd_path_delim[2];
char env_tpcd_tmp_dir[150];
char env_tpcd_run_on_multiple_nodes[10];
char env_tpcd_copy_dir[150];
char env_tpcd_update_import[10];

/* Other globals */
FILE            *instream, *outstream;  /* File pointers             */
int             verbose = 0;            /* Verbose option flag       */
int             semcontrol = 1;         /* allows/disallows smaphores usage
*/
int             updatePairStart;        /* update pair to start at    */
int             currentUpdatePair;      /* update pair running        */
int             updatePairStop;         /* update pair to stop before */
char            newtime[50]="\0";       /* Des - moved from
get_time_stamp */
char            outstreamfilename[256]; /* store filename of outstream
                                        wlc 081397 */
int             inlistmax = 400;        /* define # of keys to delete at a time
                                        wlc 081897 */
int             sqlda_allocated = 0;    /* fixing free() problem in NT
                                        wlc 090597 */
int             iImportStagingTbl=0;    /* IMPORT use import or load (de-
fault) */
char            temp_time_stamp[50];    /* holds end timestamp to be
copied into start_time_stamp of next query bbeaton */
Timer_struct      temp_time_struct;       /* holds end time value to be
copied into start_time of next query bbeaton */

/* constants for the semaphores used; 1 for throughput and 2 for
power */
#define INSERT_POWER_SEM 1
#define QUERY_POWER_SEM 2
#define THROUGHPUT_SEM 1
```

```
/*****************************************************************/
/* Start main program processing.                                */
/*****************************************************************/
int main(int argc, char *argv[])
{

  struct comm_line_opt c_l_opt = { "\0","\0", 0, 1, 0, 0, 0 };
  /* command line options         */
  Timer_struct         start_time;          /* start point for elapsed time */


  struct stmt_info     s_info = { -1, -1, 0, 1, -1, -1, "\0", "\0", "\0", "\0",
NULL };
  /* first stmt_info structure */

  struct ctrl_flags    c_flags = { 0, 1, 0, TPCDBATCH_SELECT };
  /* structure holding ctrl flags
     passed between functions  */

  /* TIME_ACC jen start */
#if defined (SQLUNIX) || defined (SQLAIX)
  struct global_struct g_struct =
  { NULL, NULL, NULL, NULL, {0,0}, {0,0}, "\0", 0.1, "\0", FALSE, 0,
    NULL, "\0", "\0", "\0", NULL };

#elif (defined (SQLOS2) || defined(SQLWINT) || defined (SQLWIN) ||
defined(SQLDOS))
  struct global_struct g_struct =
  { NULL, NULL, NULL, NULL, {0,0,0,0}, {0,0,0,0}, "\0", 0.1, "\0",
FALSE, 0,
    NULL, "\0", "\0", "\0", NULL };
#else
#error Unknown operating system
#endif
  /* TIME_ACC jen end */


  /* Get environment variables */
  if (get_env_vars() != 0)
    return -1;


  /* perform setup and initialization and get process id of agent */
  outstream = stdout;
  g_struct.c_flags = &c_flags;

  g_struct.s_info_ptr = &s_info;
  g_struct.c_l_opt = &c_l_opt;

  init_setup(argc,argv,&g_struct);               /* @d22275 tjg */

  if ((g_struct.c_l_opt->update == 1) && (semcontrol == 1))
  /* runpower: wait for insert function to complete */
  /* waiting on the INSERT_POWER_SEM semaphore */
    runpower_wait(&g_struct, INSERT_POWER_SEM);

  strcpy(temp_time_stamp, "0");
  /*****************************************************************************
   *                                                                           *
   *   This is the transition from the "driver" to the "SUT"                   *
   *                                                                           *
*****************************************************************************/

  /*****************************************************************/
  /* Read in each statement, prepare, execute, and send output to file.
*/
  /*****************************************************************/
  while (!c_flags.eo_infile) {  /* Check to see if there's no more input */

    c_flags.eo_block = 0;

    if (c_l_opt.outfile)
      output_file(&g_struct); /* determine appropriate name for output
files */
    if ((g_struct.c_l_opt->update != 3) && (g_struct.c_l_opt->update !=
4))
    {
      if (!strcmp(temp_time_stamp, "0")) /* if first query, get timestamp */
      {
        get_start_time(&start_time);
        strcpy(g_struct.s_info_ptr->start_stamp,
               get_time_stamp(T_STAMP_FORM_3,&start_time ));  /*
TIME_ACC jen*/
      }
      else   /* else get the end timestamp of previous query */
      {
        strcpy(g_struct.s_info_ptr->start_stamp, temp_time_stamp);
        start_time = temp_time_struct;
      }
      /* write the start timestamp to the file...if this is not a qualification */
      /* run, then write the seed used as well */

      fprintf( outstream,"Start timestamp %*.*s \n",
               T_STAMP_3LEN,T_STAMP_3LEN,                 /* TIME_ACC
jen*/
               g_struct.s_info_ptr->start_stamp);
      if (c_l_opt.intStreamNum >= 0)
      {
        if (g_struct.lSeed == -1)
          {
            fprintf( outstream,"Using default qgen seed file");
          }
        else
          fprintf( outstream,"Seed used = %ld",g_struct.lSeed);

        fprintf( outstream,"\n");
      }
    }
    do {  /* Loop through these statements as long as we haven't
reached
             the end of the input file or the end of a block of statements
           */

      /** Read in the next statment **/
      c_flags.select_status=Get_SQL_stmt(&g_struct);

      if (PreSQLprocess(&g_struct, &start_time) == FALSE)
        /* if after reading the next statement we see that we should
           exit this loop (i.e. eof, update functions, etc...), get out
        */
        break;


/*****************************************************************************
      *                                                                       *
      *   The SQLprocess function implements the implementation spe-
cific layer.    *
      *   It can handle arbitrary SQL statements.                             *
      *                                                                       *

*****************************************************************************/

      /* If we've got up to here then processing
         a regular SQL statement */
      SQLprocess(&g_struct);
```

```
   } while ((!c_flags.eo_block) && (!c_flags.eo_infile));     /* @d30369
tjg */


   if (PostSQLprocess(&g_struct,&start_time) == FALSE)
     /* if we've reached the end of the input file, then get out
        of this loop (i.e. no more statements).  Otherwise get
        elapsed times and display info about rows */
     break;


 }  /* end of for loop for multiple SQL statements  */


 g_struct.s_info_ptr = &s_info;  /* set the global pointer to start of
                                    linked list */

 printf("Call cleanup\n");

 cleanup(&g_struct); /* finish some semaphore stuff, cleanup files,
                        and print out summary table */

 /*****************************************************************************
  *                                                                           *
  *   In cleanup we make the transition back from the "SUT" to the
"driver"    *
  *                                                                           *

*****************************************************************************/

 return(0);

}  /* end of main  */

/*****************************************************************************/
/* Generic form of Sleep */
int SleepSome( int amount)
{
#ifndef SQLWINT
  sleep (amount);
#else
  Sleep (amount*1000);          /* 10x for NT DJD Changed "sleep" to
"Sleep" */
#endif
  return 0;
}

/*****************************************************************************/

/*********************************************************************/
/* Get environment variables.  (sks May 25 98)                       */
/*********************************************************************/
int get_env_vars(void) {
 if (strcpy(env_tpcd_dbname, getenv("TPCD_DBNAME")) == NULL) {
    fprintf(stderr, "\n The environment variable $TPCD_DBNAME is
not setup correctly.\n");
    return -1;
  }
  if (strcpy(env_user, getenv("USER")) == NULL) {
    fprintf(stderr, "\n The environment variable $USER is not setup
correctly.\n");
    return -1;
  }
  if (strcpy(env_tpcd_audit_dir, getenv("TPCD_AUDIT_DIR")) ==
NULL) {
    fprintf(stderr, "\n The environment variable $TPCD_AUDIT_DIR is
not setup correctly.\n");
    return -1;
  }
  if (strcpy(env_tpcd_tmp_dir, getenv("TPCD_TMP_DIR")) == NULL) {
    fprintf(stderr, "\n The environment variable $TPCD_TMP_DIR is
not setup correctly.\n");
    return -1;
  }
#if 0
  if (strcpy(env_tpcd_path_delim, getenv("TPCD_PATH_DELIM")) ==
NULL ||
      (strcmp(env_tpcd_path_delim, "/") &&
strcmp(env_tpcd_path_delim, "\\"))){
    fprintf(stderr, "\n The environment variable $TPCD_PATH_DELIM
is not setup correctly , env_tpcd_path_delim'%s'.\n",
env_tpcd_path_delim);

    return -1;
  }
#endif
  strcpy( env_tpcd_path_delim , "/" );  /*kmw*/
  if (strcpy(env_tpcd_run_on_multiple_nodes,
getenv("TPCD_RUN_ON_MULTIPLE_NODES")) == NULL) {
    fprintf(stderr, "\n The environment variable
$TPCD_RUN_ON_MULTIPLE_NODES");
    fprintf(stderr, "\n is not setup correctly.\n");
    return -1;
  }
  if (strcpy(env_tpcd_copy_dir, getenv("TPCD_COPY_DIR")) ==
NULL) {
    fprintf(stderr, "\n The environment variable $TPCD_COPY_DIR is
not setup correctly.\n");
    return -1;
  }
  /* If TPCD_UPDATE_IMPORT is not set then, the default is set to
false, */
  /* which is done in init_setup subroutine                        */
  strcpy(env_tpcd_update_import,
getenv("TPCD_UPDATE_IMPORT"));

  return 0;
}

/*********************************************************************/
/* Get the SQL statement and any control statements from input.    */
/*********************************************************************/
int Get_SQL_stmt(struct global_struct *g_struct)

{
  char input_ln[256]    = "\0";    /* buffer for 1 line of text        */
  char temp_str[4000]   = "\0";    /* temp string for SQL stmt         */
  char control_str[256] = "\0";    /* control string                   */

  char *test_semi;              /* ptr to test for semicolon        */
  char *control_opt;            /* ptr used in control_str parsing  */
  char *select_status;           /* ptr to first word in query       */
  char *temp_ptr;               /* general purpose temp ptr         */

  int good_sql = 0;             /* good-sql stmt flag   @d23684 tjg */
  int stmt_num_flag = 1;          /* first line of SQL stmt flag      */
  int eostmt = 0;               /* flag to signal end of statement  */


  stmt_str.data[0]='\0';         /* Initialize statement buffer      */

  if (verbose)
    fprintf (stderr,"\n----------------------------------------------\n");
  fprintf (outstream,"\n----------------------------------------------\n");

  do {
    /** Read in lines from input one at a time **/
    fscanf(instream, "\n%[^\n]\n", input_ln);

    if (strstr(input_ln,"--") == input_ln) {    /* Skip all -- comments */
```

```
      if (strstr(input_ln,"--#SET") == input_ln) {
                                /* Store control string but
                                   keep going to find SQL stmt */
        strcpy(control_str,input_ln);
        if (verbose)
          fprintf(stderr,"%s\n", uppercase(control_str));
        fprintf(outstream,"%s\n", uppercase(control_str));

        /** Start parsing control str. and update appropriate vars. **/
        control_opt = strtok(control_str," ");
        while (control_opt != NULL) {
          if (strcmp(control_opt,"--#SET")) { /* Skip the #SET token */
            if (!strcmp(control_opt,"ROWS_FETCH"))
              g_struct->s_info_ptr->max_rows_fetch =
atoi(strtok(NULL," "));

            if (!strcmp(control_opt,"ROWS_OUT"))
              g_struct->s_info_ptr->max_rows_out = atoi(strtok(NULL,"
"));
          }

          control_opt = strtok(NULL," ");
        }
      }

      /* if the block option has been set, then check if we've
        reached the end of a block of statements */
      if (g_struct->s_info_ptr->query_block)                /* @d30369 tjg
*/
        if (strstr(input_ln,"--#EOBLK") == input_ln) {
          g_struct->c_flags->eo_block = 1;
          return TPCDBATCH_EOBLOCK;
        }
      if (strstr(input_ln, "-- Query") == input_ln)
        strcpy(g_struct->s_info_ptr->qry_description,input_ln);

      if (strstr(input_ln, "--#TAG") == input_ln)
        strcpy(g_struct->s_info_ptr->tag,(input_ln+sizeof("--#TAG")));

      /* if we're using update functions, return that info
        appropriately */
      if (g_struct->c_l_opt->update != 0) {
        if (strstr(input_ln, "--#INSERT") == input_ln)
          return TPCDBATCH_INSERT;

        if (strstr(input_ln, "--#DELETE") == input_ln)
          return TPCDBATCH_DELETE;
      }

      if (strstr(input_ln, "--#COMMENT") == input_ln) {    /* @d25594
tjg */
        temp_ptr = (input_ln + 11);   /* User-specified comments go to
                                   the outfile */
        if (verbose)
          fprintf (stderr,"%s\n",temp_ptr);
        fprintf (outstream,"%s\n",temp_ptr);
      }

      eostmt=0;
    }

    /* Need this hack here to check if there's any more empty lines left
      in the input file.  Continue only if there are aren't any */
    else if (strcmp(input_ln, "\0")) /* HACK */ {   /* A regular SQL state-
ment */
      if (stmt_num_flag) {  /* print this out only if it's the first line
                               of the SQL statement.  We only want this
                               line to appear once per statement */
        if (verbose)
          fprintf(stderr,"\n%s\n", g_struct->s_info_ptr-
>qry_description);
        fprintf(outstream,"\n%s\n", g_struct->s_info_ptr-
>qry_description);

        if (verbose)
          fprintf(stderr,"\nTag: %-5.5s  Stream: %d   Sequence num-
ber: %d\n",
                  g_struct->s_info_ptr->tag,g_struct->c_l_opt-
>intStreamNum,
                  g_struct->s_info_ptr->stmt_num);   /*jen0925*/
        fprintf(outstream,"\nTag: %-5.5s  Stream: %d   Sequence num-
ber: %d\n",
                g_struct->s_info_ptr->tag,g_struct->c_l_opt-
>intStreamNum,
                g_struct->s_info_ptr->stmt_num);      /*jen0925*/


        /* Turn off this flag once the number has been printed */
        stmt_num_flag = 0;

      }  /** Print out this heading the first time you encounter a
          non-comment statement **/

      /* Test to see if we've reached the end of a statement */
      good_sql = TRUE;                                  /* @d23684 tjg */
      test_semi = strstr (input_ln,";");
      if (test_semi == NULL) {   /* if there's no semi-colon keep on go-
ing */
        strcat (stmt_str.data,input_ln);                /* jen LONG */
        strcat (stmt_str.data," ");                     /* jen LONG */
        stmt_str.len = strlen( stmt_str.data );         /* jen LONG */
        eostmt = 0;
      }

      else {                    /* else replace the ; with a \0 and continue */
        *test_semi = '\0';
        strcat (stmt_str.data,input_ln);                /* jen LONG */
        stmt_str.len = strlen( stmt_str.data );         /* jen LONG */
        eostmt = 1;
      }

      fprintf(outstream, "\n%s", input_ln);
      if (verbose)
        fprintf(stderr,"\n%s", input_ln);
    }

    /** Test to see if we've reached the EOF.  Get out if that's the case
**/
    if (feof(instream)) {
      eostmt = TRUE;
      g_struct->c_flags->eo_infile = TRUE;                  /* @d22275
tjg */
    }

  }  while (!eostmt);

  fprintf(outstream, "\n");
  if (verbose)
    fprintf(stderr,"\n");

  /** erase the old control string **/
  strcpy(control_str,"\0");

  /** Determine whether statement is a SELECT or other SQL **/
  if (good_sql) {
    strcpy(temp_str,stmt_str.data);                     /* jen LONG */
    uppercase(temp_str);  /* Make sure that select is made to
SELECT */
    select_status=strtok(temp_str," ");
```

```
    if ( (stmt_str.data[0] == '(') || (!strcmp(select_status,"SELECT")) ||
         (!strcmp(select_status,"VALUES")) ||
         (!strcmp(select_status,"WITH")) )
      return TPCDBATCH_SELECT;
    else
      return TPCDBATCH_NONSELECT;
  }

  /** If you go through a file with just comments or control statments
    with no SQL, there's nothing to process...Exit TPCDBATCH **/

  else                                          /* @d23684 tjg */
    return TPCDBATCH_NONSQL;

} /*  Get_SQL_stmt */



/*********************************************************************/
/* allocate_sqlda -- This routine allocates space for the SQLDA.   */
/*********************************************************************/

void allocate_sqlda(struct sqlda *sqlda)
{
  int   loopvar;                        /* Loop counter */

  for (loopvar=0; loopvar<sqlda->sqld; loopvar++)
  {
    switch (sqlda->sqlvar[loopvar].sqltype)
    {
     case SQL_TYP_INTEGER:                      /* INTEGER */
     case SQL_TYP_NINTEGER:
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)malloc(sizeof(sqlint32))) ==
NULL)
         mem_error("allocating INTEGER");
       break;
     case SQL_TYP_BIGINT:                       /* BIGINT */
/*kmwBIGINT*/
     case SQL_TYP_NBIGINT:
/*#ifdef SQLWINT */
/*       if ((sqlda->sqlvar[loopvar].sqldata= */
/*              (TPCDBATCH_CHAR *)malloc(sizeof(__int64))) ==
NULL)*/
/* #else */
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)malloc(sizeof(sqlint64))) ==
NULL)
/* #endif*/
         mem_error("allocating BIGINT");
       break;
     case SQL_TYP_CHAR:                         /* CHAR */
     case SQL_TYP_NCHAR:
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)calloc(256,sizeof(char))) ==
NULL)
         mem_error("allocating CHAR/VARCHAR");
       break;
     case SQL_TYP_VARCHAR:                      /* VARCHAR */
     case SQL_TYP_NVARCHAR:
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)calloc(4002,sizeof(char))) ==
NULL)
         mem_error("allocating CHAR/VARCHAR");
       break;
     case SQL_TYP_LONG:                         /* LONG VARCHAR */
     case SQL_TYP_NLONG:
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)calloc(32702,sizeof(char))) ==
NULL)
         mem_error("allocating VARCHAR/LONG VARCHAR");
       break;
     case SQL_TYP_FLOAT:                        /* FLOAT */
     case SQL_TYP_NFLOAT:
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)malloc(sizeof(double))) == NULL)
         mem_error("allocating FLOAT");
       break;
     case SQL_TYP_SMALL:                        /* SMALLINT */
     case SQL_TYP_NSMALL:
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)malloc(sizeof(short))) == NULL)
         mem_error("allocating SMALLINT");
       break;
     case SQL_TYP_DECIMAL:                      /* DECIMAL */
     case SQL_TYP_NDECIMAL:
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)malloc(20)) == NULL)
         mem_error("allocating DECIMAL");
       break;
     case SQL_TYP_CSTR:                     /* VARCHAR (null termi-
nated) */
     case SQL_TYP_NCSTR:
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)calloc(4001,sizeof(char))) ==
NULL)
         mem_error("allocating CHAR/VARCHAR");
       break;
     case SQL_TYP_DATE:                         /* DATE */
     case SQL_TYP_NDATE:
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)calloc(13,sizeof(char))) == NULL)
         mem_error("allocating DATE");
       break;
     case SQL_TYP_TIME:                         /* TIME */
     case SQL_TYP_NTIME:
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)calloc(11,sizeof(char))) == NULL)
         mem_error("allocating TIME");
       break;
     case SQL_TYP_STAMP:                        /* TIMESTAMP */
     case SQL_TYP_NSTAMP:
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)calloc(29,sizeof(char))) == NULL)
         mem_error("allocating TIMESTAMP");
       break;
    }
    if ((sqlda->sqlvar[loopvar].sqlind=
           (short *)calloc(1,sizeof(short))) == NULL)
      mem_error("allocating indicator");

  }
  sqlda_allocated = 1; /* fix free() problem on NT
                          wlc 090597 */
  return;    /* allocate_sqlda  */
}


/*********************************************************************/
/* echo_sqlda -- This routine displays the contents of an SQLDA.
*/
/*********************************************************************/


void echo_sqlda(struct sqlda *sqlda, int *col_lengths)
{
  int    col;                    /* Column counter          */

  int    col_type;                /* Type of column          */
```

```
  char   temp_string[100] = "\0";        /* Temporary string           */
  char   decimal_string[100] = "\0";     /* String holding decimals    */
  char   *temp_ptr;

  TPCDBATCH_CHAR   m,n;                  /* precision and accuracy
                                            for decimal conversion     */


  for (col=0; col<sqlda->sqld; col++)    /* Loop through column count
*/
  {
    col_type=sqlda->sqlvar[col].sqltype;                  /* @d22817 tjg */

    if (*(sqlda->sqlvar[col].sqlind))                     /* @d30369 tjg */
      fprintf(outstream, "%* n/a  ",(col_lengths[col]-3));
    else
      switch (col_type)
      {
       case SQL_TYP_INTEGER:
       case SQL_TYP_NINTEGER:

         fprintf(outstream, "%*ld  ",col_lengths[col],
               *(sqlint32 *)(sqlda->sqlvar[col].sqldata));
         break;

       case SQL_TYP_BIGINT:  /*kmwBIGINT*/
       case SQL_TYP_NBIGINT:
/*#ifdef SQLWINT*/
/*           fprintf(outstream, "%*I64d ",col_lengths[col],*/
/*                   *(__int64 *)(sqlda->sqlvar[col].sqldata));*/
/*#else*/
         fprintf(outstream, "%*lld  ",col_lengths[col],
               *(sqlint64 *)(sqlda->sqlvar[col].sqldata));
/*#endif*/
         break;

       case SQL_TYP_CHAR:
       case SQL_TYP_NCHAR:

         fprintf(outstream, "%-*s  ",col_lengths[col],sqlda-
>sqlvar[col].sqldata);

         break;
       case SQL_TYP_VARCHAR:
       case SQL_TYP_NVARCHAR:
       case SQL_TYP_LONG:
       case SQL_TYP_NLONG:                                /* @d30369 tjg */
         ((struct sqlchar *)sqlda->sqlvar[col].sqldata)->
           data[((struct sqlchar *)sqlda->sqlvar[col].sqldata)->length] =
'\0';
         fprintf(outstream, "%-*s  ",
               col_lengths[col],
               ((struct sqlchar *)sqlda->sqlvar[col].sqldata)->data);
         break;
       case SQL_TYP_FLOAT:
       case SQL_TYP_NFLOAT:
       { /* kmw */
         if ( fabs(*(double *)(sqlda->sqlvar[col].sqldata))
                          < TPCDBATCH_PRINT_FLOAT_MAX  )
           fprintf(outstream, "%#*.3f  ",col_lengths[col],
                 *(double *)(sqlda->sqlvar[col].sqldata));
         else
           fprintf(outstream, "%*e  ",col_lengths[col],
                 *(double *)(sqlda->sqlvar[col].sqldata));
         break;
       }

       case SQL_TYP_SMALL:
       case SQL_TYP_NSMALL:
         fprintf(outstream, "%*hd  ",col_lengths[col],
               *(short *)(sqlda->sqlvar[col].sqldata));
         break;
       case SQL_TYP_DECIMAL:
       case SQL_TYP_NDECIMAL:

         m=(*(struct declen *)&sqlda->sqlvar[col].sqllen).m;
         n=(*(struct declen *)&sqlda->sqlvar[col].sqllen).n;
         if (sqlrxd2a((char *)sqlda->sqlvar[col].sqldata,temp_string,m,n)
!= 0)
         {
           fprintf(stderr, "\nThe decimal value could not be con-
verted.\n");
           exit (-1);
         }
         else {

           temp_ptr = temp_string;

           if (*temp_ptr == '-')
             strcpy(decimal_string, "-");

           else
             strcpy(decimal_string, " ");

           for (temp_ptr = temp_string + 1; *temp_ptr == '0';
temp_ptr++)
             ;

           strcat(decimal_string,temp_ptr);
           fprintf(outstream, "%*s  ",col_lengths[col],decimal_string);
         }

         break;

       case SQL_TYP_CSTR:
       case SQL_TYP_NCSTR:
       case SQL_TYP_DATE:
       case SQL_TYP_NDATE:
       case SQL_TYP_TIME:
       case SQL_TYP_NTIME:
       case SQL_TYP_STAMP:
       case SQL_TYP_NSTAMP:
         sqlda->sqlvar[col].sqldata[sqlda->sqlvar[col].sqllen+1]='\0';
         strcpy(temp_string,(char *)sqlda->sqlvar[col].sqldata);
         fprintf(outstream, "%-*s  ",(col_lengths[col]),temp_string);
         break;

       default:
         fprintf(stderr,"--Unknown column type (%d).  Abort-
ing.\n",col_type);
         break;
      }
  }

  fprintf(outstream, "\n");

  return;
}


/*********************************************************/
/* Calculate the elapsed time.                           */
/*********************************************************/

void get_start_time(Timer_struct *start_time)
{
  int rc = 0;
```

```
#if defined (SQLOS2) || defined (SQLWINT)  || defined (SQLWIN) ||
defined (SQLDOS)
  /*@d33143aha*/
  ftime (start_time);
#elif defined(SQLSNI)
  rc = gettimeofday(start_time);
#elif defined(SQLPTX)
  gettimeofday_mapped(start_time);
  rc = 0;       /* gettimeofday_mapped returns void */
#elif defined (SQLUNIX) || defined (SQLAIX)                    /*TIMER
jen*/
  rc = gettimeofday(start_time,NULL);
#else
#error Unknown operating system
#endif

   if (rc != 0) {
      fprintf(stderr,"Timer call failed, aborting test\nExiting
tpcdbatch..\n");
      exit(-1);
   }
}



/*****************************************************************/
/* Calculate and return the elapsed time given a starting time.       */
/*****************************************************************/
double get_elapsed_time ( Timer_struct *start_time)
{
   int                 status = 0;
   Timer_struct          end_time;
   double              result = -1.0;
#ifndef SQLWINT
   long int            result_sec;
   long int            result_usec;
#endif


#if defined(SQLSNI)
  status = gettimeofday(&end_time);
#elif defined(SQLPTX)
  gettimeofday_mapped(&end_time);
  status = 0;   /* gettimeofday_mapped returns void */
#elif defined (SQLUNIX) || defined (SQLAIX)
  status = gettimeofday(&end_time,NULL);                       /*TIMER jen*/
#elif defined (SQLOS2) || defined (SQLWINT) || defined (SQLWIN) ||
defined(SQLDOS)
  ftime(&end_time);
#else                          /** If another operating system **/
#error Unknown operating system
#endif

   if (status != 0)
      fprintf(stderr,"Bad return from gettimeofday, don't trust timer re-
sults...\n");

   else
   {
#if defined (SQLUNIX) || defined (SQLAIX)
      result_sec = end_time.tv_sec - start_time->tv_sec;
      result = (double) result_sec;
      /* TIMER used micro seconds with timeval (not nanoseconds) */
      if ((start_time->tv_usec > 0) && \
          (start_time->tv_usec < 1000000) && \
          (end_time.tv_usec > 0) && \
          (end_time.tv_usec < 1000000))
      {
         result_usec = end_time.tv_usec - start_time->tv_usec;
         result = (double) result_sec + ((double) result_usec/1000000);
      }
#elif (defined (SQLOS2) || defined(SQLWINT) || defined (SQLWIN) ||
defined(SQLDOS))
      result = (double) (end_time.time - start_time->time);
      result = result * 1000 + (end_time.millitm - start_time->millitm);
      result = result/1000;
#else
#error Unknown operating system
#endif

   }

   /*
    * translate the time to that rounded to the CLOSEST 0.1 seconds as
    * required by the TPC-D spec.   ROUNDING
    */
   /*   result = (double)(((long)((result + 0.099999) * 10))/10.0);*/
   result = (double)(((long)((result + 0.05) * 10))/10.0);
   return (result);
}


void dumpCa(struct sqlca *ca)
{
   int i;
   fprintf(outstream,"********************  DUMP OF SQLCA
********************\n");
   fprintf(outstream,"SQLCAID   : %.8s\n", ca->sqlcaid);
   fprintf(outstream,"SQLCABC   : %d\n", ca->sqlcabc);
   fprintf(outstream,"SQLCODE   : %d\n", ca->sqlcode);
   fprintf(outstream,"SQLERRML  : %d\n", ca->sqlerrml);
   fprintf(outstream,"SQLERRMC  : %.*s\n", ca->sqlerrml, ca-
>sqlerrmc);
   fprintf(outstream,"SQLERRP   : %.8s\n", ca->sqlerrp);

   for (i = 0; i < 6; i++)
   {
   fprintf(outstream,"SQLERRD[%d]: %d\n", i, ca->sqlerrd[i] );
   }
   fprintf(outstream,"SQLWARN   : %.11s\n", ca->sqlwarn);
   fprintf(outstream,"SQLSTATE  : %.5s\n", ca->sqlstate);
   fprintf(outstream,"******************  END OF SQLCA DUMP
******************\n");
   return;
}


/*****************************************************************************/
/* error_check                                                               */
/* This function prints the contents of the sqlca error information
*/
/* structure.                                                                */
/*****************************************************************************/
long error_check(void)
{
   char          buffer[512]="\0";
   unsigned short i;
   struct sqlca   temp_sqlca;      /* temporary sqlca */     /* @d30369 tjg
*/

   temp_sqlca.sqlcode = 0;          /* initialize the temporary sqlca to
                                    avoid any memory problems */

   if (sqlca.sqlcode != 0) {
      sqlaintp(buffer, sizeof(buffer), 80, &sqlca);
      fprintf(stderr, "\n%0.200s\n", buffer);
      fprintf(outstream, "\n%0.200s\n", buffer);

      /* Decode the SQLCA in more detail  KBS 98/09/28 */
      if ((sqlca.sqlerrml)   /*  there's one or more tokens   */
```

```
        && (sqlca.sqlerrml < sizeof(sqlca.sqlerrmc)) /* and field not full
*/
        )
    {
      char *tokptr;
      int  tokl;
      *(sqlca.sqlerrmc + sqlca.sqlerrml) = '\0';  /*  prevent strtok from
scanning beyond end */
      fprintf(stderr,"\n     SQLCA:   tokens:\n");
      fprintf(outstream,"\n     SQLCA:   tokens:\n");
      tokptr=strtok(sqlca.sqlerrmc, "\xff");
      while (  tokptr                          &&
             ( (tokl = (sizeof(sqlca.sqlerrmc) - (tokptr-sqlca.sqlerrmc))) >
0)
             )
      {
        fprintf(stderr, "%.*s\n", tokl, tokptr);
        fprintf(outstream, "%.*s\n", tokl, tokptr);
        tokptr=strtok(NULL, "\xff");
      }
    }
    fprintf(stderr, "\n     SQLCA:   errp= %.8s, errd 1-6= %d %d %d %d
%d %d\n",
            sqlca.sqlerrp, sqlca.sqlerrd[0], sqlca.sqlerrd[1],
sqlca.sqlerrd[2],
            sqlca.sqlerrd[3], sqlca.sqlerrd[4], sqlca.sqlerrd[5]);
    fprintf(outstream, "\n     SQLCA:   errp= %.8s, errd 1-6= %d %d
%d %d %d %d\n",
            sqlca.sqlerrp, sqlca.sqlerrd[0], sqlca.sqlerrd[1],
sqlca.sqlerrd[2],
            sqlca.sqlerrd[3], sqlca.sqlerrd[4], sqlca.sqlerrd[5]);

    temp_sqlca = sqlca; /* Make a copy of sqlca in case it gets
changed
                        in the next statement below */   /* @d30369 tjg */

    /** Determine if the error is critical or a connection can be made **/

    EXEC SQL CONNECT ;                                  /* @d28763 tjg */

    if (sqlca.sqlcode == SQLE_RC_NOSUDB ) { /* no connection ex-
ists */

      /*Print out header for DUMP*/
      fprintf(outstream, "*************************************\n");
      fprintf(outstream, "*         CONTENTS OF SQLCA         *\n");
      fprintf(outstream, "*************************************\n\n");

      /*Print out contents of SQLCA variables*/
      fprintf(outstream, "SQLCABC  = %ld\n", temp_sqlca.sqlcabc);
      fprintf(outstream, "SQLCODE  = %ld\n", temp_sqlca.sqlcode);
      fprintf(outstream, "SQLERRMC = %0.70s\n",
temp_sqlca.sqlerrmc);
      fprintf(outstream, "SQLERRP  =  %0.8s\n", temp_sqlca.sqlerrp);

      for (i = 0; i < 6; i++)
      {
        fprintf(outstream, "sqlerrd[%d] = %lu \n", i,
temp_sqlca.sqlerrd[i]);
      }

      fprintf(outstream, "SQLWARN  = %0.11s\n",
temp_sqlca.sqlwarn);
      fprintf(outstream, "SQLSTATE = %0.5s\n", temp_sqlca.sqlstate);

      fprintf(stderr, "\nCritical SQLCODE.  Exiting TPCDBATCH\n");
      exit(-1);

    }
  }

  return (temp_sqlca.sqlcode);

}  /* error_check */



/*************************************************/
/* Displays a help screen                        */
/*************************************************/
void display_usage()
{
  printf("\ntpcdbatch  --  version %s",TPCDBATCH_VERSION);
  printf("\n\nSyntax is:\n");
  printf("tpcdbatch [-d dbname] [-f file_name] [-l file_name] [-r on/off]");
  printf("\n        [-v on/off] [-b on/off] [-u p/t/t1/t2]");
  printf("\n        [-s scale_factor] [-n stream_num] [-m inlistmax] [-
h]\n");
  printf("\n  where: -d  Database name");
  printf("\n               Default - dbname set in $DB2DBDFT");
  printf("\n        -f  Input file containing SQL statements");
  printf("\n               Default - stdin ");
  printf("\n        -l  Input file containing list of statement numbers");
  printf("\n        -r  Create set of output files containing query results");
  printf("\n               Default - off");
  printf("\n        -v  Verbose.  Sends information to stderr during");
  printf("\n            query processing");
  printf("\n               Default - off");
  printf("\n        -b  Process groups of statements as blocks ");
  printf("\n            instead of individually.");
  printf("\n               Default - off");
  printf("\n        -u  Update streams: p    - for power test");
  printf("\n                            t    - for throughput test without");
  printf("\n                                   UFs (run this instead of t2)");
  printf("\n                            t1   - for throughput test step 1");
  printf("\n                                   only running queries");
  printf("\n                            t2   - for throughput test step 2");
  printf("\n                                   running update functions");
  printf("\n        -s  Scale factor");
  printf("\n               Default - 0.1");
  printf("\n        -n  Stream number");
  printf("\n                     Default - 0");
  printf("\n                  Qualification - -1");
  printf("\n                        Power - 0");
  printf("\n                    Throughput - >= 1 (actual number depends on
the current query stream");
  printf("\n        -m  Maximum number of keys to delete at a time");
  printf("\n               Default - 400");
  printf("\n        -h  Display this help screen");
  printf("\n        -p  turns smeaphores on or off");
  printf("\n               Default - off");

  printf("\n\nControl statements specifying output and performance de-
tails");
  printf("\ncan be included before SQL statements; they will apply
for");
  printf("\nthat and subsequent statements until updated.");

  printf("\n\nSyntax:    --#SET <control option> <value>");
  printf("\n\n  option      value    default");
  printf("\nROWS_FETCH   -1 to n     -1  (all rows fetched from answer
set)");
  printf("\nROWS_OUT     -1 to n     -1  (all fetched rows sent to out-
put)");
  printf("\n\n--#TAG       tag            (user specified tag name for se-
quence#)");
  printf("\n--#COMMENT   comment         (user specified comments
for output)");
  printf("\nNote:  All statements executed with ISOLATION LEVEL
RR");
  printf("\n       and must be terminated with semi-colons.\n");
```

```
   exit (1);
}


/*************************************************/
/* Converts a string to upper case characters   */
/*************************************************/
char *uppercase( char *string )
{
   char   *c;      /* temp char used to convert word to upper case */

   for ( c = string; *c != '\0'; c++)
      *c = (char) toupper( (int) *c );

   return (string);
}

/*************************************************/
/* Converts a string to lower case characters   */
/*************************************************/
char *lowercase( char *string )
{
   char   *c;      /* temp char used to convert word to lower case */

   for ( c = string; *c != '\0'; c++)
      *c = (char) tolower( (int) *c );

   return (string);
}


/*************************************************/
/* Parses and processes command line options.      */
/*************************************************/

void comm_line_parse(int argc, char *argv[], struct global_struct
*g_struct)
{
   char authent_info[40] = "\0";
   char *testptr;
   int loopvar = 0;

   int comm_opt = 0;
#ifdef PARALLEL_UPDATES
   int running_updates=0;
   int updatePair=-1;
   int updateStream=-1;
   int function;
   int copyOnOrOff;
   int deleteChunk=0;         /*DELjen */
#endif

   while ((loopvar < argc) && (argc != 1)) {

      if (*argv[loopvar] == '-') {

         switch(*(argv[loopvar]+1)) {

          case 'f' :                                  /* @d26350 tjg */
          case 'F' :
                      strcpy(g_struct->c_l_opt->infile,argv[++loopvar]);
                      break;

          case 'l' :                                  /* @d26350 tjg */
          case 'L' :    strcpy(g_struct->c_l_opt-
>str_file_name,argv[++loopvar]);
                      break;

          case 'r' :                                  /* @d26350 tjg */
          case 'R' :
            if (!strcmp(uppercase(argv[++loopvar]),"ON"))
              g_struct->c_l_opt->outfile=1;
            else
              g_struct->c_l_opt->outfile=0;
            break;

          case 'd' :                                  /* @d26350 tjg */
          case 'D' :
                      strcpy(dbname,argv[++loopvar]);
                      break;

          case 'v' :                                  /* @d26350 tjg */
          case 'V' :
            if (!strcmp(uppercase(argv[++loopvar]),"ON"))
              verbose=1;
            else
              verbose=0;
            break;

          case 'u' :                                  /* @d26350 tjg */
          case 'U' :
            g_struct->c_l_opt->update=-1; /* init to invalid number */
            if (!strcmp(uppercase(argv[++loopvar]),"P1"))
              g_struct->c_l_opt->update=1; /* power query stream*/
            if (!strcmp(uppercase(argv[loopvar]),"P2"))
                g_struct->c_l_opt->update=3; /* power update with up-
dates*/
            if (!strcmp(uppercase(argv[loopvar]),"P"))
              g_struct->c_l_opt->update=4; /* power update without up-
dates*/
            if (!strcmp(uppercase(argv[loopvar]),"T1"))
                g_struct->c_l_opt->update=0; /*throughput query stream */
            if (!strcmp(uppercase(argv[loopvar]),"T2"))
                g_struct->c_l_opt->update=2; /* throughput update with
updates */
            if (!strcmp(uppercase(argv[loopvar]),"T"))
                g_struct->c_l_opt->update=5; /* throughput update without
updates */

            break;

          case 'b' :                                  /* @d26350 tjg */
          case 'B' :
            if (!strcmp(uppercase(argv[++loopvar]),"ON"))
              g_struct->s_info_ptr->query_block=1;
            else
              g_struct->s_info_ptr->query_block=0;
            break;

          case 'n' :                                  /* @d26350 tjg */
          case 'N' :
            g_struct->c_l_opt->intStreamNum = atoi(argv[++loopvar]);
            break;

          case 's' :                                  /* @d26350 tjg */
          case 'S' :    g_struct->scale_factor=atof(argv[++loopvar]); break;

          case 'h':
          case 'H' :                                   /* @d26350 tjg */
            display_usage();
            break;

          case 'm' :
          case 'M' :
            inlistmax = atoi(argv[++loopvar]); /* wlc 081897 */
            break;

          case 'p' :
          case 'P' :
             if (!strcmp(uppercase(argv[++loopvar]),"ON")) /* bbe 072599 */
```

```
            semcontrol = 1;
          else
            semcontrol = 0;
          break;


#ifdef PARALLEL_UPDATES
        case 'i':
          updatePair = atoi (argv[++loopvar]);
#ifdef UF2DEBUG
          fprintf (stderr, "updatePair = %d\n",updatePair);
          fflush(stderr);
#endif
          break;

        case 'j':
          function = atoi (argv[++loopvar]);
#ifdef UF2DEBUG
          fprintf (stderr, "function = %d\n",function);
          fflush(stderr);
#endif
          break;

        case 'k':
          updateStream = atoi (argv [++loopvar]);
#ifdef UF2DEBUG
          fprintf (stderr, "updateStream = %d\n",updateStream);
          fflush(stderr);
#endif
          break;

        case 'x':                               /*DEL jen -x is chunk*/
          deleteChunk = atoi (argv[++loopvar]);       /* to delete for this */
#ifdef UF2DEBUG
          fprintf (stderr, "DelChunk = %d\n",deleteChunk);
          fflush(stderr);
#endif
          break;                                /* invocation */

        case 'z':
          running_updates = 1;
          break;
#endif
         default :
          fprintf(stderr,"An invalid option has been set\n");
          display_usage();
          break;

        }  /** end switch **/
      } /** end if **/

      loopvar ++;
    } /** end while **/

    /* checking if -u option is set */
    if (g_struct->c_l_opt->update == -1) {
      fprintf(stderr, "-u option is not set, exiting ...\n");
      exit(-1);
    }


#ifdef PARALLEL_UPDATES
    if (running_updates) {
      if (updatePair == -1) {
        fprintf (stderr, "The parameters to tpcdbatch have not been
passed correctly\n");
        exit (-1);
      }
      else {
        /* check to see if we are to use copy on for the load */
        if (( getenv("TPCD_LOG") != NULL ) &&
            (!strcmp(uppercase(getenv("TPCD_LOG")),"YES")))
        {
          /* okay, we have set LOG_RETAIN on so we need to use copy
directory */
          copyOnOrOff = TRUE;
        }
        else
        {
          /* log retain off don't use copy directory */
          copyOnOrOff = FALSE;
        }

        if (function == 1)
          /* runUF1_fn (updatePair, updateStream);  aph 981205 */
          runUF1_fn (updatePair, updateStream, dbname, userid,
passwd);
        else
          if (function == 2) {
            fprintf(stderr, "A-Calling runUF2_fn  %d   %d   %d ...\n",
                            updatePair, updateStream, deleteChunk);
            /* runUF2_fn (updatePair, updateStream, deleteChunk);   aph
981205 */
            runUF2_fn (updatePair, updateStream, deleteChunk,
dbname, userid, passwd);
          }
          else {
            fprintf (stderr, "Wrong function to tpcdbatch\n");
            exit (-1);
          }
          exit (0);
      }

    }
#endif /* PARALLEL_UPDATES */

    /* If no database name is given, then use the one specified in the
       environment variable DB2DBDFT, otherwise error */
    if (!strcmp(dbname,"\0")) {
      testptr = getenv("DB2DBDFT");
      if (testptr == NULL) {
        fprintf(stderr, "\nNo database name has been specified on com-
mand ");
        fprintf(stderr, "line\nnor in environment variable DB2DBDFT.");
        display_usage();
      }
      else
        strcpy(dbname,testptr);

    }


    if (g_struct->c_l_opt->outfile &&
        !strcmp(g_struct->c_l_opt->str_file_name,"\0")) {
      fprintf(stderr, "\nMust specify input file for statement list.\n");
      display_usage();
    }

}



/*************************************************/
/* Converts DECIMAL values to ASCII text         */
/*************************************************/
int sqlrxd2a(                                   /*kmw*/
                                        /* C++ */char *decptr,
                                        /* C++ */char *asciiptr,
                                        short prec,
                                        short scal)
```

```
{/* */
   int allzero = TRUE;
   /* C++ */char *srcptr;
   unsigned char sign;
   /* C++ */char *targptr, decimal_point = '.';
   int rc = 0;                                    /*kmw*/
   int tmpint, src_nibble;
   int count, j, limit[3];

   targptr = &asciiptr[ prec + 1];
   *(1 + targptr) = '\0';
   srcptr = decptr + prec/2;


   /* Validity check sign nibble */
   if (((sign = sqlrx_get_right_nibble( *srcptr )) < 0x0a)
       || (prec > SQL_MAXDECIMAL) || (prec < scal ))
   {
      goto exit;
   }/** end end if invalid sign value **/


   limit[ 0 ] = scal; limit[ 1 ] = prec - scal; limit[ 2 ] = 0;
   src_nibble = LEFT;
   for( j = 0 ; j < 2 ; j++ )
   {
      for( count = limit[ j ] ; count > 0 ; count-- )
      {
         tmpint = ( (src_nibble == LEFT)?
                  sqlrx_get_left_nibble( *srcptr-- ) :
                  sqlrx_get_right_nibble( *srcptr ) );
         if( tmpint > 9 )
         {
            goto exit;
         }
         else
            *targptr-- = (/* C++ */char)tmpint + '0';
         src_nibble = ((src_nibble == LEFT) ? RIGHT : LEFT);
         if ( tmpint != 0 ) allzero = FALSE;
      } /** end for scal > 0 **/

      if( j == 0 )
         *targptr-- = decimal_point;
      else
         *targptr = (/* C++ */char)((allzero
                                  || (sign == SQLRX_PREFERRED_PLUS)
                                  || (sign == 0x0a)
                                  || (sign == 0x0e)
                                  || (sign == 0x0f)) ?
                                 '+' : '-' );
   } /** end for limit[ j++ ] > 0 **/

   exit :
   if( rc < 0 )
   {
      printf ("The decimal conversion has failed\n");
      exit (-1);

   }

   return(rc);
} /** sqlrxd2a **/


/*********************************************************************/
/* Does some setup and initialization like parsing command line  */
/* and connecting to database.  Returns process id of agent.     */
/*********************************************************************/

void init_setup(int argc, char *argv[], struct global_struct *g_struct)
{
   int connect=0;
#ifndef SQLWINT
   char *pid;
#endif
   char temparray[256]="\0";
   int loopvar=0;
   FILE *updateFP;
   FILE *fpSeed;
   char file_name[256] = "\0";
   short seedEntry;
   long  lSeed;
   int i;

   /** Parse and process command line options **/
   comm_line_parse (argc,argv,g_struct);

/*********************************************************************/
/* Start the mainline report processing.                             */
/*********************************************************************/
   if (!strcmp(g_struct->c_l_opt->infile,"\0")) {
      instream=stdin;
   }
   else {
      instream=NULL;
      if ( (instream = fopen(g_struct->c_l_opt->infile, READMODE)) ==
NULL ) {
         fprintf(outstream, "The input file could not be opened.\n\n");
         fprintf(stdout,"Make sure that the filename is correct.\n");
         fprintf(stdout,"filename = %s\n",g_struct->c_l_opt->infile);
         exit(-1);

      }  /* open the input file if specified  */
   }

   /* IMPORT (begin) - determine whether we should use the IMPORT
api or */
   /* LOAD api for loading into the staging tables, default is load      */
   if (env_tpcd_update_import != NULL)
   {
      if (!strcmp(uppercase(env_tpcd_update_import),"TRUE"))
      {
         iImportStagingTbl = 1;   /* use import */
      }
      /* DJD */
      else if (!strcmp(uppercase(env_tpcd_update_import),"TF"))
      {
         iImportStagingTbl = 2;   /* Table Functions */
      }
   }



   /* IMPORT (end) */

   /* we want to print the seed in the output files to show what seed was
*/
   /* used to generate the queries.  */
   /* if intStreamNum is -1 then we are running a qualification database
*/
   /* and the default seed has been used so skip this section */
   if (g_struct->c_l_opt->intStreamNum >= 0)
   {
      /* check to make sure the TPCD_RUNNUMBER environment vari-
able is set. We */
      /* use this and the stream number to determine which seed was
used to    */
      /* generate the current set of queries */
      if (getenv("TPCD_RUNNUMBER") == NULL)
      {
```

```
        fprintf(stderr,"\nThe TPCD_RUNNUMBER environment variable
is not set");
        fprintf(stderr,"....exiting\n");
        exit(-1);
      }
      if (getenv("TPCD_NUMSTREAM") == NULL)
      {
        fprintf(stderr,"\nThe TPCD_NUMSTREAM environment variable
is not set");
        fprintf(stderr,"....exiting\n");
        exit(-1);
      }

      /*************************************************************************
       * SEED jen
       * we want to print the seed used in the output files.  For the seed
usage
       * we can now reuse the seeds from run to run, therefore all the
power runs
       * will use the 1st seed in the file, and the throughput streams will
use
       * the 2nd to #streams+1 seeds.
       * determine the seed to use...e.g. given 3 streams will have the
following:

       *                              Entry in seed file
       *      TEST       Stream Number     Run 1    Run 2
       *      power      0                 1        1
       *      throughput  1                2        2
       *                 2                 3        3
       *                 3                 4        4
       *************************************************************************/
      seedEntry = g_struct->c_l_opt->intStreamNum + 1;
      /* end SEED jen */
      /* open the generated seed file...if not there, try the default */

      sprintf(file_name, "%s%sauditruns%smseedme",
env_tpcd_audit_dir,
              env_tpcd_path_delim, env_tpcd_path_delim);

      if ((fpSeed = fopen(file_name,READMODE)) == NULL )
      {
        fprintf(stderr,"\nCannot open the seed file, please ensure
that\n");
        fprintf(stderr,"the file exists.  filename = %s\n",file_name);
        exit(-1);
      }
      for (i = 1; i <= seedEntry; i++)
      {
        if (feof(fpSeed))
        {
          lSeed = -1;   /* seed not available for some reason */
        }
        fscanf(fpSeed,"%ld\n",&lSeed);
      }
      g_struct->lSeed = lSeed;
      fclose(fpSeed);
    }

    /* check to see if we are to use copy on for the load */
    if (( getenv("TPCD_LOG") != NULL ) &&
        (!strcmp(uppercase(getenv("TPCD_LOG")),"YES")))
    {
      /* okay, we have set LOG_RETAIN on so we need to use copy di-
rectory */
      g_struct->copy_on_load = TRUE;
    }
    else
    {
      /* log retain off don't use copy directory */
      g_struct->copy_on_load = FALSE;
    }

/*********************************************************************/
/* Make sure that DB2 is started.                                    */
/* CONNECT now unless this is a UF stream for a Throughput test. */
/* (aph 98/12/22)                                                    */
/*********************************************************************/

    if (g_struct->c_l_opt->update > 1)
    {
      /* This is an update function stream in a throughput run. */
      /* Just make sure that DB2 is started.  Each UF child will
CONNECT itself. */
      if (verbose) fprintf(stderr,"\nStarting the DB2 Database Manager
Now\n");
      sqlestar ();
    }
    else
    {  /* In all other cases, CONNECT to the target database. */
      do
      {
        if (!strcmp(userid,"\0"))   /** No authentication provided **/
          EXEC SQL CONNECT TO :dbname;
        else EXEC SQL CONNECT TO :dbname USER :userid USING
:passwd;
        if (sqlca.sqlcode == SQLE_RC_NOSTARTG) {
          if (verbose)
            fprintf(stderr,"\nStarting the DB2 Database Manager Now\n");
          sqlestar ();
          connect=0;
        }
        else connect=1;
      } while (!connect);
      error_check();
    }


/*************************************************************************
 *  All session initialization is performed at connect time or immediately
*
 *  following and is complete before starting the stream.                *
 *************************************************************************/

    /** Get start timestamp for stream **/
    get_start_time(&(g_struct->stream_start_time));     /* TIME_ACC
jen*/
    strcpy(g_struct->file_time_stamp,
           get_time_stamp(T_STAMP_FORM_2,&(g_struct-
>stream_start_time))); /* TIME_ACC jen*/

    if (getenv("TPCD_RUN_DIR") != NULL)
      strcpy(g_struct->run_dir,getenv("TPCD_RUN_DIR"));
    else
      strcpy(g_struct->run_dir,".");

    /* if we are running a throughput test, then we must report the */
    /* stream count information...we will report one file per stream */
    /* and amalgamate them after all streams have completed */
    /* if the number of streams is greater than 0 then this is a throughput
test*/
    switch (g_struct->c_l_opt->update)
    {
        case (2):
        case (5):
              /* update throughput function stream */
              sprintf(file_name,"%s%sstrcntuf.%s",g_struct->run_dir,
                    env_tpcd_path_delim, g_struct->file_time_stamp);
              break;
        case (3):
```

```
      case (4):
            /* update power function stream */
            sprintf(file_name,"%s%spstrcntuf.%s",g_struct->run_dir,
                  env_tpcd_path_delim, g_struct->file_time_stamp);
            break;
      case (1):
            /* power query stream */
            sprintf(file_name, "%s%spstrcnt%d.%s",g_struct->run_dir,
env_tpcd_path_delim,
                  g_struct->c_l_opt->intStreamNum,g_struct-
>file_time_stamp);
            break;
      case (0):
            /* throughput query stream */
            sprintf(file_name, "%s%sstrcnt%d.%s",g_struct->run_dir,
env_tpcd_path_delim,
                  g_struct->c_l_opt->intStreamNum,g_struct-
>file_time_stamp);
            break;
  }

  if( (g_struct->stream_report_file = fopen(file_name, WRITEMODE))
== NULL )
  {
    fprintf(stderr,"\nThe output file for the stream count information\n");
    fprintf(stderr,"could not be opened, make sure the filename is cor-
rect\n");
    fprintf(stderr,"filename = %s\n",file_name);
    exit(-1);
  }

  if (g_struct->c_l_opt->update > 1)
  {
    /* update function stream */
    fprintf(g_struct->stream_report_file,
          "Update function stream starting at %*.*s\n",
          T_STAMP_3LEN,T_STAMP_3LEN, /* TIME_ACC jen*/
          get_time_stamp(T_STAMP_FORM_3,&(g_struct-
>stream_start_time))); /* TIME_ACC jen*/
  }
  else
  {
     /* query stream */
    fprintf(g_struct->stream_report_file,
          "Stream number %d starting at %*.*s\n",
          g_struct->c_l_opt->intStreamNum,
          T_STAMP_3LEN,T_STAMP_3LEN,          /* TIME_ACC jen*/
          get_time_stamp(T_STAMP_FORM_3,&(g_struct-
>stream_start_time))); /* TIME_ACC jen*/
  }

#ifndef LINUX

  fclose(g_struct->stream_report_file);

#endif

  /* set up the update_num_file name so that if we do use sema-
phores, */
  /* we will have a filename to generate the semkey */

  sprintf(g_struct->update_num_file, "%s%s%s.%s.update.pair.num",
env_tpcd_audit_dir,
        env_tpcd_path_delim, uppercase(env_tpcd_dbname), lower-
case(env_user));

  sprintf(g_struct->sem_file, "%s.%s.semfile", env_tpcd_dbname,
env_user);
  if (g_struct->c_l_opt->intStreamNum == 0)
  {
    sprintf(g_struct->sem_file2, "%s.%s.semfile2", env_tpcd_dbname,
env_user);
  }
  if (verbose) {   /* print out the update pair number file for debugging
*/
    fprintf(stderr,"\n init_setup: strem %d update pair numb file = %s\n",
          g_struct->c_l_opt->intStreamNum,g_struct-
>update_num_file);
  }

  /* update the
$TPCD_AUDIT_DIR/$TPCD_DBNAME.$USER.update.pair.num file */
  /* update pairs have been run */
  if (( g_struct->c_l_opt->update >= 1 ) && ( g_struct->c_l_opt->update
< 4 ))
        /* on or onl, but not  */ /* bbe or > 1 */
  {
    updateFP = fopen(g_struct->update_num_file,"r");
    if (updateFP != NULL )
    {
      fscanf(updateFP,"%d",&updatePairStart);
      fclose(updateFP);
      if (g_struct->c_l_opt->intStreamNum == 0)   /* on, 1 update pair
*/
        updatePairStop = updatePairStart + 1;
      else       /* only, multiple update pairs, stream number will be to-
tal */
        updatePairStop = updatePairStart + g_struct->c_l_opt-
>intStreamNum;
      currentUpdatePair = updatePairStart;

      if (updatePairStart <= 0)
      {
        fprintf(stderr,"updatePairStart is bogus!");
        exit(-1);
      }
    }
    else
    {
      fprintf(stderr,"\n %s not set up, set this \n",g_struct-
>update_num_file);
      fprintf(stderr,"file to contain the number of the update pair to
\n");
      fprintf(stderr,"run and resubmit\n");
      exit(-1);
    }
  }

  return ;

}

/*********************************************************************/
/* A function to print out the column titles for a returned set      */
/*********************************************************************/
void print_headings (struct sqlda *sqlda, int *col_lengths)
{
  int col = 0;                    /* Column number             */
  int col_width = 0;              /* width of column           */
  int max_col_width = 0;          /* maximum column width      */
  int col_name_length = 0;        /* sizeof column name string */
  int col_type = 0;               /* column type               */

  int total_length = 0;           /* accumulator var. for
                                  length of column headings */
  int loopvar = 0;

  char col_name[256] = "\0";
  unsigned char m,n;              /* precision and accuracy
                                  for decimal conversion    */
```

```
   fprintf (outstream,"\n");

   /** loop through for each column in solution set
     and determine the maximum column width  **/

   for (col = 0; col < sqlda->sqld; col++) {
      col_name_length=sqlda->sqlvar[col].sqlname.length;
      col_type = sqlda->sqlvar[col].sqltype;
      col_width = sqlda->sqlvar[col].sqllen;
      strncpy(col_name,(char *)sqlda-
>sqlvar[col].sqlname.data,col_name_length) ;

      switch (col_type)
      {
       case SQL_TYP_SMALL:
       case SQL_TYP_NSMALL:                              /* @d30369 tjg */
         col_lengths[col] = TPCDBATCH_MAX (col_name_length,6);
         break;
       case SQL_TYP_INTEGER:
       case SQL_TYP_NINTEGER:
         col_lengths[col] = TPCDBATCH_MAX (col_name_length,11);
         break;
       case SQL_TYP_BIGINT:   /*kmwBIGINT*/
       case SQL_TYP_NBIGINT:
         col_lengths[col] = TPCDBATCH_MAX (col_name_length,19);
         break;
       case SQL_TYP_CSTR:
       case SQL_TYP_NCSTR:
       case SQL_TYP_DATE:
       case SQL_TYP_NDATE:
       case SQL_TYP_TIME:
       case SQL_TYP_NTIME:
       case SQL_TYP_STAMP:
       case SQL_TYP_NSTAMP:
       case SQL_TYP_CHAR:
       case SQL_TYP_NCHAR:
       case SQL_TYP_VARCHAR:
       case SQL_TYP_NVARCHAR:
       case SQL_TYP_LONG:
       case SQL_TYP_NLONG:
         col_lengths[col] = TPCDBATCH_MAX
(col_name_length,col_width);
         break;

       case SQL_TYP_FLOAT:
       case SQL_TYP_NFLOAT:
         /* kmw - note: TPCDBATCH_PRINT_FLOAT_WIDTH > max
long identifier */
         col_lengths[col] = TPCDBATCH_PRINT_FLOAT_WIDTH;
         break;

       case SQL_TYP_DECIMAL:
       case SQL_TYP_NDECIMAL:

         m=(*(struct declen *)&sqlda->sqlvar[col].sqllen).m;
         n=(*(struct declen *)&sqlda->sqlvar[col].sqllen).n;

         col_lengths[col] = TPCDBATCH_MAX ((int)(m+n),
col_name_length);
         /* Special handling for DECIMAL */    /* @d26350 tjg */
         break;

       default:
         fprintf(stderr,"--Unknown column type (%d).  Abort-
ing.\n",col_type);
         break;
      }

      fprintf(outstream,"%-*.*s
",col_lengths[col],col_name_length,col_name);

      total_length += (col_lengths[col] + 2);  /* 2 is from padding spaces
*/
   }

   fprintf(outstream,"\n");
   for (loopvar=0; loopvar < total_length; loopvar++)
      fprintf(outstream,"-");
   fprintf(outstream,"\n");

}


/*****************************************************************/
/* Gets the current system time and prints it out                */
/*****************************************************************/
char *get_time_stamp(int form, Timer_struct *time_pointer)
{
   Timer_struct temp_stamp;  /* TIME_ACC jen */
   struct tm *tp;
   size_t timeLength = 0;

   /* TIME_ACC jen start */
   if (time_pointer == (Timer_struct *)NULL)
      get_start_time(&temp_stamp);
   else
     temp_stamp = *time_pointer;

#if defined (SQLUNIX) || defined (SQLAIX)
   tp = localtime((time_t *)&(temp_stamp.tv_sec));
#elif (defined (SQLOS2) || defined(SQLWINT) || defined (SQLWIN) ||
defined(SQLDOS))
   tp = localtime(&(temp_stamp.time));
#else
#error Unknown operating system
#endif
   /* TIME_ACC jen stop*/

   if ((form == T_STAMP_FORM_1) || (form == T_STAMP_FORM_3))
   {
   /* SUN fix bbe start */
#if (defined (SQLWINT) || defined (SQLWIN) || defined (SQLOS2) ||
defined(SQLDOS))
      timeLength = strftime(newtime,50,"%x %X",tp);
#elif (defined (SQLUNIX) || defined (SQLAIX))
      timeLength = strftime(newtime,50,"%D %T",tp);  /* SUN  ...test this
*/
#else
#error Unknown operating system
#endif
   /* SUN fix bbe stop */
      /* TIME_ACC jen start*/
      if (form == T_STAMP_FORM_3)
      {

         /* concatenate the microsecond/milliseconds  on the end of the
*/
         /*timestamp jen1006 */
#if defined (SQLUNIX) || defined (SQLAIX)
         sprintf(newtime+timeLength,".%0.6d",temp_stamp.tv_usec);
#elif (defined (SQLOS2) || defined(SQLWINT) || defined (SQLWIN) ||
defined(SQLDOS))
         sprintf(newtime+timeLength,".%0.3d",temp_stamp.millitm);
#else
#error Unknown operating system
#endif
         /* TIME_ACC jen stop*/
      }
```

```
  }
  else
    if (form == T_STAMP_FORM_2)
      strftime(newtime,50,"%y%m%d-%H%M%S",tp);

  return (newtime);

}



/*********************************************************************/
/* Handle all the processing for the summary table                   */
/*********************************************************************/

void summary_table (struct global_struct *g_struct)
{
  double arith_mean = 0;
  double geo_mean   = 0;
  int    num_stmt   = 0;
  int    num_stmt_for_geo_mean   = 0;

  double adjusted_a_mean = 0;
  double adjusted_g_mean = 0;
  double adjusted_g_mean_intern;
  double adjusted_max_time = 0;

  double Ts   = 0;                  /* different TPC-D metrics */
  double Ts1;
  double Ts2;
/* double QppD = 0;                    MARK
  double QthD = 0;
  double QphD = 0; */

  double db_size_frac_part = 0;       /* stores the fractional part of db
size */
  double db_size = 0;                 /* size in numbers */
  char db_size_qualifier[3] = "\0";   /* MB, GB or TB */

  struct stmt_info
    *s_info_ptr,
    *s_info_head_ptr,
    *max,
    *min;


  /* Determine the size of the database from the scale factor (1 SF =
1GB) */
  if (g_struct->scale_factor < 1.0) {
      db_size = g_struct->scale_factor * 1000;
      strcpy(db_size_qualifier, "MB");
  } else if (g_struct->scale_factor >= 1000.0) {
      db_size = g_struct->scale_factor / 1000;
      strcpy(db_size_qualifier, "TB");
  } else {
      db_size = g_struct->scale_factor;
      strcpy(db_size_qualifier, "GB");
  }


  /* computes the fractional part of db_size */
  db_size_frac_part = db_size - (int) db_size;

  s_info_ptr = g_struct->s_info_ptr;  /* Just use a local copy */
  s_info_head_ptr = s_info_ptr;

  max = s_info_head_ptr;
  /* ensure that we are not already setting max to the UF timings */
  while ( strstr(max->tag, "UF") != NULL )
    max = max->next;
  min = max;

  if (g_struct->c_l_opt->outfile)     /* create the appropriate output file */
    output_file(g_struct);

  /* write the seed used for this run unless it is a qualification run */
  /* (qualification runs use the default seed for their queries) or  */
  /* unless it is the update function stream (no seeds used for this) */
  /* (this is an update stream iff update is 2) */
  if ((g_struct->c_l_opt->intStreamNum >=0) &&
      (g_struct->c_l_opt->update != 2) )
  {
    if (g_struct->lSeed == -1)
    {
      fprintf( outstream,"\nUsing default qgen seed file");
    }
    else
      fprintf (outstream, "\nSeed used for current run = %ld",g_struct-
>lSeed);
    fprintf( outstream,"\n");
  }

  /* print out the stream number if we are in a throughput stream and if
*/
  /* this is not the update stream portion of the throughput test */
  if ( (g_struct->c_l_opt->intStreamNum > 0) &&
       (g_struct->c_l_opt->update != 2)  )
  {
    fprintf( outstream, "Stream number = %d\n",g_struct->c_l_opt-
>intStreamNum);
  }
  /* print the stream start timestamp to the inter file */
  fprintf (outstream, "Stream start time stamp %*.*s\n",
           T_STAMP_3LEN,T_STAMP_3LEN,  /* TIME_ACC jen*/
           get_time_stamp(T_STAMP_FORM_3,&(g_struct-
>stream_start_time))); /* TIME_ACC jen*/
  /* print the stream stop timestamp to the inter file */
  fprintf (outstream, "Stream stop time stamp %*.*s\n",
           T_STAMP_3LEN,T_STAMP_3LEN,  /* TIME_ACC jen*/
           get_time_stamp(T_STAMP_FORM_3,&(g_struct-
>stream_end_time)));  /* TIME_ACC jen*/

  printf("At Summary of Results\n");

  fprintf (outstream, "\n\n\nSummary of Re-
sults\n==================\n");
  fprintf (outstream,
           "\nSequence #     Elapsed Time   Adjusted Time Start Time-
stamp   End Timestamp\n\n");

  /* Go through the linked list and determine which statement had the
     highest and lowest elapsed times */
  while ( (s_info_ptr != NULL) && (s_info_ptr != g_struct-
>s_info_stop_ptr) ) {

    /* check if we are in an update function...if so, we do not want to */
    /* consider the update function times as the min or max time */
    if ( strstr(s_info_ptr->tag,"UF") == NULL )
    {
      /* we are not in an update function */
      if (s_info_ptr->elapse_time > max->elapse_time)
        max = s_info_ptr;
      else
        if ((s_info_ptr->elapse_time < min->elapse_time)
            && (s_info_ptr->elapse_time > -1))
          min = s_info_ptr;
    }

    s_info_ptr = s_info_ptr->next;
```

```
  }

  s_info_ptr = s_info_head_ptr;

  /** Start from the first structure and go through until the stop
    pointer is reached **/
  while ( (s_info_ptr != NULL) && (s_info_ptr != g_struct-
>s_info_stop_ptr) ) {

    if (s_info_ptr->elapse_time != -1)  {

      s_info_ptr->adjusted_time = s_info_ptr->elapse_time;
      /* determine whether the elapsed times have to be adjusted or
not */
      /* if this is an update function, we do not adjust the elapsed
time*/
      if ( strstr(s_info_ptr->tag,"UF") == NULL )
      {
        /* this is not an update function, adjust time if necessary */
        if (max->elapse_time/min->elapse_time > 1000)
        {
          /* jmc fix geo_mean calculation...round adjusted time prop-
erly ROUNDING*/
          adjusted_max_time = max->elapse_time/1000;
          if (s_info_ptr->elapse_time < adjusted_max_time)
          {
            s_info_ptr->adjusted_time =
              (double)(((long)((adjusted_max_time + 0.05) * 10))/10.0);
            if (s_info_ptr->adjusted_time < 0.1)
              s_info_ptr->adjusted_time = 0.1;
          }
          /*jmc fix geo_mean calculation...round adjusted time properly
ROUNDING end*/
        }
      }

                                    /* a value was calculated */
      fprintf (outstream,
            "%-5d %-5.5s %15.1f %15.1f %*.*s %*.*s\n",
             s_info_ptr->stmt_num,s_info_ptr->tag,
             s_info_ptr->elapse_time,s_info_ptr->adjusted_time,
             T_STAMP_1LEN,T_STAMP_1LEN,s_info_ptr-
>start_stamp, /* TIME_ACC jen*/
             T_STAMP_1LEN,T_STAMP_1LEN,s_info_ptr-
>end_stamp); /* TIME_ACC jen*/

      /* Only update arithmetic mean for queries not update functions
*/
      if ( strstr(s_info_ptr->tag,"UF") == NULL )
      {
        arith_mean += s_info_ptr->elapse_time;
        adjusted_a_mean += s_info_ptr->adjusted_time;
      }

      if (s_info_ptr->elapse_time > 0) {  /* don't bother finding log of
                                    numbers < 0 */
        geo_mean += log(s_info_ptr->elapse_time);
        adjusted_g_mean += log(s_info_ptr->adjusted_time);
      }


      /* Only update num_stmt for queries not update functions */
      if ( strstr(s_info_ptr->tag,"UF") == NULL )
        num_stmt ++;
      num_stmt_for_geo_mean++;
    }

    else
      fprintf (outstream,"%-5d %-5.5s %-15s %-15s\n",
             s_info_ptr->stmt_num,
             s_info_ptr->tag,"Not Collected", "Not Collected");


      if (s_info_ptr != g_struct->s_info_stop_ptr)
        s_info_ptr=s_info_ptr->next;
  }

  fprintf(outstream, "\n\nNumber of statements: %d\n\n", s_info_ptr-
>stmt_num - 1);
  /* Calculate the arithmetic and geometric means */

  if (geo_mean != 0) {     /*Used to test if arith_mean != 0
                        Don't bother doing any of this if the

                         elapsed time mean is 0 */
    arith_mean = arith_mean / num_stmt;
    adjusted_a_mean = adjusted_a_mean / num_stmt;
    geo_mean = exp(geo_mean / num_stmt_for_geo_mean);
    adjusted_g_mean_intern = adjusted_g_mean; /*MARK*/
    adjusted_g_mean = exp(adjusted_g_mean /
num_stmt_for_geo_mean);

  }


  /* print out all the appropriate information including the
    different TPC-D metrics */
  /* do not bother with this if we are in an update only stream */
  fprintf (outstream, "\nGeom.  mean queries %7.3f %15.3f\n",\
          geo_mean,adjusted_g_mean);
  if (g_struct->c_l_opt->update < 2)
  {
    fprintf (outstream, "Arith. mean queries %7.3f %15.3f\n",\
          arith_mean,adjusted_a_mean);


    fprintf (outstream,
          "\n\nMax Qry %-3.3s %15.1f %15.1f %*.*s %*.*s\n",
          max->tag,max->elapse_time,max->adjusted_time,
          T_STAMP_1LEN,T_STAMP_1LEN,max->start_stamp,  /*
TIME_ACC jen*/
          T_STAMP_1LEN,T_STAMP_1LEN,max->end_stamp);  /*
TIME_ACC jen*/
    fprintf (outstream,
          "Min Qry %-3.3s %15.1f %15.1f %*.*s %*.*s\n",
          min->tag,min->elapse_time,min->adjusted_time,
          T_STAMP_1LEN,T_STAMP_1LEN,min->start_stamp,  /*
TIME_ACC jen*/
          T_STAMP_1LEN,T_STAMP_1LEN,min->end_stamp);  /*
TIME_ACC jen*/
  }

  if (g_struct->c_l_opt->intStreamNum == 0) {
    /* fprintf (outstream, "\n\nMetrics\n=======\n\n"); */

    /* Increase the Ts measurement by one second since the accuracy
of our */
    /* timestamps is only to 1 second and if the start was at 1.01 sec-
onds, */
    /* and the end was at 5.99 seconds, we get a free second ... this
will */
    /* be made explicit in the upcoming revision of the spec (after
1.0.1) */
     /* TIME_ACC jen start*/
    /* NOTE this can probably be better coded by changing
get_elapsed_time */
    /* to just calculate the elapsed time give a start and an end time,
and */
    /* to also give a precision for the calculation (sec, 10ths....).  The */
```

```
    /* call then will grab a timestamp before calling. THen we can get
rid */
    /* of the if def...and just call get_elapsed_time (whcih can handle
the */
    /* os differences on its own */

#if defined (SQLUNIX) || defined (SQLAIX)
    Ts = g_struct->stream_end_time.tv_sec - g_struct-
>stream_start_time.tv_sec + 1;
    Ts1 = (double)g_struct->stream_start_time.tv_sec + ((dou-
ble)g_struct->stream_start_time.tv_usec/1000000);
    Ts2 = (double)g_struct->stream_end_time.tv_sec + ((dou-
ble)g_struct->stream_end_time.tv_usec/1000000);

#elif (defined (SQLOS2) || defined(SQLWINT) || defined (SQLWIN) ||
defined(SQLDOS))
    Ts = g_struct->stream_end_time.time - g_struct-
>stream_start_time.time + 1;
    Ts1 = (double)g_struct->stream_start_time.time + ((dou-
ble)g_struct->stream_start_time.millitm/1000);
    Ts2 = (double)g_struct->stream_end_time.time + ((dou-
ble)g_struct->stream_end_time.millitm/1000);

#else
#error Unknown operating system
#endif

    /* TIME_ACC jen stop*/

   /* MARK
    ##Now do in calcmetricsp.pl##
    QppD = (3600 * g_struct->scale_factor) / adjusted_g_mean;
    QthD = (num_stmt * 3600 * g_struct->scale_factor) / Ts;
    QphD = sqrt(QppD*QthD);
   */
    /* if the decimal part has some meaningful value then print the da-
tabase size
    with decimal part; otherwise just print the integer part */

        fprintf (outstream,
            "\nGeometric mean interim value  = %10.3f\n\nStream Ts
%11 = %10.0f\n\nStream start int representation %11 = %f\n\nStream
stop int representation  %11 = %f",
            adjusted_g_mean_intern,Ts,Ts1,Ts2);
  }

}


/*****************************************************************/
/* free up all the elements of the sqlda after done processing */
/*****************************************************************/
void free_sqlda (struct sqlda *sqlda, int select_status)    /* @d30369
tjg */
{
  int loopvar;

 printf("In free_sqlda\n");
  if (select_status == TPCDBATCH_SELECT)
    for (loopvar=0; loopvar<sqlda->sqld; loopvar++) {
      free(sqlda->sqlvar[loopvar].sqldata);
      printf("after the 1st free\n");
      free(sqlda->sqlvar[loopvar].sqlind);
      printf("after the 2nd free\n");
    }

  free(sqlda);
  printf("after the 3rd free\n");
  sqlda_allocated = 0; /* fix free() problem on NT
                    wlc 090597 */
}


/************************************************/
/* processing to run the insert update function */
/************************************************/
void runUF1 ( struct global_struct *g_struct, int updatePair )
{

  char statement[3000];
  char sourcedir[256];


  int split_updates = 2;      /* no. of ways update records are split */
  int concurrent_inserts = 2;  /* jenCI no of concurrent updates to be */
                               /* jenCI run at once*/
  int loop_updates = 1;       /* jenCI no of updates to be run in one  */
                               /* jenCI "concurrent" invocation.  should*/
                               /* jenCI be split_updates / concurrent_inserts*/
  int i;
  int streamNum;
#ifdef SQLWINT
  /* PROCESS_INFORMATION childprocess[100]; */
  char commandline[256];
  HANDLE              su_hSem;
  char              UF1_semfile[256];
#else
  int childpid[100];
  int               su_semid; /* semaphore for controlling split updates*/
  key_t              su_semkey; /* key to generate semid */
#endif
  if (g_struct->c_l_opt->intStreamNum == 0)
     streamNum = 0;
  else
     streamNum = currentUpdatePair - updatePairStart + 1;

  fprintf( outstream,"UF1 for update pair %d, stream %d, start-
ing\n",updatePair, streamNum);

  /* Start by loading the data into the staging table at each node */
  /* The orderkeys were split earlier by the split_updates program */
  if (env_tpcd_audit_dir != NULL)
    strcpy(sourcedir,env_tpcd_audit_dir);

  else
    strcpy(sourcedir,".");

  /* Load the orderkeys into the staging table */
  /* In SMP environments one could use a load command but by using
a */
  /* script we can keep the code common */
#ifdef SQLWINT
  sprintf (statement, "perl %s\\tools\\ploaduf1 %d\n", sourcedir, up-
datePair);
#else
  sprintf (statement, "perl %s/tools/ploaduf1 %d 1", sourcedir, update-
Pair);
#endif
  if (system(statement))
  {
    fprintf (stderr, "ploaduf1 failed for UF1, examine UF1.log for cause.
Exiting.\n");
    if (verbose)
      fprintf (stderr,
          "ploaduf1 failed for UF1, examine UF1.log for cause. Exit-
ing.\n");
    exit (-1);
  }

  fprintf (outstream, "load_update finished for UF1.\n");
```

```
  if (getenv ("TPCD_SPLIT_UPDATES") != NULL)
    split_updates = atoi (getenv ("TPCD_SPLIT_UPDATES"));
  if (getenv ("TPCD_CONCURRENT_INSERTS") != NULL)
/*jenCI*/
    concurrent_inserts = atoi (getenv
("TPCD_CONCURRENT_INSERTS"));  /*jenCI*/
  loop_updates = split_updates / concurrent_inserts;
/*jenCI*/

#ifndef SQLWINT
  /*    we will use the tpcd.setup file to generate the semaphore key */
  if (getenv("TPCD_AUDIT_DIR") != NULL)                /*begin SEMA
*/
  {
    /* this is assuming that you will be running this from 0th node */
    sprintf(sourcefile, "%s%ctools%ctpcd.setup",
          getenv("TPCD_AUDIT_DIR"), PATH_DELIM,PATH_DELIM);
  }
  else
  {
    fprintf (stderr, "runUF1 Can't open UF1 semaphore
file,TPCD_AUDIT_DIR is not defined.\n");
    exit (-1);
  }
  /*end SEMA */
  su_semkey = ftok (sourcefile, 'J');
  if ( (su_semid = semget (su_semkey, 1,
IPC_CREAT|S_IRUSR|S_IWUSR)) < 0)
  {
    fprintf (stderr, "Cannot get semaphore! semget failed: errno =
%d\n",errno);
    exit (-1);
  }
#else  /* SQLWINT */
  sprintf (UF1_semfile, "%s.%s.UF1.semfile", env_tpcd_dbname,
env_user);
  su_hSem = CreateSemaphore(NULL, 0,
                        concurrent_inserts,                    /*jenCI*/
                        (LPCTSTR)(UF1_semfile));
  if (su_hSem == NULL)
  {
    fprintf(stderr,
          "CreateSemaphore (ready semaphore) failed, GetLastError:
%d, quitting\n",
          GetLastError());
    exit(-1);
  }
#endif /* SQLWINT */
  if (verbose) fprintf(stderr,"Semaphore created successfully!\n");

  fclose(outstream); /* to prevent multiple header caused by forking
                    wlc 081397 */

  for (i=0; i < concurrent_inserts; i++)                       /*jenCI*/
  {
#ifndef SQLWINT
    if ((childpid[i] = fork()) == 0)
    {
      /* runUF1_fn (updatePair, i);   aph 981205 */
      runUF1_fn (updatePair, i, dbname, userid, passwd);
    }
    else
    {
      /* This is the parent */
      if (verbose)
        fprintf (stderr, "stream #%d started with pid %d\n", i, child-
pid[i]);
    }
#else  /* SQLWINT */
    sprintf (commandline,
          "start /b %s\\auditruns\\tpcdbatch.exe -z -d %s -i %d -j 1 -k
%d",
          env_tpcd_audit_dir, dbname, updatePair, i ); /* aph 082797
*/

    system (commandline);
#endif /* SQLWINT */
    sleep (UF1_SLEEP);
  }

  /* All children have been created, now wait for them to finish */
#ifndef SQLWINT
  if (sem_op (su_semid, 0, concurrent_inserts * -1) != 0)
/*jenCI*/
  {                                                            /*jenSEM*/
    fprintf(stderr,
          "Failure to wait on insert semaphone with %d of children\n",
          concurrent_inserts);
    exit(1);
  }                                                            /*jenSEM*/
  semctl (su_semid, 0, IPC_RMID, 0);
#else
  for (i = 0; i < concurrent_inserts; i++)                     /*jenCI*/
  {
    if (verbose)
    {
      fprintf(stderr,"About to wait again ...Sets to wait for %d\n",
            concurrent_inserts - i);                           /*jenCI*/
    }
    if (WaitForSingleObject(su_hSem, INFINITE) == WAIT_FAILED)
    {
      fprintf(stderr,
            "WaitForSingleObject (su _hSem) failed in runUF1 on set
%d, error: %d, quitting\n",
            i, GetLastError());
      exit(-1);
    }
  }
  if (! CloseHandle(su_hSem))
  {
    fprintf(stderr,
          "RunUF1 Close Sem failed - Last Error: %d\n", GetLastEr-
ror());
    /* no exit here */
  }
#endif

  if( (outstream = fopen(outstreamfilename, APPENDMODE)) ==
NULL )
  {
    fprintf(stderr,"\nThe output file could not be opened.  ");
    fprintf(stderr,"Make sure that the filename is correct.\n");
    fprintf(stderr,"filename = %s\n",outstreamfilename);
    exit(-1);
  }

  fprintf( outstream,"UF1 for update pair %d complete\n",updatePair);
}


/* runUF1_fn() moved to another SQC file                    aph 981205 */


/*************************************************/
/* processing to run the delete update function */
/*************************************************/
void runUF2 ( struct global_struct *g_struct, int updatePair )
{
  char statement[3000];
```

```
   char sourcedir[256];

   int split_deletes = 1;    /*   no. of ways update records are split
@dxxxxxhar */
   int concurrent_deletes  = 1;    /* number of database partitions DEL-
jen */
   int chunks_per_concurrent_delete = 1;

   int i;
   int streamNum;
#ifdef SQLWINT
   char commandline[256];
   HANDLE               su_hSem;
   char               UF2_semfile[256];
#else
   int childpid[100];
   char sourcefile[256];
   int                su_semid; /* semaphore for controlling split updates*/
   key_t               su_semkey; /* key to generate semid */
#endif
    if (g_struct->c_l_opt->intStreamNum == 0)
       streamNum = 0;
   else
       streamNum = currentUpdatePair - updatePairStart + 1;

   fprintf( outstream,"UF2 for update pair %d, stream %d, start-
ing\n",updatePair, streamNum);

   /* We need to know both how many chunks there are and how many
chunks*/
   /* are to be executed by each concurrent UF2 process.   More
chunks means */
   /* both smaller transactions (less deadlock) and more potential con-
currency */

   /* How many "chunks" have the orderkeys been divided into? */
   if (getenv ("TPCD_SPLIT_DELETES") != NULL)
      split_deletes = atoi (getenv ("TPCD_SPLIT_DELETES"));
   /* How many deletes should run concurrently */
   if (getenv ("TPCD_CONCURRENT_DELETES") != NULL)
      concurrent_deletes = atoi (getenv
("TPCD_CONCURRENT_DELETES"));
   /* How many chunks in each concurrently running delete process */
   chunks_per_concurrent_delete = split_deletes / concurrent_deletes;


   /* Start by loading the data into the staging table at each node */
   /* The orderkeys were split earlier by the split_updates program */
   if (env_tpcd_audit_dir != NULL)
      strcpy(sourcedir,env_tpcd_audit_dir);
   else
      strcpy(sourcedir,".");

   /* Load the orderkeys into the staging table */
   /* In SMP environments one could use a load command but by using
a */
   /* script we can keep the code common */


#ifdef SQLWINT
   sprintf (statement, "perl %s\\tools\\ploaduf2 %d\n", sourcedir, up-
datePair);
#else
   sprintf (statement, "perl %s/tools/ploaduf2 %d 2", sourcedir, update-
Pair);
#endif
   if (system(statement))
   {
      fprintf (stderr, "ploaduf2 failed for UF2, examine UF2.log for cause.
Exiting.\n");
      exit (-1);
   }
   fprintf (outstream, "ploaduf2 finished for UF2.\n");

   fclose(outstream); /* to prevent multiple header caused by forking
                         wlc 081397 */

   /* Next we need to get ready to launch a bunch of concurrent proc-
esses */
#ifndef SQLWINT
   /*    we will use the tpcd.setup file to generate the semaphore key
begin SEMA */
   if (getenv("TPCD_AUDIT_DIR") != NULL)
   {
      sprintf(sourcefile, "%s%ctools%ctpcd.setup",
            getenv("TPCD_AUDIT_DIR"), PATH_DELIM, PATH_DELIM);
   }
   else
   {
      fprintf (stderr, "runUF2 Can't open UF2 semaphore file,
TPCD_AUDIT_DIR is not defined.\n");
      exit (-1);
   }

   su_semkey = ftok (sourcefile, 'D');   /* use D for deletes    */
   /* end SEMA */
   if ( (su_semid = semget (su_semkey, 1,
IPC_CREAT|S_IRUSR|S_IWUSR)) < 0)
   {
      fprintf (stderr, "UF2 Can't get semaphore! semget failed: errno =
%d\n",
               errno);
      exit (-1);
   }
#else
   sprintf (UF2_semfile, "%s.%s.UF2.semfile", env_tpcd_dbname,
env_user);
   fprintf(stderr,"UF2 semfile = %s\n",UF2_semfile);
   su_hSem = CreateSemaphore(NULL, 0,
                         concurrent_deletes,
                         (LPCTSTR)(UF2_semfile));
   if (su_hSem == NULL)
   {
      fprintf(stderr,
            "CreateSemaphore (ready semaphore) failed, GetLastError:
%d, quitting\n",
            GetLastError());
      exit(-1);
   }
   fprintf(stderr,"Semaphore created successfully!\n");
#endif

   for (i=0; i < concurrent_deletes; i++)
   {
#ifndef SQLWINT
      if ((childpid[i] = fork()) == 0)
      {
         fprintf(stderr, "B-Calling runUF2_fn  %d   %d   %d ...\n",
                           updatePair, i,chunks_per_concurrent_delete);
         /* runUF2_fn (updatePair, i, chunks_per_concurrent_delete);
aph 981205 */
         runUF2_fn (updatePair, i, chunks_per_concurrent_delete,
dbname, userid, passwd);
      }
      else
      {
         /* This is the parent */
         if (verbose)
            fprintf (stderr, "stream #%d started with pid %d\n", i, child-
pid[i]);
```

```
    }
#else
    {
      /*  SECURITY_ATTRIBUTES sec_process;
          SECURITY_ATTRIBUTES sec_thread; */
      /* NEED TO FIX THIS UP - KBS 98/10/20 */

      sprintf (commandline,
          "start /b %s\\auditruns\\tpcdbatch.exe -z -d %s -i %d -j 2 -k
%d -x %d",
          env_tpcd_audit_dir, dbname, updatePair, i,
chunks_per_concurrent_delete  ); /* aph */
      /* the -x parm should be passed at 0...not 100% sure of this jen
*/
      fprintf(stderr, "commandline= %s\n", commandline);
      system (commandline);
      sleep (UF2_SLEEP);
    }
#endif
  }

  /* All children have been created, now wait for them to finish */
#ifndef SQLWINT
  fprintf(stderr, "About to wait on the semaphore...\n");
  if (sem_op (su_semid, 0, concurrent_deletes * -1) != 0)
/*jenSEM*/
  {                                                   /*jenSEM*/
    fprintf(stderr,
          "Failure to update wait on delete semaphone with %d chil-
dren\n",
          concurrent_deletes);
    exit(1);
  }                                                   /*jenSEM*/
  semctl (su_semid, 0, IPC_RMID, 0);
#else
// for (i = 0; i < split_deletes; i++)   //DJD Waits forever.....
  for (i = 0; i < concurrent_deletes; i++)
  {
    if (verbose)
    {
//      fprintf(stderr,"About to wait again ...Sets to wait for %d\n",
//              split_deletes - i);
      fprintf(stderr,"About to wait again ...Sets to wait for %d\n",
            concurrent_deletes - i);
    }
    if (WaitForSingleObject(su_hSem, INFINITE) == WAIT_FAILED)
    {
      fprintf(stderr,
            "WaitForSingleObject (su_hSem) failed on set %d, error:
%d, quitting\n",
            i, GetLastError());
      exit(-1);
    }
  }
  if (! CloseHandle(su_hSem))
  {
    fprintf(stderr, "Close Sem failed - Last Error: %d\n", GetLastEr-
ror());
    /* no exit here */
  }
#endif

  if( (outstream = fopen(outstreamfilename, APPENDMODE)) ==
NULL )
  {
    fprintf(stderr,"\nThe output file could not be opened.  ");
    fprintf(stderr,"Make sure that the filename is correct.\n");
    fprintf(stderr,"filename = %s\n",outstreamfilename);
    exit(-1);
  }
  fprintf( outstream,"UF2 for update pair %d complete\n",updatePair);

}


/* runUF2_fn() moved to another SQC file                    aph 981205 */


/*--------------------------------------------------------------*/
/*       General semaphore function.                            */
/*--------------------------------------------------------------*/
#ifndef SQLWINT
int sem_op (int semid, int semnum, int value)
{
  struct sembuf sembuf;    /* = {semnum ,value,0}; */
  sembuf.sem_num = semnum;
  sembuf.sem_op  = value;
  sembuf.sem_flg = 0;

  if (semop(semid,&sembuf,1) < 0)
  {
    fprintf(stderr,"ERROR*** sem_op errorno = %d\n", errno);
    return(-1);
    /*  exit(1); */
  }
  return (0);       /* successful return  jenSEM */
}
#endif

/*****************************************************************/
/* Determines the proper name for the output file to
  be generated for a particular TPC-D query, update function, or
  interval summary                                              */
/*****************************************************************/
void output_file(struct global_struct *g_struct)
{
  char file_name[256] = "\0";
  char run_dir[150]   = "\0";
  char time_stamp[50] = "\0";
  char delim[2]       = "\0";
  int qnum;


  strcpy(run_dir,g_struct->run_dir);
  sprintf(delim,"%s",env_tpcd_path_delim);
  strcpy(time_stamp,g_struct->file_time_stamp);

  if (g_struct->stream_list == NULL)
    if ((g_struct->stream_list =
              fopen(g_struct->c_l_opt->str_file_name, READMODE))
== NULL)
    {
      fprintf(stderr,"\nThe stream list file could not be opened.");
      fprintf(stderr,"Make sure that the filename is correct.\n");

      exit(-1);
    }

  fscanf(g_struct->stream_list,"%d",&qnum);

  switch (g_struct->c_l_opt->intStreamNum)
  {
   case -1: /* qualifiying */
     sprintf(file_name,
"%s%sqryqual%02d.%s",run_dir,delim,qnum,time_stamp);
     break;
   case 0: /* power tests */
     if (qnum < 0) /* update functions */
```

```
        sprintf(file_name,
"%s%smps00uf%d.%02d.%s",run_dir,delim,abs(qnum), \
                currentUpdatePair,time_stamp);
      else
        sprintf(file_name,
"%s%smpqry%02d.%s",run_dir,delim,qnum,time_stamp);
      break;

    default:
      /*      if (qnum < 0)  - replaced by berni 96/03/26 */
      if (g_struct->c_l_opt->update == 2 ||
          g_struct->c_l_opt->update == 5)
        sprintf(file_name,
"%s%smts%02duf%d.%02d.%s",run_dir,delim, \
                currentUpdatePair - updatePairStart + 1,abs(qnum), curren-
tUpdatePair,time_stamp);
      else
        sprintf(file_name, "%s%smts%dqry%02d.%s",run_dir,delim, \
                g_struct->c_l_opt->intStreamNum,qnum,time_stamp);
      break;

  }

  if (g_struct->c_flags->eo_infile)
    if (g_struct->c_l_opt->update == 2 ||
        g_struct->c_l_opt->update == 5)
      sprintf(file_name,
"%s%smtufinter.%s",run_dir,delim,time_stamp);
    else
      switch (g_struct->c_l_opt->intStreamNum) {
       case -1:
        sprintf(file_name,
"%s%sqryqualinter.%s",run_dir,delim,time_stamp);
         break;
       case 0:
        /*sprintf(file_name,
"%s%smpinter.%s",run_dir,delim,time_stamp);*/
         if (g_struct->c_l_opt->update == 1)
          sprintf(file_name,
"%s%smpqinter.%s",run_dir,delim,time_stamp);
         else
          sprintf(file_name,
"%s%smpufinter.%s",run_dir,delim,time_stamp);
         break;
       default:
         if (g_struct->c_l_opt->intStreamNum > 0)
           sprintf(file_name,
                 "%s%smts%dinter.%s",
                 run_dir,delim,g_struct->c_l_opt-
>intStreamNum,time_stamp);
         else
           fprintf(stderr,"Invalid stream number specified\n");
         break;
      }

  strcpy(outstreamfilename, file_name); /* wlc 081397 */

  if (!feof(instream) || g_struct->c_flags->eo_infile)
    /* Only create an output file if there are input
       statements left to process, or if we're all done
       and want to print out the summary table file */
    if( (outstream = fopen(file_name, WRITEMODE)) == NULL ) {
      fprintf(stderr,"\nThe output file could not be opened.  ");
      fprintf(stderr,"Make sure that the filename is correct.\n");
      fprintf(stderr,"filename = %s\n",file_name);
      exit(-1);
    }

  return;
}
```

```
/*****************************************************************/
/* Determine whether or not we should break out of the block loop
   because of an end of file, end of block, or update function.
   Also handle some semaphore stuff for update functions         */
/*****************************************************************/
int PreSQLprocess(struct global_struct *g_struct, Timer_struct
*start_time)
{
  int               rc = 1;
  FILE               *updateFP;
#ifndef SQLWINT
  int               semid;              /* semaphore for controlling UFs*/
  key_t              semkey;             /* key to generate semid */
#else
  int               SemTimeout = 600000;    /* Des time out period of 1
minute */
#endif

  switch (g_struct->c_flags->select_status)
  {
   case TPCDBATCH_NONSQL:
     g_struct->s_info_stop_ptr = g_struct->s_info_ptr;
     /* if we're at the end of the input file, set the stop
        pointer to this structure */
     rc = FALSE;
     break;
   case TPCDBATCH_EOBLOCK:
     rc = FALSE;
     break;
   case TPCDBATCH_INSERT:
     /* we have to check whether or not this is a throughput */
     /* test, and if it is, we have to set up a semaphore to */
     /* control when the update functions are run.  We want  */
     /* them to be run after all the query streams have finished. */
     /* What we do is set up the semaphore here, decrement it */
     /* in the query streams, and wait for it to get cleared */
     /* before we allow the UFs to run.                      */
     /* Note: we only set up the semaphore if:               */
     /*       1. we are running the throughput test (num of */
     /*          streams > 0)                               */
     /*       2. we are at the first UF1 (i.e. this is the  */
     /*          case where currentUpdatePair = updatePairStart */
     /* we also want to check the sem_on element in the global */
     /* structure to see if we want to use semaphores or let */
     /* the calling script do the synchronization of the update */
     /* stream                                               */
     if ( semcontrol == 1 )
     {
       /* yes we are to be using semaphores */
       /* is this the 1st time into update function 1 (uf1)? */
       if (currentUpdatePair == updatePairStart )
       {
             /* create the semaphores */
             create_semaphores(g_struct);
             if (g_struct->c_l_opt->intStreamNum != 0)
             /* wait period for runthroughput updates */
                   throughput_wait(g_struct);
       }
       /* otherwise continue to run*/
     }
     if ((g_struct->c_l_opt->update == 3) || (g_struct->c_l_opt->update
== 4))
     {
      get_start_time(start_time);
      strcpy(g_struct->s_info_ptr->start_stamp,
           get_time_stamp(T_STAMP_FORM_3,start_time ));  /*
TIME_ACC jen*/
```

```
      /* write the start timestamp to the file...if this is not a qualification
*/
      /* run, then write the seed used as well */
      fprintf( outstream,"Start timestamp %*.*s \n",
            T_STAMP_3LEN,T_STAMP_3LEN,                /*
TIME_ACC jen*/
            g_struct->s_info_ptr->start_stamp);
      if (g_struct->c_l_opt->intStreamNum >= 0)
      {
        if (g_struct->lSeed == -1)
        {
          fprintf( outstream,"Using default qgen seed file");
        }
        else
          fprintf( outstream,"Seed used = %ld",g_struct->lSeed);
        fprintf( outstream,"\n");
      }
    }
    if (g_struct->c_l_opt->update < 4){
    /* run only if updates are enabled */
      runUF1(g_struct, currentUpdatePair);
    }

    rc = FALSE;
    if ((g_struct->c_l_opt->intStreamNum == 0) && (semcontrol == 1))
    /* RUNPOWER: release first semaphore so the queries can run */
      release_semaphore(g_struct, INSERT_POWER_SEM);
    break;
  case TPCDBATCH_DELETE:
    if ((g_struct->c_l_opt->intStreamNum == 0) && (semcontrol == 1))
    {
    /* RUNPOWER: wait for queries to finish */
    /* waiting on QUERY_POWER_SEM semaphore */
      runpower_wait(g_struct, QUERY_POWER_SEM);
    }
    if ((g_struct->c_l_opt->update == 3) || (g_struct->c_l_opt->update
== 4))
    {
      get_start_time(start_time);
      strcpy(g_struct->s_info_ptr->start_stamp,
           get_time_stamp(T_STAMP_FORM_3,start_time ));  /*
TIME_ACC jen*/
      /* write the start timestamp to the file...if this is not a qualification
*/
      /* run, then write the seed used as well */
      fprintf( outstream,"Start timestamp %*.*s \n",
            T_STAMP_3LEN,T_STAMP_3LEN,                /*
TIME_ACC jen*/
            g_struct->s_info_ptr->start_stamp);
      if (g_struct->c_l_opt->intStreamNum >= 0)
      {
        if (g_struct->lSeed == -1)
        {
          fprintf( outstream,"Using default qgen seed file");
        }
        else
          fprintf( outstream,"Seed used = %ld",g_struct->lSeed);
        fprintf( outstream,"\n");
      }
    }
    if (g_struct->c_l_opt->update < 4){
    /* run only if updates are enabled */
      runUF2(g_struct, currentUpdatePair);
      if (g_struct->c_l_opt->intStreamNum == 0)
      {/* RUNPOWER */
        fprintf(stderr, "UF2 completed\n");
      }
    }
    currentUpdatePair += 1;
    /* update the update.pair.num file to reflect the successfully com-
pleted */
    /* update pair */
    if (g_struct->c_l_opt->update < 4)
    {   /*jen*/
#ifndef NO_INCREMENT
      /* don't update the pair, only for my testing - Haider */
      updateFP = fopen(g_struct->update_num_file,"w");
      fprintf(updateFP,"%d\n",currentUpdatePair);
      fclose(updateFP);

#endif
    }  /*jen*/
    rc = FALSE;
    break;

  }
  return(rc);
}


/*****************************************************************/
/* Handles actual processing of SQL statement.  Initializes the SQLDA
  for returned rows, does PREPARE, DECLARE, and OPEN state-
ments and
  executed multiple FETCHes as needed.  If not a SELECT statement,
  goes into EXECUTE IMMEDIATE section                          */
/*****************************************************************/
void SQLprocess(struct global_struct *g_struct)
{
  int rc = 0;                        /* 912RETRY */
  int rows_fetch = 0;
  long sqlcode = SQL_RC_E911;            /* Temporary sqlcode to
test
                                     for deadlocks */
  int max_wait = 1;                       /* Maximum number of retries
                                     for deadlock scenario */

  int col_lengths[TPCDBATCH_MAX_COLS];     /* array containing
widths of

                                     columns in returned set  */
  struct stmt_info *s_info_ptr;

  s_info_ptr = g_struct->s_info_ptr;
/*****************************************************************/
/* grab storage for the SQLDA                                    */
/*****************************************************************/
  if ((sqlda=(struct sqlda *)malloc(SQLDASIZE(100))) == NULL)
    mem_error("allocating sqlda");

  sqlda->sqln = TPCDBATCH_MAX_COLS;                    /*
@d30369 tjg */


  /* Error-recovery code for errors resulting from multi-stream errors */

  while (((sqlcode == SQL_RC_E911) ||
        (sqlcode == SQL_RC_E912) ||
        (sqlcode == SQL_RC_E901)) &&
       (max_wait < MAXWAIT) &&
       (rc==0) )
  {

    sqlcode = 0;          /* Re-initialize sqlcode to avoid infinite-loop */
    if (g_struct->c_flags->select_status == TPCDBATCH_SELECT)
    {
      /* Enter this loop if SQL stmt is a SELECT   */
      EXEC SQL PREPARE STMT1 INTO :*sqlda FROM :stmt_str;
```

```
        sqlcode = error_check();
        if (sqlcode < 0)
        {
          fprintf (stderr,"\nPrepare failed. Stopping this query.\n");
          rc = -1;
        }
        else  /* print out the column headings for the answer set */
        {
          print_headings(sqlda,col_lengths);              /* @d22817 tjg */

          allocate_sqlda(sqlda);      /* This is where we set storage for
the */
                                      /* SQLDA based on the column types in   */
                                      /* the answer set table.                */

          EXEC SQL DECLARE DYNCUR CURSOR FOR STMT1;

          EXEC SQL OPEN DYNCUR;
          sqlcode = error_check();

          if (sqlcode < 0)       /* we ran into an error of some kind KBS
98/09/28 */
          {
            max_wait ++;
            fprintf (stderr, "\nAn error has been detected on
open...Retrying...\n");
            SleepSome(10);
          }
          else
          {

/*********************************************************************/
            /* Fetch appropriate number of rows and determine whether
or not to    */
            /* send them to file.                                    */

/*********************************************************************/

            rows_fetch = 0;

            do
            {
              /* Keep fetching as long as we haven't finished reading
              all the rows and we haven't gone past the limits set
              in the control string */

              EXEC SQL FETCH DYNCUR USING DESCRIPTOR
:*sqlda;
              if (sqlca.sqlcode == 100)
              {
                sqlcode = sqlca.sqlcode;
              }
              else
              {
                sqlcode = error_check();
              }
              if (sqlcode == 0)
              {
                rows_fetch++;
                if ( (rows_fetch <= s_info_ptr->max_rows_out) ||
                     (s_info_ptr->max_rows_out == -1) )
                  echo_sqlda(sqlda,col_lengths);
              }
              else if (sqlcode < 0)
              {
                max_wait++;
                fprintf (stderr, "\nAn error has been detected on
fetch...Retrying...\n");
                SleepSome(10);
              }
            }  while ( (sqlcode == 0) && \
                     ( (s_info_ptr->max_rows_fetch == -1) || \
                       (rows_fetch < s_info_ptr->max_rows_fetch) ) );
          } /* end of successful open */
        } /* end of successful prepare */
      } /** End of block for handling SELECT statements **/

      else
      {         /** SQL statement is not a SELECT **/
        EXEC SQL EXECUTE IMMEDIATE :stmt_str;
        sqlcode = error_check();

        if (sqlcode < 0 )
        {
          max_wait ++;
          fprintf (stderr, "\nAn error has been detected on execute im-
mediate...Retrying...\n");
          SleepSome(10);
        }
      } /* end of block for handling NON-select statements */

      if ( (sqlcode >= 0 ) &&
         (g_struct->c_flags->select_status == TPCDBATCH_SELECT))
      {
        /* we opened a cursor before */
        EXEC SQL CLOSE DYNCUR;
        sqlcode = error_check();

        if ((s_info_ptr->max_rows_fetch == -1) ||
           (rows_fetch < s_info_ptr->max_rows_fetch))
#ifndef SQLPTX
          fprintf (outstream, "\n\nNumber of rows retrieved is: %6d",
                rows_fetch);
        else
          fprintf (outstream, "\n\nNumber of rows retrieved is: %6d",
                s_info_ptr->max_rows_fetch);
#else
          fprintf (outstream, "\n\nNumber of rows retrieved is: %6d",
                rows_fetch);
        else
          fprintf (outstream, "\n\nNumber of rows retrieved is: %6d",
                s_info_ptr->max_rows_fetch);
#endif
      }                                           /* @d28763 tjg */

      if (s_info_ptr->query_block == FALSE)  /* if block is off don't loop */
        g_struct->c_flags->eo_block = TRUE;

    } /* end of while loop to retry if needed */

} /* end of SQLprocess */

/*********************************************************************/
/* performs some operations after a statement has been processed,
   including doing a COMMIT if necessary, and calculating the
   elapsed time.  Also initializes a new stmt_info structure
   for the next block of statements                                  */
/*********************************************************************/
int PostSQLprocess(struct global_struct *g_struct, Timer_struct
*start_time)
{
  struct stmt_info *s_info_ptr;
  Timer_struct        end_t;         /* end point for elapsed time */

#if DEBUG
  fprintf (outstream, "In PostSQLprocess\n");
#endif

  s_info_ptr = g_struct->s_info_ptr;
```

```
  if (g_struct->c_flags->select_status == TPCDBATCH_NONSQL)
    return FALSE;  /* get out if we've reached the end of input file */

  if (g_struct->c_l_opt->update > 1)
  {
    /* This is an update function stream.  There is no need to
COMMIT.  */
    /* Each UF child will COMMIT its own transactions. */
    ;
  }
  else
  {  /* For non-UF cases, COMMIT now. */
   if (g_struct->c_l_opt->a_commit) {
     EXEC SQL COMMIT WORK;
     error_check();                              /* @d22275 tjg */
   }
  }

  fflush(outstream);

  s_info_ptr->elapse_time = get_elapsed_time(start_time);

  if (g_struct->c_flags->time_stamp == TRUE)          /* @d25594
tjg */
    get_start_time(&end_t); /* Get the end time */
    strcpy(s_info_ptr->end_stamp,
    get_time_stamp(T_STAMP_FORM_3,&end_t) );
    /*get_time_stamp(T_STAMP_FORM_3,(time_t)NULL) );*/

  /* BBE: Pass on time stamp values for the next query */
  temp_time_struct = end_t;
  strcpy(temp_time_stamp, s_info_ptr->end_stamp);

  /* write the start timestamp to the file */
  fprintf( outstream,"\n\nStop timestamp %*.*s \n",
        T_STAMP_3LEN,T_STAMP_3LEN,  /* TIME_ACC jen*/
        s_info_ptr->end_stamp);

  /* DJD print elapsed time in seconds  */
  fprintf( outstream,"Query Time = %15.1f secs\n", s_info_ptr-
>elapse_time);


  /** Allocate space for a new stmt_info structure **/    /* @d24993 tjg
*/
  s_info_ptr->next =
    (struct stmt_info *) malloc(sizeof(struct stmt_info));
  if (s_info_ptr->next != NULL)  {
    memset(s_info_ptr->next, '\0', sizeof(struct stmt_info));
    /** Transfer details from one structure to another for
      to apply for the next statement **/
    s_info_ptr->next->stmt_num = s_info_ptr->stmt_num + 1;
    s_info_ptr->next->max_rows_fetch = s_info_ptr->max_rows_fetch;
    s_info_ptr->next->max_rows_out = s_info_ptr->max_rows_out;

    s_info_ptr->next->query_block = s_info_ptr->query_block;
    s_info_ptr->next->elapse_time = -1;

    s_info_ptr = s_info_ptr->next;

  }
  else {
    mem_error("allocating next stmt structure. Exiting\n");
    exit(-1);
  }

  /** Set the stop and travelling pointer to the current info structure **/
  g_struct->s_info_stop_ptr = g_struct->s_info_ptr = s_info_ptr;

  printf("In PostSQLProcess\n");
  if (sqlda_allocated)
    free_sqlda(sqlda,g_struct->c_flags->select_status);
    /* fix free() problem on NT
      wlc 090597 */

  if (g_struct->c_l_opt->outfile != 0)
    fclose(outstream);

  return (TRUE);
}

/*****************************************************************************/
/* Does some cleaning up once all the statements are processed.  Dis-
connects
  from the database, cleans up some semaphore stuff from the update
functions,
  prints out the summary table, and closes all file handles.              */
/*****************************************************************************/
int cleanup(struct global_struct *g_struct)
{
#ifndef SQLWINT
  int              semid;          /* semaphore for controlling UFs*/
  key_t            semkey;         /* key to generate semid */
#endif
  char file_name[256] = "\0";

  /** End timestamp for stream **/
  /*g_struct->stream_end_time = time(NULL);*/
  get_start_time(&(g_struct->stream_end_time));  /* TIME_ACC jen */

  switch (g_struct->c_l_opt->update)
  {
      case (2):
      case (5):
            /* update throughput function stream */
            sprintf(file_name,"%s%sstrcntuf.%s",g_struct->run_dir,
                  env_tpcd_path_delim, g_struct->file_time_stamp);
            break;
      case (3):
      case (4):
            /* update power function stream */
            sprintf(file_name,"%s%spstrcntuf.%s",g_struct->run_dir,
                  env_tpcd_path_delim, g_struct->file_time_stamp);
            break;
      case (1):
            /* power query stream */
            sprintf(file_name, "%s%spstrcnt%d.%s",g_struct->run_dir,
env_tpcd_path_delim,
                  g_struct->c_l_opt->intStreamNum,g_struct-
>file_time_stamp);
            break;
      case (0):
            /* throughput query stream */
            sprintf(file_name, "%s%sstrcnt%d.%s",g_struct->run_dir,
env_tpcd_path_delim,
                  g_struct->c_l_opt->intStreamNum,g_struct-
>file_time_stamp);
            break;
  }

#ifndef LINUX

  if( (g_struct->stream_report_file = fopen(file_name,
APPENDMODE)) == NULL )
  {
    fprintf(stderr,"\nThe output file for the stream count information\n");
    fprintf(stderr,"could not be opened, make sure the filename is cor-
rect\n");
```

```
    fprintf(stderr,"filename = %s\n",file_name);
    exit(-1);
  }

#endif

  /* print out the stream stop time in the stream count information file*/
  if (g_struct->c_l_opt->update > 1)
  {
    /* update function stream */
    fprintf(g_struct->stream_report_file,
          "Update function stream stopping at %*.*s\n",
          T_STAMP_3LEN,T_STAMP_3LEN,  /* TIME_ACC jen*/
          get_time_stamp(T_STAMP_FORM_3,&(g_struct-
>stream_end_time)));  /* TIME_ACC jen*/
  }
  else
  {
    /* query stream(s) */
    fprintf(g_struct->stream_report_file,
          "Stream number %d stopping at %*.*s\n",
          g_struct->c_l_opt->intStreamNum,
          T_STAMP_3LEN,T_STAMP_3LEN,  /* TIME_ACC jen*/
          get_time_stamp(T_STAMP_FORM_3,&(g_struct-
>stream_end_time)));  /* TIME_ACC jen*/
  }
  fclose(g_struct->stream_report_file);



  /* No need to check for errors here.
    Also, the UF stream in a Throughput run
    has no connection in tpcdbatch.sqc.            aph 98/12/26
  error_check();
  */

  /* if we are in a query stream AND this is a throughput test, then
need */
  /* do to some semaphore stuff   (0 implies update functions are off)
*/
  /* AND we are supposed to be using semaphores */

  if ( ( semcontrol == 1 ) &&
       ( g_struct->c_l_opt->update < 2))
    /* only queries need to release the semaphore at this point */
  {
    if (g_struct->c_l_opt->intStreamNum == 0)
      release_semaphore(g_struct, QUERY_POWER_SEM); /* power
stream */
    else
      release_semaphore(g_struct, THROUGHPUT_SEM); /* through-
put stream */

  EXEC SQL CONNECT RESET;
#ifndef SQLWINT
    if (verbose)
    {
      fprintf(stderr,
            "cleanup: semkey = %ld, semid = %d, file = %s, stream =
%d\n",
            semkey,semid,g_struct->update_num_file,
            g_struct->c_l_opt->intStreamNum);
    }
#endif

  }


  /** Summary table processing **/                    /* @d24993 tjg */
  summary_table(g_struct);

  fprintf (outstream, "\n\n");

  fclose(outstream);          /* Close the output data stream.     */
  fclose(instream);           /* Close the SQL input stream.       */

  return (TRUE);
}

void create_semaphores(struct global_struct *g_struct)
{

#ifndef SQLWINT
  int             semid;            /* semaphore for controlling UFs*/
  key_t             semkey;           /* key to generate semid */
#else
  HANDLE            hSem;
  HANDLE            hSem2;
  int             SemTimeout = 600000;    /* Des time out period of 1
minute */
#endif
      fprintf(stderr,"numstreams = %d\n",g_struct->c_l_opt-
>intStreamNum);
      fprintf(stderr,"Update stream creating semaphore(s) for update
and query sequencing\n");
#ifdef SQLWINT

      fprintf(stderr,"semfile = %s\n",g_struct->sem_file);
      if (g_struct->c_l_opt->intStreamNum == 0)
      /*RUNPOWER*/
      {
            fprintf(stderr,"semfile2 = %s\n",g_struct->sem_file2);
            hSem = CreateSemaphore(NULL, 0,1,(LPCTSTR)(g_struct-
>sem_file));
            hSem2 = CreateSemaphore(NULL,
0,1,(LPCTSTR)(g_struct->sem_file2));
            if ((hSem == NULL) || (hSem2 == NULL))
            {
              fprintf(stderr,
                    "CreateSemaphores (ready semaphore) failed, Get-
LastError: %d, quitting\n",
                    GetLastError());
              exit(-1);
            }
            fprintf(stderr,"Semaphores created successfully!\n");
      }
      else
      {
      /* RUNTHROUGHPUT creates semaphores based on the num-
ber of query streams while the number of streams for runpower is con-
stant */
            hSem = CreateSemaphore(NULL, 0,
                        g_struct->c_l_opt->intStreamNum,
                        (LPCTSTR)(g_struct->sem_file));

            if (hSem == NULL)
            {
                fprintf(stderr,
                    "CreateSemaphore (ready semaphore) failed, Get-
LastError: %d, quitting\n",
                      GetLastError());
                exit(-1);
            }
            fprintf(stderr,"Semaphore created successfully!\n");

      }
#else             /* AIX, SUN, etc. */
      /* create a semaphore key...use the name of a file that */
      /* you know exists */
      fprintf(stderr,"semfile = %s\n", g_struct->update_num_file);
```

```
        semkey = ftok(g_struct->update_num_file,'J');
        if (g_struct->c_l_opt->intStreamNum == 0)
        /* RUNPOWER */
        {

                if ( (semid = sem-
get(semkey,2,IPC_CREAT|S_IRUSR|S_IWUSR)) < 0)
                {
                        fprintf(stderr,
                                "Throughput can't get initial semaphore! semget
failed errno = %d\n",
                                errno);
                        exit(1);
                }
        }
        else
        /* THROUGHPUT */
        {

                if ( (semid = sem-
get(semkey,1,IPC_CREAT|S_IRUSR|S_IWUSR)) < 0)
                {
                        fprintf(stderr,

                                "Throughput can't get initial semaphore! semget
failed errno = %d\n",
                                errno);
                        exit(1);
                }
                if (verbose)
                {
                        fprintf(stderr,
                                "insert: semkey = %ld, semid = %d, file = %s,
value = %d\n",
                                semkey,semid,g_struct->update_num_file,
                                (g_struct->c_l_opt->intStreamNum * -1));
                }
        }


#endif
}

/*throughput update */
void throughput_wait(struct global_struct *g_struct)
{
#ifndef SQLWINT
  int           semid;          /* semaphore for controlling UFs*/
  key_t         semkey;         /* key to generate semid */
#else
  HANDLE        hSem;
  int           j;
  int           SemTimeout = 600000;    /* Des time out period of 1
minute */
#endif

#ifdef SQLWINT
        hSem = open_semaphore(g_struct, THROUGHPUT_SEM);
        for (j = 0; j < g_struct->c_l_opt->intStreamNum; j++)
        {
                if (verbose)
                  fprintf(stderr,"About to wait again ...\n");
                if (WaitForSingleObject(hSem, INFINITE) ==
WAIT_FAILED)
                {
                        fprintf(stderr,
                            "WaitForSingleObject (hSem) failed on stream %d,
error: %d, quitting\n",
                              j, GetLastError());
                        exit(-1);
                }
                if (verbose)
                        fprintf(stderr,"Streams to wait for  %d\n", j);
        }
        fprintf(stderr,"finished waiting on stream semaphore! Ready to
run updates!\n");
        /* close the semaphore handle */
        if (! CloseHandle(hSem)) {
          fprintf(stderr, "Close Sem failed - Last Error: %d\n", GetLastEr-
ror());
          /* no exit here */
        }
#else
        semid = open_semaphore(g_struct);
        /* call the sem_op routine to decrement the semaphore by */
        /* however many streams .... by calling this function with*/
        /* a negative number, this stream is forced to wait until */
        /* the semaphore gets back to 0 */
        if (sem_op(semid, 0, (g_struct->c_l_opt->intStreamNum * -1)) !=
0)
        {                                               /*jenSEM*/
                fprintf(stderr,
                        "Failure to wait on throughput semaphone for %d
streams\n",
                        g_struct->c_l_opt->intStreamNum);
                exit(1);
        }                                               /*jenSEM*/
        fprintf(stderr,"finished waiting on stream semaphore! Ready to
run updates!\n");
        semctl(semid,0,IPC_RMID,0);  /* we've finished waiting, now */
                                /* remove the semaphore */
#endif
}

void runpower_wait(struct global_struct *g_struct, int sem_num)
{
   char semfile[150];
#ifdef SQLWINT
   HANDLE hSem;


   if (sem_num == 1)
      strcpy (semfile, g_struct->sem_file);
   else
      strcpy (semfile, g_struct->sem_file2);


#else    /* AIX */
   int              semid;          /* semaphore for controlling UFs*/
   key_t            semkey;         /* key to generate semid */

   strcpy (semfile, g_struct->update_num_file);

#endif

  if (g_struct->c_l_opt->update == 1)
    fprintf(stderr,"querystream waiting for update stream (UF1) to signal
semaphore based on %s\n", semfile);
  else
    fprintf(stderr,"updatestream (UF2) waiting on querystream sema-
phore to signal semaphore based on %s\n", semfile);

#ifdef SQLWINT

   hSem = open_semaphore(g_struct, sem_num);
   if (verbose)
     fprintf(stderr,"Runpower queries about to wait ...\n");
   if (WaitForSingleObject(hSem, INFINITE) == WAIT_FAILED)
   {
     fprintf(stderr,
```

```
        "WaitForSingleObject (hSem) failed on stream 0, error: %d,
quitting\n",
        GetLastError());
        exit(-1);
    }
    if (! CloseHandle(hSem))
    {
          fprintf(stderr, "Close Sem failed - Last Error: %d\n", GetLas-
tError());
          /* no exit here */
    }

#else

    semid = open_semaphore(g_struct);

    /* call the sem_op routine to decrement the semaphore by */
    /* however many streams .... by calling this function with*/
    /* a negative number, this stream is forced to wait until */
    /* the semaphore gets back to 0 */
    /* aix semaphores start at 0, not 1, so sem_num -1 is used */
    if (sem_op(semid, sem_num - 1, -1) != 0)
    {                                          /*jenSEM*/
      fprintf(stderr,
            "Failure to wait on runpower semaphone for %d streams\n",
            g_struct->c_l_opt->intStreamNum);
      exit(1);
    }                                          /*jenSEM*/
#endif
    if (g_struct->c_l_opt->update == 1)
     fprintf(stderr,"querystream finished waiting on updatestream sema-
phore\n");
    else
     fprintf(stderr,"updatestream finished waiting on querystream sema-
phore\n");
}

void release_semaphore(struct global_struct *g_struct, int sem_num)
{
#ifndef SQLWINT
   int              semid;          /* semaphore for controlling UFs*/
   key_t            semkey;         /* key to generate semid */
#else
   HANDLE           hSem;
   int              SemTimeout = 600000;    /* Des time out period of 1
minute */
#endif

#ifdef SQLWINT
      hSem = open_semaphore(g_struct, sem_num); /* query */
      if (! ReleaseSemaphore(hSem,
                        1,
                        (LPLONG)(NULL)))
      {
          fprintf(stderr, "ReleaseSemaphore failed, Sem#: %d LastEr-
ror: %d, quit\n",
                sem_num, GetLastError());
          exit(-1);
      }
#else
      semid = open_semaphore(g_struct); /* query */
      /* aix semaphores start at 0, not 1, so sem_num -1 is used */
      if (sem_op(semid, sem_num - 1, 1) != 0)
/*jenSEM*/
       {                                       /*jenSEM*/
            fprintf(stderr,
                "Failed to increment semaphore %d for throughput
stream %d\n",
                sem_num, g_struct->c_l_opt->intStreamNum);
            fprintf(stderr,
                "file for generation of semaphore is: %s\n",
                g_struct->update_num_file);
            exit(1);
       }

#endif
      if (g_struct->c_l_opt->intStreamNum == 0)
      {  /* RUNPOWER */
        if (sem_num == 1)
        {
          fprintf(stderr, "UF1 completed.\n");
        }
        else
        {
          fprintf(stderr, "query stream completed.\n");
        }
      }
}

#ifdef SQLWINT /* Compile only in NT */
HANDLE open_semaphore(struct global_struct *g_struct, int num)
{
      HANDLE  hSem;
      LPCTSTR semfile;

      if (num == 1)
            semfile = (LPCTSTR)g_struct->sem_file;
      else
            semfile = (LPCTSTR)g_struct->sem_file2;

      while ((hSem = OpenSemaphore(SEMAPHORE_ALL_ACCESS |
                              SEMAPHORE_MODIFY_STATE |
                              SYNCHRONIZE,
                             TRUE,
                             semfile))
                  == (HANDLE)(NULL))
      {
        /*
        ** if cannot open the semaphore, wait for 0.1 second
        */
        fprintf(stderr,"Retry Open semaphore %s\n",semfile);

        Sleep(1000);
      }
      return hSem;
}

#else  /* Compile only in non-NT (i.e. AIX) */
int open_semaphore(struct global_struct *g_struct)
{
      int               semid;          /* semaphore for controlling UFs*/
      key_t             semkey;         /* key to generate semid */
      int num;

      if (g_struct->c_l_opt->intStreamNum == 0)
            num = 2;
      else
            num = 1;

      semkey = ftok(g_struct->update_num_file,'J');
      while ((semid = semget(semkey,num,0)) < 0)
      {
            if (errno == ENOENT)
            {
                  sleep(2);
                  fprintf(stderr,"cleanUp: looping for access to sema-
phore stream %d  ",
                        g_struct->c_l_opt->intStreamNum);
                  fprintf(stderr,"semkey=%ld semid = %d
file=%s\n",semkey,semid,
```

```
                        g_struct->update_num_file);
            }
            else
            {
                fprintf(stderr,"query stream %d semget failed errno =
%d\n",
                        g_struct->c_l_opt->intStreamNum,errno);
                exit(1);
            }
        }
        return semid;
}
#endif
```

D.3 tpcdUF.sqc

```
#define UF1DEBUG
#define UF2DEBUG

#if (defined(SQLPTX) && defined(SQLSUN))
#define exit(rc) _exit(rc)
#else
#define exit(rc) exit(rc)
#endif /* SQLPTX & SQLSUN*/


#include "tpcdbatch.h"
/** EXEC SQL INCLUDE SQLCA; **/

#include "sqlca.h"
extern struct sqlca sqlca;

/*****************************************************************/
/* Function Prototypes                                           */
/*****************************************************************/
extern int SleepSome( int amount );
extern long error_check(void);                          /* @d28763 tjg
*/
extern void dumpCa(struct sqlca*);      /*kmw*/
extern int sem_op (int semid, int semnum, int value);
extern char *get_time_stamp(int form, Timer_struct *timer_pointer);
/* TIME_ACC jen */

/*****************************************************************/
/* Declare the SQL host variables.                               */
/*****************************************************************/
EXEC SQL BEGIN DECLARE SECTION;
  char UF_dbname[9] = "\0";
  char UF_userid[9] = "\0";
  char UF_passwd[9] = "\0";
  sqlint32 UF_chunk = 0;
  short month = 0;
EXEC SQL END DECLARE SECTION;

/*****************************************************************/
/* Declare the global variables.                                 */
/*****************************************************************/
extern char env_tpcd_tmp_dir[150];
extern FILE *instream, *outstream;  /* File pointers */
extern char sourcefile[256];    /* Used for semaphores and table func-
tions?*/
extern struct {                         /* jen LONG */
            short len;
            char data[32700];
          } stmt_str;                   /* jen LONG */
```

```
/*****************************************************************/
/* UF1 child                                                     */
/* (i is the application number.)                                */
/*****************************************************************/
void runUF1_fn ( int updatePair, int i, char *dbname, char *userid, char
*passwd )
{
  int rc = 0;
  int split_updates = 2;       /* no. of ways update records are split */
  int concurrent_inserts = 2;  /* jenCI no of concurrent updates to be */
                               /* jenCI run at once*/
  int loop_updates = 1;        /* jenCI no of updates to be run in one  */
                               /* jenCI "concurrent" invocation.  should*/
                               /* jenCI be split_updates / concurrent_inserts*/
  int startChunk = 0;          /* jenCI number of first chunk to insert for */
                               /* jenCI this child */
  int stopChunk = 0;           /* jenCI number of last chunk to insert for */
                               /* jenCI this child */
  long insertedLineitem = 0;  /*kmw*/
  long insertedOrders = 0;    /*kmw*/
  long saveInsertedOrders = 0; /*kbs*/

  long sqlcode;
  int maxwait;

#ifndef SQLWINT
  int               su_semid;
  key_t             su_semkey;
#else
  HANDLE            su_hSem;
  char              UF1_semfile[256];
#endif

  char myoutstreamfile[256];
  FILE *myoutstream;

  strcpy(UF_dbname, dbname);
  strcpy(UF_userid, userid);
  strcpy(UF_passwd, passwd);

  /* Get ready to start logging diagnostic output */
  sprintf (myoutstreamfile, UF1OUTSTREAMPATTERN,
env_tpcd_tmp_dir, PATH_DELIM,
          updatePair, i);
  if ( (myoutstream = fopen (myoutstreamfile, WRITEMODE)) ==
NULL)
  {
    fprintf (stderr, "\nThe output file '%s' for update pair %d set %d
could not be opened. runUF1_fn\n",
             myoutstreamfile,updatePair,i);
    rc=-1;
    goto UF1_exit;
  }
  outstream=myoutstream;    /* initialize outstream for error_check
dxxxxhar*/

  fprintf( myoutstream,"\nUF1 for update pair %d set %d starting at
%*.*s\n",
          updatePair, i,
          T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
          get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));
/* TIME_ACC jen*/


  if (getenv ("TPCD_SPLIT_UPDATES") != NULL)
    split_updates = atoi (getenv ("TPCD_SPLIT_UPDATES"));
  if (getenv ("TPCD_CONCURRENT_INSERTS") != NULL)
/*jenCI*/
    concurrent_inserts = atoi (getenv
("TPCD_CONCURRENT_INSERTS"));  /*jenCI*/
```

```
  loop_updates = split_updates / concurrent_inserts;
/*jenCI*/

  /* determine the starting and stopping point of the chunks that this
jenCI*/
  /* invocation will apply. i is starting chunk number with range 0
jenCI*/
  /* through (concurrent_inserts -1)                              jenCI*/
  startChunk = i * loop_updates;                                /*jenCI*/
  stopChunk = startChunk + (loop_updates - 1);
/*jenCI*/


  /* Establish a connection to the database */
  if (!strcmp(userid,"\0"))   /** No authentication provided **/
    EXEC SQL CONNECT TO :UF_dbname;
  else
    EXEC SQL CONNECT TO :UF_dbname USER :UF_userid USING
:UF_passwd;
  error_check();
  if (sqlca.sqlcode < 0)
  {
    rc=-1;
    goto UF1_exit;
  }

  /* Start processing each chunk in my range */
#ifdef UF1DEBUG
  fprintf (myoutstream,"Before loop_a startChunk = %d, stopChunk =
%d\n", startChunk, stopChunk);
  fflush(myoutstream);
#endif
  for ( UF_chunk = startChunk; UF_chunk <= stopChunk;
UF_chunk++ )          /*jenCI*/
  {                                                          /*jenCI*/
    /* wlc  062797 */
    sqlcode = SQL_RC_E911;
    month = (short)UF_chunk;  /* Cast 'short' added bbe */
    maxwait = 1;
    rc = 0;

#ifdef UF1DEBUG
  fprintf (myoutstream, "Before While_a Chunk= %d \n",UF_chunk);
  fflush(myoutstream);
#endif
    /* Loop to handle any deadlocks */
    while (sqlcode == SQL_RC_E911 && maxwait <= MAXWAIT &&
rc==0)
    {

      sqlcode = 0;
#ifdef UF1DEBUG
  fprintf (myoutstream, "in loop before orders exec sql\n");
  fflush(myoutstream);
#endif
      EXEC SQL INSERT INTO TPCD.ORDERS
            SELECT
O_ORDERKEY,O_CUSTKEY,O_ORDERSTATUS,O_TOTALPRICE,

O_ORDERDATE,O_ORDERPRIORITY,O_CLERK,O_SHIPPRIORITY
,O_COMMENT
            FROM TPCDTEMP.ORDERS_NEW
            WHERE APP_ID = :UF_chunk;
            /*AND 12*(YEAR(O_ORDERDATE)-
1992)+MONTH(O_ORDERDATE)-01 = :month;*/

      if (sqlca.sqlcode < 0)
        sqlcode = error_check();

      if (sqlcode == SQL_RC_E911)
      {       /* we've hit a deadlock */
        fprintf (myoutstream,
              "\nDeadlock detected inserting from
tpcdtemp.orders_new for chunk %d for pair
%d..Retrying...\n",UF_chunk,updatePair);
        SleepSome(UF_DEADLOCK_SLEEP);
        maxwait++;                          /* jen DEADLOCK */
      }
      else if (sqlcode < 0)
      {
        fprintf(myoutstream,
            "Insert into orders pair %d chunk %d failed
sqlcode=%d\n",
              updatePair,UF_chunk,sqlcode);
        dumpCa(&sqlca);
        rc = -1;
      }
      else
      {
        /* Everything worked with ORDERS, proceed with LINEITEM */
        saveInsertedOrders = sqlca.sqlerrd[2];

        sqlcode = 0;
#ifdef UF1DEBUG
        fprintf (myoutstream, "in lineitem for update pair number %d
set %d chunk %d\n",
              updatePair, i,UF_chunk);
        fflush(myoutstream);
#endif

        EXEC SQL INSERT INTO TPCD.LINEITEM
          SELECT
L_ORDERKEY,L_PARTKEY,L_SUPPKEY,L_LINENUMBER,L_QUAN
TITY,
          L_EXTENDEDPRICE,L_DISCOUNT,L_TAX,

L_RETURNFLAG,L_LINESTATUS,L_SHIPDATE,L_COMMITDATE,L_
RECEIPTDATE,
          L_SHIPINSTRUCT,L_SHIPMODE,L_COMMENT
          FROM TPCDTEMP.LINEITEM_NEW WHERE APP_ID =
:UF_chunk;
          /*(AND L_ORDERKEY IN
            (SELECT O_ORDERKEY FROM TPCD.ORDERS
              WHERE 12*(YEAR(O_ORDERDATE)-
1992)+MONTH(O_ORDERDATE)-01 = :month);*/

        if (sqlca.sqlcode < 0)
          sqlcode = error_check();

        if (sqlcode == SQL_RC_E911)
        {       /* we've hit a deadlock */
          fprintf (myoutstream,
              "\nA deadlock has been detected inserting from
tpcdtemp.lineitem%d_%d...Retrying...\n",
              updatePair, UF_chunk);
          SleepSome(UF_DEADLOCK_SLEEP);
          maxwait++;                          /* jen DEADLOCK */
        }
        else if (sqlcode < 0)
        {
          fprintf(myoutstream,
            "Insert into lineitem pair %d chunk %d failed
sqlcode=%d\n",
              updatePair,UF_chunk,sqlcode);
          dumpCa(&sqlca);
          rc = -1;
        }
        else
        {
#ifdef UF1DEBUG
```

```
            fprintf (myoutstream, "lineitem insert succeeded\n");
            fflush(myoutstream);
#endif
            /* accumulate the number of row inserted */
            /* Order count ONLY updated if both orders and lineitem */
            /* go through */
            insertedOrders += saveInsertedOrders;               /* kbs */
            insertedLineitem += sqlca.sqlerrd[2];
            rc=0;
            EXEC SQL COMMIT WORK;
            error_check();

#ifdef UF1DEBUG
            /* report the number of row inserted */
            fprintf(myoutstream, "    interim %ld rows for chunk %d into
TPCD.ORDERS at %*.*s\n",
                    insertedOr-
ders,UF_chunk,T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC
jen*/
                    get_time_stamp(T_STAMP_FORM_1,(Timer_struct
*)NULL));  /* TIME_ACC jen*/
            /* report the number of row deleted *s inserted */
            fprintf(myoutstream,
                    "    interim %ld rows for chunk %d into TPCD.LINEITEM
at %*.*s\n",
                    insertedLineitem,UF_chunk,
                    T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
                    get_time_stamp(T_STAMP_FORM_1,
                            (Timer_struct *)NULL));  /* TIME_ACC jen*/

            fprintf( myoutstream,
                     "    inserts for update pair %d chunk %d complete at
%*.*s\n\n",
                     updatePair, UF_chunk,
                     T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC
jen*/
                     get_time_stamp(T_STAMP_FORM_1,
                             (Timer_struct *)NULL));  /* TIME_ACC jen*/

#endif
          }
        } /* process lineitem INSERTs */
      } /* while loop for deadlocks */
    } /* while processing chunks */

    /* report the number of row deleted */
    fprintf(myoutstream, "%ld rows inserted into TPCD.ORDERS at
%*.*s\n",
          insertedOrders,T_STAMP_1LEN,T_STAMP_1LEN,  /*
TIME_ACC jen*/
          get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));
/* TIME_ACC jen*/
    fprintf(myoutstream, "%ld rows inserted into TPCD.LINEITEM at
%*.*s\n",
          insertedLineitem,T_STAMP_1LEN,T_STAMP_1LEN,  /*
TIME_ACC jen*/
          get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));
/* TIME_ACC jen*/

    if (sqlcode < 0)
    {
      if (sqlcode == SQL_RC_E911)
      {
        fprintf (myoutstream,"# of deadlocks exceeds %i\n", MAXWAIT);
      }
      rc=-1;
      EXEC SQL ROLLBACK WORK;
      error_check();                /* @d22275 tjg */

      goto UF1_exit;
    }

/* UF1_conn_reset:  */
    EXEC SQL CONNECT RESET;
    error_check();                  /* @d22275 tjg */


UF1_exit:
    fclose (myoutstream);
    /* exiting, increment the semaphore */

    /* we used the first flat file to generate the semaphore key */

#ifndef SQLWINT
    /*    we will use the tpcd.setup file to generate the semaphore key
begin SEMA */
    if (getenv("TPCD_AUDIT_DIR") != NULL)
    {
      /* this is assuming that you will be running this from 0th node */
      sprintf(sourcefile, "%s%ctools%ctpcd.setup",
            getenv("TPCD_AUDIT_DIR"), PATH_DELIM,PATH_DELIM);
    }
    else
    {
      fprintf (stderr, "Can't open UF1 semaphore file TPCD_AUDIT_DIR
is not defined.\n");
      exit (-1);
    }
    /* end SEMA */

    su_semkey = ftok (sourcefile, 'J');
    while ( (su_semid = semget (su_semkey, 1, 0)) < 0)
    {
      if (errno == ENOENT) {
        sleep(2);
      }
      else {
        fprintf(stderr,"update set %d: semget failed errno = %d\n",
              i, errno);
        exit(1);
      }
    }
    if (sem_op (su_semid, 0, 1) != 0)                           /*jen SEM*/
    {
      fprintf(stderr,"Failure to increment semaphore UF1 set %d\n",i);
      fprintf(stderr,"  semaphore sourcefile = %s  su_semid =
su_semid\n",sourcefile);
      exit(1);
    }                                                   /*jenSEM*/

#else  /* SQLWINT */
    sprintf (UF1_semfile, "%s.%s.UF1.semfile",
          getenv("TPCD_DBNAME"), getenv("USER"));
    fprintf(stderr,"UF1 semfile = %s\n",UF1_semfile);
    while ((su_hSem = OpenSemaphore(SEMAPHORE_ALL_ACCESS |
                          SEMAPHORE_MODIFY_STATE |
                          SYNCHRONIZE,
                          TRUE,
                          UF1_semfile))
        == (HANDLE)(NULL))
    {
      /*
      ** if cannot open the semaphore, wait for 0.1 second
      */
      fprintf(stderr,"Retry Open semaphore %s\n", UF1_semfile);

      sleep(1);
    }

    if (! ReleaseSemaphore(su_hSem,
```

```
                    1,
                    (LPLONG)(NULL)))
  {
    fprintf(stderr, "ReleaseSemaphore failed, LastError: %d, quit\n",
          GetLastError());
    exit(-1);
  }
#endif /* SQLWINT */
  exit(rc);                    /* child exiting after finishing up */
}



/*****************************************************************/
/* UF2 child                                                     */
/*****************************************************************/
void runUF2_fn ( int updatePair, int thisConcurrentDelete, int num-
Chunks, char *dbname, char *userid, char *passwd )
{
  int rc = 0;
  long sqlcode;
  int maxwait;
  int startChunk = thisConcurrentDelete*numChunks; /* where do we
start? */
  long deletedLineitems = 0; /*kmw*/
  long deletedOrders = 0;    /*kmw*/
  long savedDeletedLineitems = 0; /*kbs*/

#ifndef SQLWINT
  int                su_semid; /* semaphore for controlling split updates*/
  key_t              su_semkey; /* key to generate semid */
#else
  HANDLE             su_hSem;
  char               UF2_semfile[256];
#endif

  char myoutstreamfile[256];
  FILE *myoutstream, *src_fh=NULL;

  strcpy(UF_dbname, dbname);
  strcpy(UF_userid, userid);
  strcpy(UF_passwd, passwd);


  /* Generate the unique filename for this concurrent delete process */
  sprintf (myoutstreamfile, UF2OUTSTREAMPATTERN,
env_tpcd_tmp_dir, PATH_DELIM,
         updatePair, thisConcurrentDelete);
  if ( (myoutstream = fopen (myoutstreamfile, WRITEMODE)) ==
NULL)
  {
    fprintf (stderr,
          "\nThe output file '%s' for update pair %d set %d could not
be opened runUF2_fn.\n",
          myoutstreamfile,updatePair,thisConcurrentDelete);
    rc=-1;
    goto UF2_exit;
  }

  outstream=myoutstream;    /* initialize outstream for error_check
dxxxxhar*/

#ifdef UF2DEBUG
  fprintf (myoutstream, "RunUF2 Called %d   %d   %d\n",
                    updatePair, thisConcurrentDelete, numChunks );
  fflush(myoutstream);
#endif
  fprintf( myoutstream,
         "\nUF2 for update pair %d set %d starting at %*.*s\n",
         updatePair, thisConcurrentDelete,
T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
         get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));
/* TIME_ACC jen*/

#ifdef UF2DEBUG
  fprintf (myoutstream, "before connect\n");
  fflush(myoutstream);
#endif

  if (!strcmp(userid,"\0"))   /** No authentication provided **/
    EXEC SQL CONNECT TO :UF_dbname;
  else
    EXEC SQL CONNECT TO :UF_dbname USER :UF_userid USING
:UF_passwd;
  error_check();


#ifdef UF2DEBUG
  fprintf (myoutstream, "after connect  startchunk= %d, EndChunk =
%d\n",
                          startChunk, startChunk+numChunks);
  fflush(myoutstream);
#endif

  /* Start processing each chunk in my range */
  for ( UF_chunk = startChunk; UF_chunk < startChunk+numChunks;
UF_chunk++ )
  {

    /* Set things up for the loop which will retry if there is a deadlock */
    sqlcode = SQL_RC_E911;
    month = (short)UF_chunk;
    maxwait = 1;
    rc = 0;

#ifdef UF2DEBUG
  fprintf (myoutstream, "Chunk = %d\n", UF_chunk);
  fflush(myoutstream);
#endif
    while (sqlcode == SQL_RC_E911 && maxwait <= MAXWAIT && rc
== 0)
    {

#ifdef UF2DEBUG
  fprintf (myoutstream, "in loop before orders exec sql\n");
  fflush(myoutstream);
#endif
      sqlcode = 0;

      EXEC SQL DELETE FROM TPCD.LINEITEM
            WHERE L_ORDERKEY IN
              (SELECT O_ORDERKEY FROM
TPCDTEMP.ORDERS_DEL
               WHERE APP_ID = :UF_chunk);
               /*AND O_ORDERKEY IN
                (SELECT O_ORDERKEY FROM TPCD.ORDERS
                 WHERE 12*(YEAR(O_ORDERDATE)-
1992)+MONTH(O_ORDERDATE)-01 = :month));*/
      if (sqlca.sqlcode < 0)
        sqlcode = error_check();

      if (sqlcode == SQL_RC_E911)
      {       /* we've hit a deadlock */
        fprintf (myoutstream,
          "\nA deadlock detected while deleting from LINEITEM: up-
date pair %d set %d chunk %d.  Retrying.\n",
          updatePair, thisConcurrentDelete, UF_chunk);
        dumpCa(&sqlca);
        SleepSome(UF_DEADLOCK_SLEEP);
```

```
        maxwait++;      /* jen DEADLOCK */
      }
      else if (sqlcode < 0)
      {
        fprintf (myoutstream,"\n%s\n", stmt_str.data);
        fprintf (myoutstream,"\nsqlcode %d occurred deleting from
TPCD.LINEITEM\n", sqlca.sqlcode);
        dumpCa(&sqlca);
        fprintf (myoutstream,
              "for update pair number %d set %d chunk %d..Exiting\n",
              updatePair, thisConcurrentDelete,UF_chunk);
        rc=-1;
      }
      else
      {
        /* accumulate the number of row deleted */
        savedDeletedLineitems = sqlca.sqlerrd[2];   /*kbs*/

#ifdef UF2DEBUG
        fprintf (myoutstream, "in loop for update pair number %d set
%d chunk %d\n",
              updatePair, thisConcurrentDelete,UF_chunk);
        fflush(myoutstream);
#endif

        /* delete the orders now */

        EXEC SQL DELETE FROM TPCD.ORDERS
        WHERE O_ORDERKEY IN
          (SELECT O_ORDERKEY FROM
TPCDTEMP.ORDERS_DEL WHERE APP_ID = :UF_chunk);
          /*AND 12*(YEAR(O_ORDERDATE)-
1992)+MONTH(O_ORDERDATE)-01 = :month;*/

        if (sqlca.sqlcode < 0)
          sqlcode = error_check();

        if (sqlcode == SQL_RC_E911)
        {         /* we've hit a deadlock */
#ifdef UF2DEBUG
  fprintf (myoutstream, "orders deadlocked\n");
  fflush(myoutstream);
#endif
          fprintf (myoutstream,
          "\nA deadlock detected while deleting from ORDERS: update
pair %d set %d chunk %d.  Retrying.\n",
          updatePair, thisConcurrentDelete, UF_chunk);
          dumpCa(&sqlca);
          SleepSome(UF_DEADLOCK_SLEEP);
          maxwait++;      /* jen DEADLOCK */
        }
        else if (sqlcode < 0)
        {
#ifdef UF2DEBUG
  fprintf (myoutstream, "orders failed\n");
  fflush(myoutstream);
#endif
          fprintf (myoutstream,"\nAn error %d occurred deleting from
TPCD.ORDERS\n",sqlca.sqlcode);
          dumpCa(&sqlca);
          fprintf (myoutstream,"for update pair number %d set %d
chunk %d..Exiting\n",
                updatePair, thisConcurrentDelete,UF_chunk);
          rc=-1;
        }
       else
        {
#ifdef UF2DEBUG
  fprintf (myoutstream, "orders succeeded\n");
  fflush(myoutstream);
#endif
          /* accumulate the number of row deleted */
          /* Order count ONLY updated if both orders and lineitem */
          /* go through */
          deletedLineitems += savedDeletedLineitems;   /* kbs */
          deletedOrders += sqlca.sqlerrd[2];
          rc=0;
          EXEC SQL COMMIT WORK;
          error_check();
#ifdef UF2DEBUG
          /* report the number of rows deleted */
          fprintf(myoutstream, "    interim %ld rows for chunk %d from
TPCD.ORDERS at %*.*s\n",
                deletedOr-
ders,UF_chunk,T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC
jen*/
                get_time_stamp(T_STAMP_FORM_1,(Timer_struct
*)NULL));  /* TIME_ACC jen*/
          fprintf(myoutstream, "    interim %ld rows for chunk %d from
TPCD.LINEITEM at %*.*s\n",
                delet-
edLineitems,UF_chunk,T_STAMP_1LEN,T_STAMP_1LEN,  /*
TIME_ACC jen*/
                get_time_stamp(T_STAMP_FORM_1,(Timer_struct
*)NULL));  /* TIME_ACC jen*/
          fprintf( myoutstream,
                "    deletes for update pair %d chunk %d complete at
%*.*s\n\n",
                updatePair, UF_chunk,
                T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC
jen*/
                get_time_stamp(T_STAMP_FORM_1,
                      (Timer_struct *)NULL));  /* TIME_ACC jen*/
#endif
        }
      } /* process orders  deletes */
    } /* while trying to delete one chunk loop */
  } /* while there are more chunks */

#ifdef UF2DEBUG
  fprintf (myoutstream, "after loop\n");
  fflush(myoutstream);
#endif
  /* report the number of row deleted */
  fprintf(myoutstream, "%ld rows deleted from TPCD.ORDERS at
%*.*s\n",
        deletedOrders,T_STAMP_1LEN,T_STAMP_1LEN,  /*
TIME_ACC jen*/
        get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));
/* TIME_ACC jen*/
  fprintf(myoutstream, "%ld rows deleted from TPCD.LINEITEM at
%*.*s\n",
        deletedLineitems,T_STAMP_1LEN,T_STAMP_1LEN,  /*
TIME_ACC jen*/
        get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));
/* TIME_ACC jen*/

  if (sqlca.sqlcode < 0)
  {
    fprintf (myoutstream,"# of deadlocks %d exceeds %i\n", max-
wait,MAXWAIT);
    rc=-1;
    EXEC SQL ROLLBACK WORK;
    error_check();                /* @d22275 tjg */
  }

/* UF2_conn_reset:  */                  /*971101jen*/
  EXEC SQL CONNECT RESET;
  error_check();                /* @d22275 tjg */
```

```
UF2_exit:
  fclose (myoutstream);

  /* exiting, increment the semaphore */
#ifndef SQLWINT
  /* we used the tpcd.setup file to generate the semaphore key      be-
gin SEMA */
  if (getenv("TPCD_AUDIT_DIR") != NULL)
  {
    sprintf(sourcefile, "%s%ctools%ctpcd.setup",
          getenv("TPCD_AUDIT_DIR"), PATH_DELIM, PATH_DELIM);
  }
  else
  {
    fprintf (stderr, "Can't open UF2 semaphore file TPCD_AUDIT_DIR
is not defined.\n");
    exit (-1);
  }

  su_semkey = ftok (sourcefile, 'D');   /* use D for deletes */
  /* end SEMA */
  while ((su_semid = semget(su_semkey,1,0)) < 0)
  {
    if (errno == ENOENT)
      sleep(2);
    else {
      fprintf(stderr,"UF2 update stream %d: semget failed errno =
%d\n",
            updatePair, errno);
      exit(1);
    }
  }
  if (sem_op (su_semid, 0, 1) != 0 )     /*jenSEM*/
  {                                      /*jenSEM*/
    fprintf(stderr,"Failure to increment semaphone UF2 set %d\n", this-
ConcurrentDelete);
    exit(1);
  }                                                  /*jenSEM*/

#else
  sprintf (UF2_semfile, "%s.%s.UF2.semfile",
        getenv("TPCD_DBNAME"), getenv("USER"));
  fprintf(stderr,"UF2 semfile = %s\n",UF2_semfile);
  while ((su_hSem = OpenSemaphore(SEMAPHORE_ALL_ACCESS |
                        SEMAPHORE_MODIFY_STATE |
                        SYNCHRONIZE,
                        TRUE,
                        UF2_semfile))
      == (HANDLE)(NULL)) {
    /*
    ** if cannot open the semaphore, wait for 0.1 second
    */
    fprintf(stderr,"Retry Open semaphore %s\n", UF2_semfile);

    SleepSome(1);
  }

  if (! ReleaseSemaphore(su_hSem,
                1,
                (LPLONG)(NULL)))
  {
    fprintf(stderr, "ReleaseSemaphore failed, LastError: %d, quit\n",
          GetLastError());
    exit(-1);
  }
#endif

  exit(rc);                 /* child exiting after finishing up */
}
```

D.4 Makefile

```
#########################################################
#############
# MAKEFILE for tpcdbatch program
# Enter the Following:
#
#     make tpcdbatch     -- makes tpcdbatch
#
#     make cleanup       -- removes builds from tpcdbatch program
#
#  NOTE:  You must have the TPCD_DBNAME environment variable
set or
#        this will not work, I'm trying to figure out a way to see
#        if it is set, and if not, to default to tpcd, but so far
#        no luck.
#########################################################
#############

#LOCAL=tpcd

BASE=$(HOME)/sqllib
COMPILE_FLAGS= -m64 -c -DSQLAIX -DLINUX -I$(BASE)/include -g
#COMPILE_FLAGS= -c -DSQLAIX -I$(BASE)/include -g
# if using an installed db2 image use the 2nd link_flags value
LINK_FLAGS= -m64 -o $@ -L$(BASE)/lib -ldb2
#LINK_FLAGS= -o $@ -L/usr/lpp/db2_05_00/lib -ldb2
COMPILER=cc
LIB_LINKER=ld
LIB_LINK_FLAGS= -o $@ -H512 -T512 -bE:$@.exp -L$(BASE)/lib -
ldb2 -lc

cleanup :
        rm -f tpcdbatch tpcdbatch.bnd tpcdbatch.o tpcdbatch.c
tpcdbatch.u tpcdUF.bnd tpcdUF.o tpcdUF.c tpcdUF.u 2>/dev/null

all : tpcdbatch


tpcdbatch.c : tpcdbatch.sqc
        @echo -e 'connect to tpcd \n prep tpcdbatch.sqc BINDFILE
PACKAGE ISOLATION RR BLOCKING ALL OPTLEVEL 1 DATETIME
ISO \n connect reset \n terminate \n' | db2 -c +p -v +t

tpcdUF.c : tpcdUF.sqc
        @echo -e 'connect to tpcd \n prep tpcdUF.sqc BINDFILE
PACKAGE ISOLATION RR BLOCKING ALL OPTLEVEL 1 DATETIME
ISO \n connect reset \n terminate \n' | db2 -c +p -v +t

tpcdbatch : tpcdUF.c tpcdbatch.c
          $(COMPILER) $(COMPILE_FLAGS) tpcdUF.c
          $(COMPILER) $(COMPILE_FLAGS) tpcdbatch.c
          $(COMPILER) $(LINK_FLAGS) tpcdUF.o tpcdbatch.o
```

D.5 Runpower

```
:   # -*-Perl-*-

eval 'exec perl5 -S $0 ${1+"$@"}' # Horrible kludge to convert this
    if 0;                  # into a "portable" perl script

# usage  runpower [UF]
# where UF is the optional parameter that says to run the power test
# with the update functions.  By default, the update functions are not
# run
```

```
push(@INC, split(':', $ENV{'PATH'}));

# Get TPC-D specific environment variables
require 'getvars';

# Use the macros in here so that they can handle the platform differences.
# macro.pl should be sourced from cmvc, other people wrote and maintain it.
require "macro.pl";
require "tpcdmacro.pl";

# Make output unbuffered.
select(STDOUT);
$| = 1 ;

if (@ARGV > 0)
{
    $runUF=$ARGV[0];
}
else
{
    $runUF="no";
}


if (length($ENV{"TPCD_AUDIT_DIR"}) <= 0)
{
  die "TPCD_AUDIT_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_RUN_DIR"}) <= 0)
{
  die "TPCD_RUN_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_DBNAME"}) <= 0)
{
  die "TPCD_DBNAME environment variable not set\n";
}
if (length($ENV{"TPCD_RUNNUMBER"}) <= 0)
{
  die "TPCD_RUNNUMBER environment variable not set\n";
}
if (length($ENV{"TPCD_SF"}) <= 0)
{
  die "TPCD_SF environment variable not set\n";
}
if (length($ENV{"TPCD_PLATFORM"}) <= 0)
{
  die "TPCD_PLATFORM environment variable not set\n";
}
if (length($ENV{"TPCD_PATH_DELIM"}) <= 0)
{
  die "TPCD_PATH_DELIM environment variable not set\n";
}
if (length($ENV{"TPCD_PRODUCT"}) <= 0)
{
  die "TPCD_PRODUCT environment variable not set\n";
}
if (length($ENV{"TPCD_AUDIT"}) <= 0)
{
    die "Must set TPCD_AUDIT env't var.  Real audit timing sequence run if yes\n";
}
if (length($ENV{"TPCD_PHYS_NODE"}) <= 0)
{
    die "TPCD_PHYS_NODE env't var not set\n";
}
if (length($ENV{"TPCD_LOG_DIR"}) <= 0)
{
    $ENV{"TPCD_LOG_DIR"} = "NULL";
}
if (length($ENV{"TPCD_MODE"}) <= 0)
{
  die "TPCD_MODE environment variable not set - uni/smp/mln \n";
}
if (length($ENV{"TPCD_ROOTPRIV"}) <= 0)
{
  die "TPCD_ROOTPRIV environment variable not set - yes/no \n";
}

#set up local variables
$runNum=$ENV{"TPCD_RUNNUMBER"};
$runDir=$ENV{"TPCD_RUN_DIR"};
$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$dbname=$ENV{"TPCD_DBNAME"};
$sf=$ENV{"TPCD_SF"};
$platform=$ENV{"TPCD_PLATFORM"};
$delim=$ENV{"TPCD_PATH_DELIM"};
$gatherstats=$ENV{"TPCD_GATHER_STATS"};
$product=$ENV{"TPCD_PRODUCT"};
$RealAudit=$ENV{"TPCD_AUDIT"};
$inlistmax=$ENV{"TPCD_INLISTMAX"};
$pn=$ENV{"TPCD_PHYS_NODE"};
$logDir=$ENV{"TPCD_LOG_DIR"};
$rootPriv=$ENV{"TPCD_ROOTPRIV"};
$mode=$ENV{"TPCD_MODE"};
$getSnaps=$ENV{"TPCD_GET_SNAPS"};
$getProfiles=$ENV{"TPCD_PROFILES"};

print "Are we here\n";

if (( $mode eq "uni" ) || ( $mode eq "smp" ))
{
  $all_ln="once";
  $all_pn="once";
  $once="once";
}
else
{
  $all_ln="all_ln";
  $all_pn="all_pn";
  $once="once";
}


if ($inlistmax eq "default")
{
  $inlistmax = 400;
}

# the auditruns directory is where we have already generate the sql files for the
# updates and the power tests

# append isolation level information about tpcdbatch to the miso file
# the miso file is created here but appended to for power and throughput
#information

$misofile="$runDir${delim}miso$runNum";
if ( -e $misofile )
{
    &rm("$misofile");
}
# if we are in real audit mode then we must start the db manager now since
# there must be no activity on the database between the time the build script
# has finished and the time the power test is started
if ( $RealAudit eq "yes" )
```

```
{
   system("db2start");
   system("db2 activate database $dbname");
}

# do not activate the database
#if ( $RealAudit ne "yes" )
#{
#   system("db2 activate database $dbname");
#}



#Report current log info to the run# directory in a file called start-
Log.Info
system("perl getLogInfo.pl startLog");
open(MISO, ">$misofile") || die "Can't open $misofile: $!\n";
$curTs = `perl gettimestamp "long"`;
print MISO "Timestamp and isolation level of tpcdbatch before power
run at : $curTs\n";
close(MISO);
if ( $product eq "pe" )
{
   system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_id,isolation from sysibm.sysplan where
name like 'TPCD%'\"; db2 connect reset; db2 terminate  >> $run-
Dir${delim}miso$runNum ");
}
else
{
   &verifyTPCDbatch("$misofile","$dbname");
}

if ($platform eq "aix")
{

  # Create the sysunused file.  This reports what disks are attached,
and which
  # ones are being used.  Its use spans both the runpower and run-
throughtput tests
  system("echo \"The following disks are assigned to the indicated vol-
ume groups\" > $runDir/sysunused$runNum") && die "cannot create
$runDir/sysunused$runNum";

  system("lspv >> $runDir/sysunused$runNum");
  system("echo \"The following volume groups are currently online\" >>
$runDir/sysunused$runNum");
  $curTs = `perl gettimestamp "long"`;
  system("echo \"$curTs\" >> $runDir/sysunused$runNum");
  system("lsvg -o >> $runDir/sysunused$runNum");
  # show the disks that are used/unused
  system("getdisks \"Before the start of the Power Test\"");

}
else
{
  # for all other platforms
  system("echo Assume that all portions of the system are used >>
$runDir${delim}sysunused$runNum");
}

&getConfig("p");
if ( $rootPriv eq "yes" )
{
   # get the o/s tuning parameters...currently AIX only and only if your
   # user has root privileges to run this
   &getOSTune("p");
}
if ($gatherstats eq "on")
{
   # gather vm io and net stats
   if ($platform eq "aix" || $platform eq "sun" || $platform eq "ptx" ||
       $platform eq "hp" || $platform eq "linux")
   {
      # gather vmstats and iostats (and net stats if in mpp mode)
      print "Calling getstats \n";
      system("perl getstats p > getstats.log &");
      if ( $getProfiles eq "y") {
        system("perl getprofiles p &");
      }
      if ( $getSnaps eq "y") {
        system("./getSnaps.sh $runDir &");
      }
   }
   else
   {
       print "Stats gather not set up for current platform $platform\n";
   }
}

# print to screen what type of run is running and set variables to run
# the query and update streams in parallel
if ($runUF ne "UF")
{
   $semcontrol = "off";
   print "Beginning power stream....no update functions\n";

   $streamEx = "";
   $streamExNT = "";
}
else
{
   $semcontrol = "on";
   print "Beginning power stream....with update functions\n";
   if ( $platform eq "nt" )
   {
      $streamExNT = "start /b";
      $streamEx = "";
   }
   else
   {
      $streamExNT = "";
      $streamEx = "&";
   }
}

# bbe This new line (below) runs queries for power test

print "Starting tpcdbatch...\n";
$ret=system("$streamExNT $audit-
Dir${delim}auditruns${delim}tpcdbatch -d $dbname -f $run-
Dir${delim}qtextpow.sql -r on -b on -s $sf -u p1 -m $inlistmax -n 0 -l
$auditDir${delim}auditruns${delim}querytext${delim}streampow.list -p
$semcontrol $streamEx");


if ( $runUF eq "UF" )
{
  $ret2 = system("$auditDir${delim}auditruns${delim}tpcdbatch -d
$dbname -f $runDir${delim}qtextquf.sql -r on -b on -s $sf -u p2 -m
$inlistmax -n 0 -l $audit-
Dir${delim}auditruns${delim}querytext${delim}streampuf.list");
}
else
{
   $ret2 = 0; # If UFs were not running, then the stream cannot fail
}


if (($ret2 == 0) && ($ret == 0))
```

```
{
   print "Power stream completed succesfully.\n";
}
else
{
   print "Power stream failed. ret=$ret\n";
}

system("echo power_stop_token > /usr/tpcd/tools/power_stop");

if ($platform eq "aix")
{
   # show that the same disks are still used or unused
   system("getdisks \"After completion of the Power Test\"");

   #clean up
}
if ($gatherstats eq "on")
{
   # gather vm io and net stats
   if ($platform eq "aix" || $platform eq "sun" || $platform eq "ptx" ||
$platform eq "linux")
   {
      # kill the stats that were being gathered
      if ($platform eq "ptx")
      {
         $rc= `perl5 zap "-f" "sar"`;
         $rc= `perl5 zap "-f" "sadc"`;
      }
      else
      {
         $rc= `perl5 zap "-f" "vmstat"`;
#           $rc= `perl5 zap "-f" "iostat"`;
      }
      if ( $pn > 1 )
      {
         $rc= `perl5 zap "-f" "netstat"`;
      }
      $rc= `perl5 zap "-f" "getstats"`;
   }
}

open(MISO, ">>$misofile") || die "Can't open $misofile: $!\n";
$curTs = `perl gettimestamp "long"`;
print MISO "Timestamp and isolation level of tpcdbatch after power run
at : $curTs\n";
close(MISO);

if ( $product eq "pe" )
{
   system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_id,isolation from sysibm.sysplan where
name like 'TPCD%'\";db2 connect reset;db2 terminate  >> $run-
Dir${delim}miso$runNum");
}
else
{
   &verifyTPCDbatch("$misofile","$dbname");
}
if ( $RealAudit ne "yes" )
{
   $curTs = `perl gettimestamp "short"`;
   # grab the db and dbm snapshot before we deactivate
   system("db2 get snapshot for all on $dbname > $run-
Dir${delim}dbrun$runNum.snap.$curTs");
   system("db2 get snapshot for database manager >> $run-
Dir${delim}dbrun$runNum.snap.$curTs");
}

######################

# now copy the reports from the count of streams files into one final
file
&cat("$runDir${delim}pstrcnt*","$runDir${delim}mpstrcnt$runNum");
#(NOTE: there is a dependancy that this mpstrcnt file exist before the
# calcmetrics.pl script is called, both because it is used as input for
# calcmetrics.pl, and because the output from calcmetrics is used as
# the trigger for watchstreams to complete, and watchstreams cats its
# output at the end of the mstrcnt file.

# generate the mpinter?.metrics file in the run directory
#require 'calcmetricsp.pl';
if ( $runUF eq "UF")
{
  system("perl calcmetricsp.pl UF");
}
else
{
   system("perl calcmetricsp.pl");
}

# concatenate all the throughput inter files that were used to
# generate these results into the calcmetrics output file (mpin-
terX.metrics)
#cd $TPCD_RUN_DIR
&cat("$runDir${delim}mpqinter*","$runDir${delim}mpinter$runNum.met
rics");

if ($runUF eq "UF") {

&cat("$runDir${delim}mpufinter*","$runDir${delim}mpinter$runNum.me
trics");
}

#if ($runUF eq "no") {
#  &rm("$runDir${delim}mpuf*");
#}

######################

# no longer activate/deactivate the database
#if ( $RealAudit ne "yes" )
#{
#    # deactivate the database
#    system("db2 deactivate database $dbname");
#}

# do not stop the database after the power test
#if ( $RealAudit ne "yes" )
#{
#    system("db2stop");
#}

1;

sub getConfig
{
  $testtype=$_[0];
  print "Getting database configuration.\n";
  $dbtunefile="$runDir${delim}m${testtype}dbtune${runNum}";
  open(DBTUNE, ">$dbtunefile") || die "Can't open $dbtunefile: $!\n";
  $timestamp=`perl gettimestamp "long"`;
  print DBTUNE "Database and Database manager configuration taken
at : $timestamp";
  close(DBTUNE);
  system("db2level >> $dbtunefile");
  system("db2 get database configuration for $dbname >> $dbtune-
file");
  system("db2 get database manager configuration >> $dbtunefile");
  system("db2set >> $dbtunefile");
```

```
  if (( $mode eq "mln" ) || ( $mode eq "mpp"))
  {
     $cfgfile="$runDir${delim}dbtune${runNum}.";
     #removed by Alex due to hang
     #system("db2_all '||\" typeset -i ln=##; db2 get db cfg for $dbname
> $cfgfile\${ln} ; db2 get dbm cfg >> $cfgfile\${ln}; db2set >>
$cfgfile\${ln}; db2 terminate '");
  }


}

sub getOSTune
{
  $testtype=$_[0];
  if ( $platform eq "aix" )
  {
      print "Getting OS and VMdatabase configuration.\n";
      $ostunefile="$runDir${delim}m${testtype}ostune${runNum}";
      open(OSTUNE, ">$ostunefile") || die "Can't open $ostunefile: $!\n";
      $timestamp=`perl gettimestamp "long"`;
      print OSTUNE "Operating System and Virtual Memory configura-
tion taken at : $timestamp";
      close(OSTUNE);
      sys-
tem("${delim}usr${delim}samples${delim}kernel${delim}schedtune >>
$ostunefile");
      system("${delim}usr${delim}samples${delim}kernel${delim}vmtune
>> $ostunefile");
  }
  else
  {
      print "OS parameters retrieval not supported for $platform \n";
  }

}

sub verifyTPCDbatch
{
  $logfile=$_[0];
  $dbname=$_[1];
  $file="verifytpcdbatch.clp";
  open(VERTBL, ">$file") || die "Can't open $file: $!\n";
  print VERTBL "connect to $dbname;\n";
  print VERTBL "select name,creator,valid,last_bind_time,isolation
from sysibm.sysplan where name like 'TPCD%';\n";
  print VERTBL "connect reset;\n";
  print VERTBL "terminate;\n";
  close(VERTBL);
  system("db2 -vtf $file >> $logfile");
}
```

D.6 runthroughput

```
:   # -*-Perl-*-

eval 'exec perl5 -S $0 ${1+"$@"}' # Horrible kludge to convert this
    if 0;                         # into a "portable" perl script

# usage  runthroughput [UF]
# where UF is the optional parameter that says to run the throughput
test
# with the update functions.  By default, the update functions are not
# run
# If UF is not supplied and a number is supplied, then that number is
taken
# as the number of concurrent throughput streams to run.  This is also
optional

push(@INC, split(':', $ENV{'PATH'}));

# Get TPC-D specific environment variables
require 'getvars';

# Use the macros in here so that they can handle the platform differ-
ences.
# macro.pl should be sourced from cmvc, other people wrote and
maintain it.
require "macro.pl";
require "tpcdmacro.pl";

$runUF="no";
if (@ARGV > 0)
{
  if ($ARGV[0] eq "UF")
  {
    $runUF=$ARGV[0];
  }
}

if (length($ENV{"TPCD_AUDIT_DIR"}) <= 0)
{
  die "TPCD_AUDIT_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_RUN_DIR"}) <= 0)
{
  die "TPCD_RUN_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_DBNAME"}) <= 0)
{
  die "TPCD_DBNAME environment variable not set\n";
}
if (length($ENV{"TPCD_RUNNUMBER"}) <= 0)
{
  die "TPCD_RUNNUMBER environment variable not set\n";
}
if (length($ENV{"TPCD_NUMSTREAM"}) <= 0)
{
  die "TPCD_NUMSTREAM environment variable not set\n";
}
if (length($ENV{"TPCD_RUN_ON_MULTIPLE_NODES"}) <= 0)
{
  die "TPCD_RUN_ON_MULTIPLE_NODES environment variable not
set\n";

}
if (length($ENV{"TPCD_SF"}) <= 0)
{
  die "TPCD_SF environment variable not set\n";
}
if (length($ENV{"TPCD_PRODUCT"}) <= 0)
{
  die "TPCD_PRODUCT environment variable not set\n";
}
if (length($ENV{"TPCD_PLATFORM"}) <= 0)
{
  die "TPCD_PLATFORM environment variable not set\n";
}
if (length($ENV{"TPCD_AUDIT"}) <= 0)
{
    die "Must set TPCD_AUDIT env't var.  Real audit timing sequence
run if yes\n";
}
if (length($ENV{"TPCD_LOG_DIR"}) <= 0)
{
    $ENV{"TPCD_LOG_DIR"} = "NULL";
}
if (length($ENV{"TPCD_MODE"}) <= 0)
```

```
{
  die "TPCD_MODE environment variable not set - uni/smp/mln \n";
}
if (length($ENV{"TPCD_ROOTPRIV"}) <= 0)
{
  die "TPCD_ROOTPRIV environment variable not set - yes/no \n";
}

#set up local variables
$runNum=$ENV{"TPCD_RUNNUMBER"};
$numStream=$ENV{"TPCD_NUMSTREAM"};
$runDir=$ENV{"TPCD_RUN_DIR"};
$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$dbname=$ENV{"TPCD_DBNAME"};
$sf=$ENV{"TPCD_SF"};
$product=$ENV{"TPCD_PRODUCT"};
$multinode=$ENV{"TPCD_RUN_ON_MULTIPLE_NODES"};
$platform=$ENV{"TPCD_PLATFORM"};
$delim=$ENV{"TPCD_PATH_DELIM"};
$RealAudit=$ENV{"TPCD_AUDIT"};
$inlistmax=$ENV{"TPCD_INLISTMAX"};
$gatherstats=$ENV{"TPCD_GATHER_STATS"};
$logDir=$ENV{"TPCD_LOG_DIR"};
$rootPriv=$ENV{"TPCD_ROOTPRIV"};
$mode=$ENV{"TPCD_MODE"};

$path="$auditDir${delim}auditruns";

if (( $mode eq "uni" ) || ( $mode eq "smp" ))
{
  $all_ln="once";
  $all_pn="once";
  $once="once";
}
else
{
  $all_ln="all_ln";
  $all_pn="all_pn";
  $once="once";
}

# return 1 if the given pattern(parameter $_[0]) matches any file
sub existfile {
  if ($platform eq "aix" || $platform eq "sun" || $platform eq "ptx" || $plat-
form eq "linux")
  {
    `ls $_[0] 2> /dev/null | wc -l` + 0 != 0;
  }
  else
  {
    `dir /b $_[0] 2> NUL | wc -l` + 0 != 0;
  }
}

if ($inlistmax eq "default")
{
   $inlistmax = 400;
}

# no longer stop and start the dbm between runs when not in realaudit
mode
#if ( $RealAudit ne "yes" )
#{
#    # if we are not in real audit mode then we must start the db man-
ager now
#    system("db2start");
#    # activate the database
#    system("db2 activate database $dbname");
#}

$misofile="$runDir${delim}miso$runNum";
# append isolation level information about tpcdbatch to the miso file
open(MISO, ">>$misofile") || die "Can't open $misofile: $!\n";
$curTs = `perl gettimestamp "long"`;
print MISO "Timestamp and isolation level of tpcdbatch before
throughput run at : $curTs\n";
close(MISO);

if ( $product eq "pe" )
{
   system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_id,isolation from sysibm.sysplan where
name like 'TPCD%'\" >> $runDir${delim}miso$runNum ");
}
else
{
   &verifyTPCDbatch("$misofile","$dbname");
}

# kick off the script that will monitor for the database applications dur-
ing
# the running of the throughput tests.  This will quit when the
mtinterX.metrics
# (where X=runnumber) file has been created.

# set variables to run streams in parallel
if ( $platform eq "nt" )
{
  $streamExNT = "start /b";
  $streamEx = "";
}
else
{
  $streamExNT = "";
  $streamEx = "&";
}
if ( $platform eq "aix"  || $platform eq "sun" || $platform eq "nt" || $plat-
form eq "hp" || $platform eq "linux")
{
   system("$streamExNT perl watchstreams $streamEx");
}
else
{
   die "platform not supported, can't start watchstreams in back-
ground";
}

# show the disks that are used/unused
if ($platform eq "aix")
{
   system("getdisks \"Before the start of the Throughput Test\"");
}
if ($gatherstats eq "on")
{
   # gather vm io and net stats
   if ($platform eq "aix" || $platform eq "sun" || $platform eq "ptx" ||
$platform eq "hp" || $platform eq "linux")
   {
      # gather vmstats and iostats (and net stats if in mpp mode)
      system("perl getstats t &");
   }
   else
   {
      print "Stats gather not set up for current platform $platform\n";
   }
}

if ( $multinode ne "yes" )
{
```

```
  # we are running the query streams and update stream from the
same node or
  # from a serial db....use semaphores for control of the update stream

  # the auditruns directory is where we have already generated the sql
files
  # for the updates and the power tests

  $loopStream=1;

  for ( $loopStream = 1; $loopStream <= $numStream; $loop-
Stream++)
  {
    print "starting stream $loopStream\n";
    system("echo Executing stream $loopStream out of $numStream.");
    # run the queries
    if ( $platform eq "aix" || $platform eq "sun" || $platform eq "nt" ||
$platform eq "ptx" ||
         $platform eq "hp" || $platform eq "linux")
    {
       system("$streamExNT $path${delim}tpcdbatch -d $dbname -f
$runDir${delim}qtextt$loopStream.sql -r on -b on -s $sf -u t1 -m
$inlistmax -n $loopStream -l
$path${delim}querytext${delim}stream$loopStream.list $streamEx");
    }
    else
    {
       die "platform $platform not supported yet";
    }
  }

  # run the update function stream....this will wait until the queries have
  # completed to kick off the updates
    print "starting update stream\n";

  if ($runUF eq "no") {
    $ret=system("$auditDir${delim}auditruns${delim}tpcdbatch -d
$dbname -f $runDir${delim}quft.sql -r on -b on -s $sf -u t -m $inlistmax
-n $numStream -l $runDir${delim}streamuf.list");
  }
  else {
    $ret=system("$auditDir${delim}auditruns${delim}tpcdbatch -d
$dbname -f $runDir${delim}quft.sql -r on -b on -s $sf -u t2 -m $inlist-
max -n $numStream -l $runDir${delim}streamuf.list");
  }
  print "update stream done\n";


  &getConfig("t");
  if ( $rootPriv eq "yes" )
  {
     # get the o/s tuning parameters...currently AIX only and only if your
     # user has root privileges to run this
     &getOSTune("t");
  }
}
else
{
  # we are running the query streams and update stream from different
nodes, use
  # files and rksh to control the update stream
  system("runthru.pe");
}

if ($platform eq "aix")
{
    # show the disks that are used/unused
    system("getdisks \"After the completion of the Throughput Test\"");
}
if ($gatherstats eq "on")
{
    # gather vm io and net stats
    if ($platform eq "aix" || $platform eq "sun" || $platform eq "ptx" ||
$platform eq "linux")
    {
      # kill the stats that were being gathered
       if ($platform eq "ptx")
       {
          $rc= `perl5 zap "-f" "sar"`;
          $rc= `perl5 zap "-f" "sadc"`;
       }
       else
       {
          $rc= `perl5 zap "-f" "vmstat"`;
          $rc= `perl5 zap "-f" "iostat"`;
       }
       if ( $pn > 1 )
       {
          $rc= `perl5 zap "-f" "netstat"`;
       }
       $rc= `perl5 zap "-f" "getstats"`;
    }
}

open(MISO, ">>$misofile") || die "Can't open $misofile: $!\n";
$curTs = `perl gettimestamp "long"`;
print MISO "Timestamp and isolation level of tpcdbatch after through-
put run at : $curTs\n";
close(MISO);

if ( $product eq "pe" )
{
    system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_id,isolation from sysibm.sysplan where
name like 'TPCD%'\" >> $runDir${delim}miso$runNum");
}
else
{
   &verifyTPCDbatch("$misofile","$dbname");
}

if ( $RealAudit ne "yes" )
{
    $curTs = `perl gettimestamp "short"`;
    # grab the db and dbm snapshot before we deactivate
    system("db2 get snapshot for all on $dbname > $run-
Dir${delim}dbTrun$runNum.snap.$curTs");
    system("db2 get snapshot for database manager >> $run-
Dir${delim}dbTrun$runNum.snap.$curTs");
}

# now copy the reports from the count of streams files into one final
file
&cat("$runDir${delim}strcnt*","$runDir${delim}mstrcnt$runNum");
#(NOTE: there is a dependancy that this mstrcnt file exist before the
# calcmetrics.pl script is called, both because it is used as input for
# calcmetrics.pl, and because the output from calcmetrics is used as
# the trigger for watchstreams to complete, and watchstreams cats its
# output at the end of the mstrcnt file.

# generate the mtinter?.metrics file in the run directory
#require 'calcmetrics.pl';

if ( $runUF ne "no")
{
  system("perl calcmetrics.pl $numStream UF");
}
else
{
  system("perl calcmetrics.pl $numStream");
```

```
}

# concatenate all the throughput inter files that were used to
# generate these results into the calcmetrics output file
(mtinterX.metrics)
#cd $TPCD_RUN_DIR
&cat("$runDir${delim}mts*inter*","$runDir${delim}mtinter$runNum.metr
ics");

if ($runUF ne "no") {

&cat("$runDir${delim}mtufinter*","$runDir${delim}mtinter$runNum.metr
ics");
}

if (&existfile("$runDir${delim}mp*")) {
  # generate the mplot stuff
  system("perl gen_mplot");

  # generate the mlog information file
  require 'buildmlog';
}

#if ($runUF eq "no") {
#  &rm("$runDir${delim}mtuf*");
#}

# deactivate the database  this needs to remain at the end of run
throughput so
# asynchronous writing of the log files completes.
system("db2 deactivate database $dbname");
$rc=&dodb_noconn("db2 get db cfg for $dbname | grep -i log >> $run-
Dir${delim}endLog.Info",$all_ln);
if ( $logDir ne "NULL" )
{
    $rc=&dodb_noconn("$dircmd $logDir >> $run-
Dir${delim}endLog.Info",$all_ln);
}

#system("db2_all \"]}db2 get db cfg for tpcd | grep -i log >> $run-
Dir${delim}endLog.Info ; db2 terminate\' ");
#system("ls -ltra /node??vg.log/NODE00* >>
$runDir${delim}endLog.Info");

#Create Catalog info
$rc = system("perl catinfo.pl p");


if ( $rc != 0 )
{
  warn "catinfo failed!!!\n";
}

#Report current log info to the run# directory in a file called end-
Log.Info
system("perl getLogInfo.pl endLog");

# if we are in audit mode we must do a db2stop at the end of the
power/throughput run
if ( $RealAudit eq "yes" )
{
    system("db2stop");
}

1;

sub getConfig
{
  $testtype=$_[0];
  print "Getting database configuration.\n";
  $dbtunefile="$runDir${delim}m${testtype}dbtune${runNum}";
  open(DBTUNE, ">$dbtunefile") || die "Can't open $dbtunefile: $!\n";
  $timestamp=`perl gettimestamp "long"`;
  print DBTUNE "Database and Database manager configuration taken
at : $timestamp";
  close(DBTUNE);
  system("db2level >> $dbtunefile");
  system("db2 get database configuration for $dbname >> $dbtune-
file");
  system("db2 get database manager configuration >> $dbtunefile");
  system("db2set >> $dbtunefile");

}

sub getOSTune
{
  $testtype=$_[0];
  if ( $platform eq "aix" || $platform eq "linux")
  {
      print "Getting OS and VMdatabase configuration.\n";
      $ostunefile="$runDir${delim}m${testtype}ostune${runNum}";
      open(OSTUNE, ">$ostunefile") || die "Can't open $ostunefile: $!\n";
      $timestamp=`perl gettimestamp "long"`;
      print OSTUNE "Operating System and Virtual Memory configura-
tion taken at : $timestamp";
      close(OSTUNE);
      sys-
tem("${delim}usr${delim}samples${delim}kernel${delim}schedtune >>
$ostunefile");
      system("${delim}usr${delim}samples${delim}kernel${delim}vmtune
>> $ostunefile");
  }
  else
  {
      print "OS parameters retrieval not supported for $platform \n";
  }

}

sub verifyTPCDbatch
{
  $logfile=$_[0];
  $dbname=$_[1];
  $file="verifytpcdbatch.clp";
  open(VERTBL, ">$file") || die "Can't open $file: $!\n";
  print VERTBL "connect to $dbname;\n";
  print VERTBL "select name,creator,valid,last_bind_time,isolation
from sysibm.sysplan where name like 'TPCD%';\n";
  print VERTBL "connect reset;\n";
  print VERTBL "terminate;\n";
  close(VERTBL);
  system("db2 -vtf $file >> $logfile");

}
```

D.7 ploaduf1

```
#!/bin/ksh
RFpair=$1
#~/tpcd/tools/load_line_uf $RFpair &
/usr/tpcd/tools/load_line_uf $RFpair &

#~/tpcd/tools/load_orders_uf $RFpair
/usr/tpcd/tools/load_orders_uf $RFpair &

wait
db2 terminate
```

D.8 ploaduf2

```
#!/bin/ksh
RFpair=$1;
db2 connect to tpcd
db2 "load from delete.$RFpair of del modified by coldel| fastparse
messages /dev/null replace into TPCDTEMP.ORDERS_DEL nonre-
coverable partitioned db config mode load_only part_file_location
/vol1/ufdata;"
db2 commit;
db2 connect reset
db2 terminate
```

Appendix - E ACID Transaction Source Code

E.1 acid.sqc

```
#include "acid.h"

#if (defined(SQLPTX) || defined(SQLWINT) || defined(SQLSUN) ||
defined(Linux))
double nearest(double);
#endif /* SQLPTX */


#define DEADLOCK -911

/*
#define TRUNC2(d) ((floor((d)*100.0))/100.0)
*/
/*
#define TRUNC2(d) ((floor(nearest((d)*100.0)))*0.01)
*/
/*
#define TRUNC2(d)  ((floor(nearest((d)*1000.0)/10.0)/100.0))
*/
#define TRUNC2(d)  ((floor(nearest((d)*100000.0)/1000.0)/100.0))

void sqlerror(char * , struct sqlca *);

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
char dbname[8];   /* = "tpcd"; */
EXEC SQL END DECLARE SECTION;

#ifdef SQLWINT

 /*
 ** redefine gettimeofday so I don't have to
 ** change too much aix-specific code
 */
/*#typedef struct timeval { unsigned tv_sec; unsigned tv_usec; }; */
typedef struct timezone { int dummy; };
struct timeb timer;

 void gettimeofday( struct timeval *tv, struct timezone *tz)
 {
   ftime(&timer);
   tv->tv_sec = timer.time;
   tv->tv_usec = timer.millitm  * 1000;
   tz->dummy = 0;
 }
#endif

/*-------------------------------------------------------------------*/
/*        acidQ                                                      */
/*-------------------------------------------------------------------*/
int acidQ (struct acidQ_struct *acid)
{
   time_t timeT;
   FILE *out;
   char out_fn[50];
   struct timeval tv;
   struct timezone tz;
   int mypid;
   int rc = 0;

   EXEC SQL BEGIN DECLARE SECTION;
   sqlint32    okey;
   sqlint32    lEprice;
   double  eprice;
   EXEC SQL END DECLARE SECTION;

   okey = acid->o_key;

   /* mypid = getpid(); */
   mypid = acid->tag;

   sprintf(out_fn,
"%s%cacidQ.out.%d",getenv("TPCD_TMP_DIR"),del(),mypid);
   out=fopen(out_fn,"a");
   if (out == NULL)
   {
     fprintf(stderr, "ERROR input file %s could not be appended
to!!\n",out_fn);
    }

   gettimeofday(&tv, &tz);
   time(&timeT);
   fprintf(out,"\n---------- START of acidQ tag: %d ----------\n\n",mypid);
   fprintf(out, "acidQ tag: %d, begin transaction time: (%us %06uu)
%s",
          mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
   fprintf(out, "okey: %d\n", okey);

   gettimeofday(&tv, &tz);
   time(&timeT);
   fprintf(out,"acidQ tag: %d, before read of LINEITEM: (%us %06uu)
%s",
          mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));

   /*
   ** use the same sql code as used in the consistsql.pl to
   ** run the consistency acid queries.  Note we assign an long int
   ** to lEprice (we make it 10s of pennies by * 1000).  Then divide
   ** by 1000.0 and cast it to a double (eprice) for printing
   */

   EXEC SQL
     SELECT

INTEGER(DECIMAL(SUM(DECIMAL(INTEGER(INTEGER(DECIMAL
        (INTEGER(100*DECIMAL(L_EXTENDEDPRICE,20,3)), 20,3) *
        (1-L_DISCOUNT)) * (1+L_TAX)),20,3)/100.0),20,3) * 1000)
      into :lEprice
     FROM
       TPCD.LINEITEM
     WHERE
       L_ORDERKEY = :okey;

   if (sqlca.sqlcode != 0) {
     rc = sqlca.sqlcode;
     fprintf(out,"acidQ **ERROR** sqlcode = %d\n",sqlca.sqlcode);
     sqlerror("acidQ: select sum(l_extendedprice)", &sqlca);
     goto Qerror;
   }
   eprice = (double)lEprice / 1000.0;  /* translate to double for printout*/

   gettimeofday(&tv, &tz);
   time(&timeT);
   fprintf(out,"ACID tag: %d, after read of LINEITEM: (%us %06uu)
%s",
          mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
   fprintf(out, "okey: %d \t sum(l_extendedprice): %0.3f\n",
          okey, eprice);

   EXEC SQL COMMIT;
```

```
   if (sqlca.sqlcode != 0) {
      rc = sqlca.sqlcode;
      fprintf(out,"acidQ **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      sqlerror("acidQ: COMMIT", &sqlca);
      goto Qerror;
   }
   acid->l_extendedprice = eprice;

   rc = 0;
   goto Qexit;

 Qerror:
   EXEC SQL rollback work;
   if (sqlca.sqlcode != 0) sqlerror("acidQ: ROLLBACK FAILED",
&sqlca);

 Qexit:
   fprintf(out,"\n---------- END of acidQ tag: %d ----------\n\n",mypid);
   fflush(out);fclose(out);
   return(rc);
}

/*-------------------------------------------------------------*/
/*        ast_acidQ                                            */
/*-------------------------------------------------------------*/
int ast_acidQ (struct acidQ_struct *acid)
{
   time_t timeT;
   FILE *out;
   char out_fn[50];
   struct timeval tv;
   struct timezone tz;
   int mypid;
   int rc = 0;


   EXEC SQL BEGIN DECLARE SECTION;
   double    ast_lEprice;
   double    ast_eprice;
   EXEC SQL END DECLARE SECTION;


   /* mypid = getpid(); */
   mypid = acid->tag;

   sprintf(out_fn,
"%s%cast_acidQ.out.%d",getenv("TPCD_TMP_DIR"),del(),mypid);
   out=fopen(out_fn,"a");
   gettimeofday(&tv, &tz);
   time(&timeT);
   fprintf(out,"\n---------- START of ast_acidQ tag: %d ----------
\n\n",mypid);
   fprintf(out, "ast_acidQ tag: %d, begin transaction time: (%us %06uu)
%s",
         mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));

   gettimeofday(&tv, &tz);
   time(&timeT);
   fprintf(out,"ast_acidQ tag: %d, before read of LINEITEM: (%us
%06uu) %s",
         mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));

   /*
   ** use the same query acidQ except don't select for specfic okey.
   ** this ensures that the ast will be used instead of the base table
   ** Have to use ast_lEprice as double since this sum is so big
   */
   EXEC SQL
     SELECT
       SUM ( L_EXTENDEDPRICE*(1-L_DISCOUNT)*(1 + L_TAX))
         into :ast_lEprice
       FROM
         TPCD.LINEITEM;

   if (sqlca.sqlcode != 0) {
      rc = sqlca.sqlcode;
      fprintf(out,"ast_acidQ **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      sqlerror("ast_acidQ: select sum(l_extendedprice)", &sqlca);
      goto Qerror;
   }
   ast_eprice = ast_lEprice;  /* use ast_eprice  for printout to be
consistent*/

   gettimeofday(&tv, &tz);
   time(&timeT);
   fprintf(out,"AST_ACID tag: %d, after read of LINEITEM: (%us
%06uu) %s",
         mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
   fprintf(out, "sum(l_extendedprice): %0.3f\n",
         ast_eprice);

   EXEC SQL COMMIT;
   if (sqlca.sqlcode != 0) {
      rc = sqlca.sqlcode;
      fprintf(out,"ast_acidQ **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      sqlerror("ast_acidQ: COMMIT", &sqlca);
      goto Qerror;
   }
   acid->l_extendedprice = ast_eprice;

   rc = 0;
   goto Qexit;

 Qerror:
   EXEC SQL rollback work;
   if (sqlca.sqlcode != 0) sqlerror("ast_acidQ: ROLLBACK FAILED",
&sqlca);

 Qexit:
   fprintf(out,"\n---------- END of ast_acidQ tag: %d ----------\n\n",mypid);
   fflush(out);fclose(out);
   return(rc);
}
/*-------------------------------------------------------------*/
/*        acidT                                                */
/*-------------------------------------------------------------*/
int acidT (struct acidT_struct *acid)
{

   time_t timeT;
   FILE *out;
   char out_fn[50];
   struct timeval tv;
   struct timezone tz;
   int mypid;
   int rc = 0;

   EXEC SQL BEGIN DECLARE SECTION;
   sqlint32      o_key, l_key, delta;
   sqlint32      l_partkey, l_suppkey;
   double    l_quantity, l_tax, l_discount, l_extendedprice;
   double    o_totalprice;
   double    new_quantity, rprice, cost, new_extprice, new_ototal,
ototal;
   EXEC SQL END DECLARE SECTION;

   EXEC SQL DECLARE l_cursor CURSOR FOR
     SELECT l_partkey, l_suppkey, l_quantity,
     l_tax, l_discount,
     l_extendedprice
```

```
        FROM tpcd.lineitem
        WHERE l_orderkey = :o_key
        AND l_linenumber = :l_key
        FOR UPDATE OF l_extendedprice, l_quantity;

    EXEC SQL DECLARE o_cursor CURSOR FOR
      SELECT o_totalprice
        FROM tpcd.orders
        WHERE o_orderkey = :o_key
        FOR UPDATE OF o_totalprice;

    if (acid->termination < 0 || acid->termination > 3) acid->termination =
0;
    o_key = acid->o_key;
    l_key = acid->l_key;
    delta = acid->delta;

    if (acid->logging) {
      /* mypid = getpid(); */
      mypid = acid->tag;
      sprintf(out_fn,
"%s%cacidT.out.%d",getenv("TPCD_TMP_DIR"),del(),mypid);
      out=fopen(out_fn,"a");
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"\n---------- START of acidT tag: %d ----------\n\n",mypid);
      fprintf(out, "acidT tag: %d, begin transaction time: (%us %06uu)
%s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
      fprintf(out, "o_key: %d\tl_key: %d\tdelta: %d\n", o_key, l_key,
delta);
    }
#ifdef DEBUG
    printf("o_key: %d\tl_key: %d\tdelta: %d\n", o_key, l_key, delta);
#endif

 retry_tran:

    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, before read of LINEITEM: (%us %06uu)
%s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }

    EXEC SQL OPEN l_cursor;
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: OPEN l_cursor", &sqlca);
      goto Terror;
    }

    EXEC SQL FETCH l_cursor INTO
      :l_partkey, :l_suppkey, :l_quantity, :l_tax,
      :l_discount, :l_extendedprice;
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: FETCH l_cursor", &sqlca);
      goto Terror;
    }

#ifdef DEBUG
    printf("l_quantity = %0.3f\n",l_quantity);
    printf("l_tax = %0.3f  \n",l_tax);
    printf("l_discount = %0.3f \n",l_discount);
    printf("l_extendedprice = %0.3f \n", l_extendedprice);
#endif

    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, after read of LINEITEM: (%us %06uu)
%s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
      fprintf(out, "l_partkey: %d  l_suppkey: %d  l_quantity: %0.3f\nl_tax:
%0.3f  l_discount: %0.3f  l_extendedprice: %0.3f\n",
              l_partkey, l_suppkey, l_quantity, l_tax, l_discount,
l_extendedprice);
    }

    rprice = TRUNC2( l_extendedprice/l_quantity );
    cost = TRUNC2( rprice * delta );
    new_extprice = l_extendedprice + cost;
    new_quantity = l_quantity + delta;

#ifdef DEBUG
    printf("rprice = %0.3f\n", rprice );
    printf("cost = %0.3f\n", cost );
    printf("new_extprice = %0.3f\n", new_extprice );
    printf("new_quantity = %0.3f\n", new_quantity );
#endif

    EXEC SQL UPDATE tpcd.lineitem
      SET l_extendedprice = :new_extprice,
          l_quantity = :new_quantity
      WHERE CURRENT OF l_cursor;

    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: UPDATE l_cursor", &sqlca);
      goto Terror;

    }

    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, after update of LINEITEM: (%us %06uu)
%s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
      fprintf(out, "updated l_extendedprice: %0.3f\n", new_extprice );
      fprintf(out, "updated l_quantity: %0.3f\n", new_quantity );
    }

    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, before read of ORDER: (%us %06uu)
%s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
```

```
  EXEC SQL OPEN o_cursor;
  if (sqlca.sqlcode != 0) {
    if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
      fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
    } else {
      fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: OPEN o_cursor", &sqlca);
    goto Terror;
  }

  EXEC SQL FETCH o_cursor INTO :o_totalprice;
  if (sqlca.sqlcode != 0) {
    if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging)
    {
      fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
    }
    else
    {
      fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
    }
    sqlerror("acidT: FETCH o_cursor", &sqlca);
    goto Terror;
  }

#ifdef DEBUG
  printf("o_totalprice = %0.3f\n",o_totalprice);
#endif

  if (acid->logging) {
    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"acidT tag: %d, after read of ORDER: (%us %06uu)
%s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    fprintf(out, "o_totalprice: %0.3f\n", o_totalprice);
  }

#ifdef DEBUG
  {
   double zeroone= l_extendedprice * (1.0- l_discount);
   double zeroonetimes= (l_extendedprice * (1.0- l_discount))*100.0;
   double firstone = TRUNC2(l_extendedprice * (1.0-l_discount));
   double notone= TRUNC2 ( l_extendedprice * (1.0-l_discount)) *
(1.0+l_tax);
   double secondone= TRUNC2( TRUNC2( l_extendedprice * (1.0-
l_discount) ) * (1.0+l_tax) );
   printf("firstone=   %f\n", firstone);
   printf("zeroone=   %f\n", zeroone);
   printf("zeroonetimes=   %f\n", zeroonetimes);
   printf("notone=   %f\n", notone);
   printf("secondone=   %f\n", secondone);
  }
#endif
  ototal = o_totalprice -
           TRUNC2( TRUNC2( l_extendedprice * (1-l_discount) ) *
(1+l_tax) );
  new_ototal = TRUNC2( new_extprice * (1.0-l_discount) );
  new_ototal = TRUNC2( new_ototal * (1.0+l_tax) );
  new_ototal = ototal + new_ototal;

#ifdef DEBUG
  printf("o_totalprince= %f\n",o_totalprice);
  printf("ototal= %0.3f\n",ototal);
  printf("ototal= %f\n",ototal);
  printf("new_ototal= %0.3f\n",new_ototal);
#endif

  EXEC SQL UPDATE tpcd.orders
    SET o_totalprice = :new_ototal
    WHERE CURRENT OF o_cursor;
  if (sqlca.sqlcode != 0) {
    if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
      fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
    } else {
      fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: UPDATE o_cursor", &sqlca);
    goto Terror;
  }

  if (acid->logging) {
    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"acidT tag: %d, after update of ORDER: (%us %06uu)
%s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    fprintf(out, "updated o_totalprice: %0.3f\n", new_ototal) ;
  }

  if (acid->logging) {
    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"acidT tag: %d, before insert into HISTORY: (%us
%06uu) %s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
  }

  EXEC SQL INSERT INTO tpcd.history values
    (:l_partkey, :l_suppkey, :o_key, :l_key, :delta, CURRENT
TIMESTAMP);
  if (sqlca.sqlcode != 0) {
    if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
      fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
    } else {
      fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: INSERT INTO history", &sqlca);
    goto Terror;
  }

  if (acid->logging) {
    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"acidT tag: %d, after insert into HISTORY: (%us %06uu)
%s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
  }

  /* sleep for 1 second for 80% of the transactions */
#ifdef SQLWINT
  if ( ((rand() % (100)) + 1) < 80 ) sleep(1);
#else
  if ( ((random() % (100)) + 1) < 80 ) sleep(1);
#endif

  switch (acid->termination) {
   case 1:
    {
      if (acid->logging)
      {
```

```
        gettimeofday(&tv, &tz);
        time(&timeT);
        fprintf(out,"acidT tag: %d, wait before COMMIT: (%us %06uu)
%s",
                mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
      }
    }
    sleep(60);
  case 0:
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, immediately before COMMIT: (%us
%06uu) %s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    EXEC SQL CLOSE L_CURSOR;
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: CLOSE L_CURSOR", &sqlca);
      goto Terror;
    }
    EXEC SQL CLOSE O_CURSOR;
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: CLOSE O_CURSOR", &sqlca);
      goto Terror;
    }
    EXEC SQL COMMIT;
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: COMMIT", &sqlca);
      goto Terror;
    }
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, after COMMIT: (%us %06uu) %s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    break;
  case 3:
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, wait before ROLLBACK: (%us %06uu)
%s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    sleep(60);
  case 2:
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, immediately before ROLLBACK: (%us
%06uu) %s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    EXEC SQL CLOSE L_CURSOR;
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: CLOSE L_CURSOR", &sqlca);
      goto Terror;
    }
    EXEC SQL CLOSE O_CURSOR;
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {

        fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: CLOSE O_CURSOR", &sqlca);
      goto Terror;
    }
    EXEC SQL rollback work;
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: ROLLBACK", &sqlca);
      goto Terror;
    }
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, after ROLLBACK: (%us %06uu) %s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
      break;
  }

  acid->l_partkey = l_partkey;
  acid->l_suppkey = l_suppkey;
  acid->l_quantity = l_quantity;
  acid->l_tax = l_tax;
  acid->l_discount = l_discount;
  acid->l_extendedprice = l_extendedprice;
  acid->o_totalprice = o_totalprice;

  rc = 0;
  goto Texit;

 Terror:
```

```
    EXEC SQL CLOSE L_CURSOR;
    EXEC SQL CLOSE O_CURSOR;
  EXEC SQL rollback work;
  if (sqlca.sqlcode != 0) sqlerror("acidT: ROLLBACK FAILED",
&sqlca);

 Texit:
  if (acid->logging) {
    fprintf(out,"\n---------- END of acidT tag: %d ----------\n\n",mypid);
    fflush(out);fclose(out);
  }
  return(rc);
}

/*-------------------------------------------------------------------*/
/*       updateQ                                                     */
/*-------------------------------------------------------------------*/
int updateQ (struct update_struct *us)
{
  FILE *out;
  time_t timeT;
  struct timeval tv;
  struct timezone tz;
  int qnum;
  int rc = 0;
  int i;
  int secs2sleep;
  char buff[256];
  struct acidtype {int logging;} a, *acid;

  EXEC SQL BEGIN DECLARE SECTION;
  double    acctbal;
  double    discount;
  double    price;
  sqlint32      availqty;
  sqlint32      size;
  EXEC SQL END DECLARE SECTION;

  qnum = us->qnum;

  acid = &a;
  acid->logging= 1;

  sprintf(buff, "%s%cupdate.out",getenv("TPCD_TMP_DIR"),del());
  out=fopen(buff,"a");

  gettimeofday(&tv, &tz);
  time(&timeT);
  fprintf(out,"\n---------- START of update ----------\n\n");
  fprintf(out, "update query number: %d, begin transaction time: (%us
%06uu) %s",
          qnum, tv.tv_sec, tv.tv_usec, ctime(&timeT));

  sqlca.sqlcode = 0;
  discount = 0.25;
  price = 5000.50;
  acctbal = 1000.00;
  availqty = 10;
  size = 5;

  for (i=1; i <= 2; i++) {
    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"update query number: %d, pass %d, immediately
before UPDATE: (%us %06uu) %s",
            qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));

    switch (qnum)
    {
     case 1:
     {
       EXEC SQL
         UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
         WHERE L_ORDERKEY IN (326,512,928,995);
       if (sqlca.sqlcode != 0) {
         rc = sqlca.sqlcode;
         if (acid->logging)
         {
           fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                   qnum, i, sqlca.sqlcode);
         }
         else
         {
           fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                   qnum, i, sqlca.sqlcode);
         }
         sqlerror("update query number 1", &sqlca);
         goto Uerror;
       }
       discount = discount * (-1);
       secs2sleep = 300;
       break;
     }
     case 2:
     {
       EXEC SQL
         UPDATE TPCD.SUPPLIER set S_ACCTBAL = S_ACCTBAL
+ :acctbal
         WHERE S_NAME in
('Supplier#000000647','Supplier#000000070','Supplier#000000802');
       if (sqlca.sqlcode != 0) {
         rc = sqlca.sqlcode;
         if (acid->logging)
         {
           fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                   qnum, i, sqlca.sqlcode);
         }
         else
         {
           fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                   qnum, i, sqlca.sqlcode);
         }
         sqlerror("update query number 2", &sqlca);
         goto Uerror;
       }
       acctbal = acctbal * (-1);
       secs2sleep = 90;
       break;
     }
     case 3:
     {
       EXEC SQL
         UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
         WHERE L_ORDERKEY IN (260930, 402497, 457859,
509889, 58117,
                              538311, 588421, 416167, 97830, 90276);
       if (sqlca.sqlcode != 0) {
         rc = sqlca.sqlcode;
         if (acid->logging)
         {
           fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                   qnum, i, sqlca.sqlcode);
         }
```

```
          else
          {
            fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 3", &sqlca);
          goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 300;
        break;
      }
      case 4:
      {
        if ( i ==1 ) {
          EXEC SQL
            UPDATE TPCD.ORDERS set O_ORDERDATE =
O_ORDERDATE - 6 MONTHS
            WHERE O_ORDERKEY =  67461;
          /* WHERE O_ORDERKEY IN
(22400,28515,34338,46596,67461,92644,98307);*/
        } else {
          EXEC SQL
            UPDATE TPCD.ORDERS set O_ORDERDATE =
O_ORDERDATE + 6 MONTHS
            WHERE O_ORDERKEY =  67461;
        }
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          else
          {
            fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 4", &sqlca);
          goto Uerror;
        }
        secs2sleep = 300;
        break;
      }
      case 5:
      {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
          WHERE L_ORDERKEY IN
(70976,566279,152897,84226,232483);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          else
          {
            fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 5", &sqlca);
          goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 300;
        break;
      }
      case 6:
      {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
          WHERE L_ORDERKEY in
(33,131,161,195,229,230,231,323,353,356);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          else
          {
            fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 6", &sqlca);
          goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 300;
        break;
      }
      case 7:
      {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
          WHERE L_ORDERKEY IN
(562917,410659,16550,398401,157634,429920,45411);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          else
          {
            fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 7", &sqlca);
          goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 300;
        break;
      }
      case 8:
      {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
          WHERE L_ORDERKEY IN
(129569,343591,270242,254983,98500,28963);
        if (sqlca.sqlcode != 0) {
```

```
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          else
          {
            fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 8", &sqlca);
          goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 300;
        break;
      }
      case 9:
      {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
          WHERE L_ORDERKEY IN
(113509,232997,246691,379233,448162,32134);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          else
          {
            fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 9", &sqlca);
          goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 300;
        break;
      }
      case 10:
      {

        EXEC SQL
          UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
          WHERE L_ORDERKEY IN
(516487,245411,265799,253025,6914,562020);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          else
          {
            fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 10", &sqlca);
          goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 300;
        break;
      }
      case 11:
      {
        EXEC SQL
          UPDATE TPCD.PARTSUPP set PS_AVAILQTY =
PS_AVAILQTY + :availqty
          WHERE PS_PARTKEY IN
(12098,5134,13334,17052,3452,12552,1084,5797);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          else
          {
            fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 11", &sqlca);
          goto Uerror;
        }
        availqty = availqty * (-1);
        secs2sleep = 180;
        break;
      }
      case 12:
      {
        if ( i ==1 ) {
          EXEC SQL
            UPDATE TPCD.LINEITEM set L_RECEIPTDATE =
L_RECEIPTDATE - 3 YEARS
            WHERE L_ORDERKEY IN
(33,70,195,355,677,837,960,962,1028);
        } else {
          EXEC SQL
            UPDATE TPCD.LINEITEM set L_RECEIPTDATE =
L_RECEIPTDATE + 3 YEARS
            WHERE L_ORDERKEY IN
(33,70,195,355,677,837,960,962,1028);
        }
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          else
          {
            fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 12", &sqlca);
          goto Uerror;
        }
        secs2sleep = 300;
        break;
      }
```

```
    case 13:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
        WHERE L_ORDERKEY IN
(263,9476,32355,34854,53445,56901);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 13", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 90;
      break;
    }
    case 14:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
        WHERE L_ORDERKEY IN (32,225,326,448,449,483,512);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 14", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 180;
      break;
    }
    case 15:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
        WHERE L_ORDERKEY IN (1,4,7,35,135,131300);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 15", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 180;
      break;
    }
    case 16:
    {
      EXEC SQL
        UPDATE TPCD.PART set P_SIZE = P_SIZE + :size
        WHERE P_PARTKEY IN (4,7,15,1313);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 16", &sqlca);
        goto Uerror;
      }
      size = size * (-1);
      secs2sleep = 180;
      break;
    }
    case 17:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_EXTENDEDPRICE =
L_EXTENDEDPRICE + :price
        WHERE L_ORDERKEY IN
(4065,110372,165061,265702,87138);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 17", &sqlca);
        goto Uerror;
      }
      price = price * (-1);
      secs2sleep = 90;
      break;
    }
    default:
    {
      fprintf(out,"ERROR: Invalid query number specified %d\n",
qnum);
```

```
        rc = 1;
        goto Uexit;
      }
    }

    gettimeofday(&tv, &tz);
    time(&timeT);

    if (acid->logging)
      fprintf(out,"update query number: %d, pass %d, after UPDATE:
(%us %06uu) %s",
              qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    else
      fprintf(stderr,"update query number: %d, pass %d, after
UPDATE: (%us %06uu) %s",
              qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));

    if ( i == 2 ) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"update query number: %d, pass %d, sleeping for %d
seconds: (%us %06uu) %s",
              qnum, i, secs2sleep, tv.tv_sec, tv.tv_usec, ctime(&timeT));
      fflush(out);
      system("touch /tmp/tpcd/update.sync.sleep");
      sleep(secs2sleep);
    }

    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"update query number: %d, pass %d, immediately
before COMMIT: (%us %06uu) %s",
            qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));

    EXEC SQL COMMIT;
    if (sqlca.sqlcode != 0) {

      rc = sqlca.sqlcode;
      fprintf(out,"update pass %d, **ERROR** sqlcode = %d\n", i,
sqlca.sqlcode);
      sqlerror("update: COMMIT", &sqlca);
      goto Uerror;
    }
    gettimeofday(&tv, &tz);
    time(&timeT);
    if (acid->logging)
      fprintf(out,"update query number: %d, pass %d, after COMMIT:
(%us %06uu) %s",
              qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    else
      fprintf(stderr,"update query number: %d, pass %d, after
COMMIT: (%us %06uu) %s",
              qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));
  }

  rc = 0;
  goto Uexit;

Uerror:
  EXEC SQL rollback work;
  if (sqlca.sqlcode != 0) sqlerror("update: ROLLBACK FAILED",
&sqlca);
  system("touch /tmp/tpcd/update.sync.sleep");

Uexit:
  fprintf(out,"\n---------- END of update ----------\n\n");
  fflush(out);fclose(out);
  return(rc);
}
```

```
/*-------------------------------------------------------------*/
/*      connect_to_TM                                          */
/*-------------------------------------------------------------*/
void connect_to_TM( void )
{
  char *dbname_ptr;
  if ((dbname_ptr = getenv("TPCD_QUAL_DBNAME")) != NULL) {
    fprintf(stderr,"********** %s **********\n",dbname_ptr);
    strcpy (dbname, dbname_ptr);
  }

  EXEC SQL CONNECT TO :dbname IN SHARE MODE;
  if (sqlca.sqlcode <  0) {
    fprintf(stderr, "CONNECT TO %s failed SQLCODE = %d\n",
dbname, sqlca.sqlcode);
    exit(-1);
  }
  return;
}

/*-------------------------------------------------------------*/
/*      disconnect_from_TM                                     */
/*-------------------------------------------------------------*/
void disconnect_from_TM ( void )
{
  EXEC SQL CONNECT RESET;
  if (sqlca.sqlcode <  0) {
    fprintf(stderr, "DISCONNECT failed SQLCODE = %d\n",
sqlca.sqlcode);
    exit(-1);
  }
  return;
}

/*-------------------------------------------------------------*/
/*      sqlerror                                               */
/*-------------------------------------------------------------*/
void sqlerror(char *msg, struct sqlca *psqlca)
{
  FILE *err_fp;


  char  err_fn[256];


  int j,k;

  sprintf(err_fn,
"%s%cacid.sqlerrors",getenv("TPCD_TMP_DIR"),del());
  err_fp=fopen(err_fn,"a");
  fprintf(err_fp,"acid: sqlcode: %4d %s\n", psqlca->sqlcode, msg);
  fprintf(stderr,"acid: sqlcode: %4d %s\n", psqlca->sqlcode, msg);
  fflush(stderr);
  if (psqlca->sqlerrmc[0] != ' ' || psqlca->sqlerrmc[1] != ' ') {
    fprintf(err_fp,"acid: slerrmc:  ");
    for(j = 0; j < 5; j++)
    {
      for(k = 0; k < 14; k++) fprintf(err_fp,"%x ", psqlca-
>sqlerrmc[j*10+k]);
      fprintf(err_fp,"    ");
      for(k = 0; k < 14; k++) fprintf(err_fp,"%c",  psqlca-
>sqlerrmc[j*10+k]);
      fprintf(err_fp,"\n");
      if (j < 4) fprintf(err_fp,"          ");
    }
  }

  fprintf(err_fp,"acid: sqlerrp:  ");
  for(j = 0; j < 8; j++)    fprintf(err_fp,"%c", psqlca->sqlerrp[j]);
  fprintf(err_fp,"\n");
```

```
  fprintf(err_fp,"acid: sqlerrd: ");
  for(j = 0; j < 6; j++)    fprintf(err_fp," %d", psqlca->sqlerrd[j]);
  fprintf(err_fp,"\n");

  if (psqlca->sqlwarn[0] != ' ') {
    fprintf(err_fp,"acid: sqlwarn: ");
    for(j = 0; j < 8; j++)    fprintf(err_fp,"%c ", psqlca->sqlwarn[j]);
    fprintf(err_fp,"\n");
  }

  fprintf(err_fp,"\n");
  fflush(err_fp);fclose(err_fp);
}

#ifdef SQLWINT
void sleep(int sec)
{
 Sleep(sec * 1000);
}
#endif


char del(void)
{
#ifdef SQLWINT
 return '\\';
#else
 return '/';
#endif
}

#if defined(SQLPTX) || defined(SQLWINT) || defined(SQLSUN) ||
defined(Linux)
/* added fot PTX as this one is not there in libm */
double nearest(double x)
{
      double y, z;

      y = x;
      if (x < 0)
            y = -x;
      z = y - (int)y;
      if (z == 0.5) {
            if ((int)floor(y) % 2) {
                  return((x < 0) ? -ceil(y) : ceil(y));
            } else {
                  return((x < 0) ? -floor(y) : floor(y));
            }
      } else  if (z < 0.5)
                  return((x < 0) ? -floor(y) : floor(y));
            else
                  return((x < 0) ? -ceil(y) : ceil(y));
}
#endif /* SQLPTX */
```

E.2 acid.h

```
/**************************************************************************/

/*      File: acid.h                                                      */
/**************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#ifdef SQLWINT
 #include <windows.h>
 #include <sys\timeb.h>
 #include <sys\stat.h>
 #include <stdlib.h>
 #include <io.h>
#else
 #include <unistd.h>
 #include <sys/time.h>
 #include <sys/timeb.h>
#endif

#include <string.h>
#include <math.h>

#define acidtime(tvsec,tvusec) tvsec*1000+tvusec/1000
#define TSLEN 20

#if 0 /* needed on NT, not on AIX */
typedef struct timeval {
      long    tv_sec;         /* seconds */
      long    tv_usec;        /* and microseconds */
};
#endif

struct update_struct {
  int     qnum;
};

struct acidQ_struct {
  int     tag;
  long    o_key;
  double  l_extendedprice;
};

struct acidT_struct {
  int     termination;
  int     tag;
  int     logging;
  long    o_key;
  long    l_key;
  long    delta;
  long    l_partkey;
  long    l_suppkey;
  double  l_quantity;
  double  l_tax;
  double  l_discount;
  double  l_extendedprice;
  double  o_totalprice;
};

/*
** in acid.sqc
*/

int updateQ (struct update_struct *us);

char del(void);

#ifdef SQLWINT
void sleep (int sec);
#endif
```

E.3 Makefile

```
DBNAME =            $(TPCD_QUAL_DBNAME)
#DBNAME             =           TPCD
```

```
INCLUDE =       $(HOME)/sqllib/include

#CFLAGS        =       -I$(INCLUDE) -g -Dpascal= -
DLINT_ARGS \
#       -Dfar= -D_loadds= -DSQLA_NOLINES -qflag=i:i -
qlanglvl=ansi

#LFLAGS  =      -lm -lcurses -ls -ll -ly -liconv -lbsd
CFLAGS =       -I$(INCLUDE) -g -Dpascal= -DLINT_ARGS  \
        -DSQLA_NOLINES -DLinux -m64
# .. sun       -DSQLA_NOLINES

LFLAGS  =      -lm
# sun .... LFLAGS  =           -lm


LIB     =      -L$(HOME)/sqllib/lib -ldb2

#CC     =      g++
CC      =      gcc

HDR     =      acid.h
C    =  mainacid.c
SQC     =      acid.sqc
SRC     =      $(HDR)          $(C)            $(SQC)
OBJ     =      acid.o
EXEC    =      mainacid

TARGET  =      $(EXEC) tsec

.SUFFIXES: .o .c .sqc .bnd

.c.o:
        $(CC) -c $< $(CFLAGS)

all:    $(TARGET)

mainacid: $(SRC) $(OBJ) mainacid.o
        $(CC) -o $@ $(CFLAGS) $(OBJ) mainacid.o $(LIB)
$(LFLAGS)

acid.c: acid.sqc  $(HDR)
        - db2 connect to $(DBNAME); \
        db2 prep acid.sqc BINDFILE ISOLATION RR
NOLINEMACRO PACKAGE; \
        db2 bind acid.bnd GRANT PUBLIC; \
        db2 connect reset; \
        db2 terminate

acid.o: acid.c
        $(CC) $(CFLAGS) -c acid.c -o acid.o

tsec: tsec.c
        $(CC) $(CFLAGS) $(LFLAGS) -o tsec tsec.c

clean:
        rm -f *.o *.bnd $(EXEC) tsec
rm -f acid.c
```

Appendix - F Third Party Pricing

F.1 SUSE Linux Enterprise Server 9

SUSE LINUX Enterprise Server 9

Price List

Reader Rating ★★★½☆ from 21 ratings rate this

tell a friend printer friendly

The following pricing is an excerpt from the VLA price list.

Media

Product Name	Price
SUSE LINUX Enterprise Server 9 Software Media Kit for X86, AMD64, Intel EM64T, Itanium Processor Family, IBM POWER, IBM zSeries and IBM S/390 Strong Encryption (128+ bit) Available in English, German, Portuguese, French, Italian, Spanish, Korean, Simplified Chinese and Japanese	$35

X86 and AMD64 and Intel EM64T

Product Name	Price
SUSE LINUX Enterprise Server 1 server, up to 2 CPUs, per year	$349
SUSE LINUX Enterprise Server 1 server, up to 16 CPUs, per year	$899
SUSE LINUX Enterprise Server Additional 8 CPUs for 1 server, per year	$579

Itanium Processor Family and IBM POWER

Product Name	Price
SUSE LINUX Enterprise Server 1 server, up to 2 CPUs, per year	$689
SUSE LINUX Enterprise Server 1 server, up to 16 CPUs, per year	$1,299
SUSE LINUX Enterprise Server Additional 8 CPUs for 1 server, per year	$799

To get more information, visit Novell SLES 9: http://www.novell.com/products/linuxenterpriseserver/pricing.html.

F.2 Novell Linux Support Services Orders Form

Novell Linux Support Services Order Form

Customer Information

Company Name

Street Address

City | State | ZIP/Postal Code

( ) Phone | ( ) Fax

Contract Number

Bill To

Company Name

Name

Street Address

City | State | ZIP/Postal Code

( ) Phone | ( ) Fax

NOTE: A signature is required on page 2 to activate this contract. Thank you.

IMPORTANT NOTE:

In order to receive support for SUSE Linux products you must be current on your Upgrade Protection (by default MLA customers have this included with maintenance). After Upgrade Protection has been purchased, you must activate the service by registering your activation codes on the SUSE portal at www.suse.com [Go to 'Register Software' link on top right] (MLA customers must also register).

For Office Use Only

Novell Linux Support Services Packages

Prices Effective May 1, 2004

SUPPORT OPTION	PART NUMBER			
SUSE Linux Server Support	MLA: 051002671-001 CLA: 051-002672-001 VLA: 051-002673-001	US$900		
Linux Small Business Support	MLA: 051-002674-001 CLA: 051-002675-001 VLA: 051-002676-001	US$3,800		

To get more information, visit Novell 24x7x4 Support: http://support.novell.com/linux/linux_server_support.html.