TPC Benchmark™ H Full Disclosure Report

for

IBM® System x™ 3650

using

IBM DB2® Universal Database 8.2

Submitted for Review

October 6, 2006

First Edition - October 2006

THE INFORMATION CONTAINED IN THIS DOCUMENT IS DISTRIBUTED ON AN AS IS BASIS WITHOUT ANY WARRANTY EITHER EXPRESSED OR IMPLIED. The use of this information or the implementation of any of these techniques is the customer's responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used.

This publication was produced in the United States. IBM may not offer the products, services, or features discussed in this document in other countries, and the information is subject to change without notice. Consult your local IBM representative for information on products and services available in your area.

© Copyright International Business Machines Corporation 2006. All rights reserved.

Permission is hereby granted to reproduce this document in whole or in part, provided the copyright notice as printed above is set forth in full text on the title page of each item reproduced.

U.S. Government Users - Documentation related to restricted rights: Use, duplication, or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Trademarks

IBM, the IBM logo, DB2, and System x are trademarks or registered trademarks of International Business Machines Corporation.

The following terms used in this publication are trademarks of other companies as follows: TPC Benchmark, TPC-H, QppH QthH and QphH are trademarks of Transaction Processing Performance Council; Intel and Xeon are trademarks or registered trademarks of Intel Corporation; Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Other company, product, or service names, which may be denoted by two asterisks (**), may be trademarks or service marks of others.

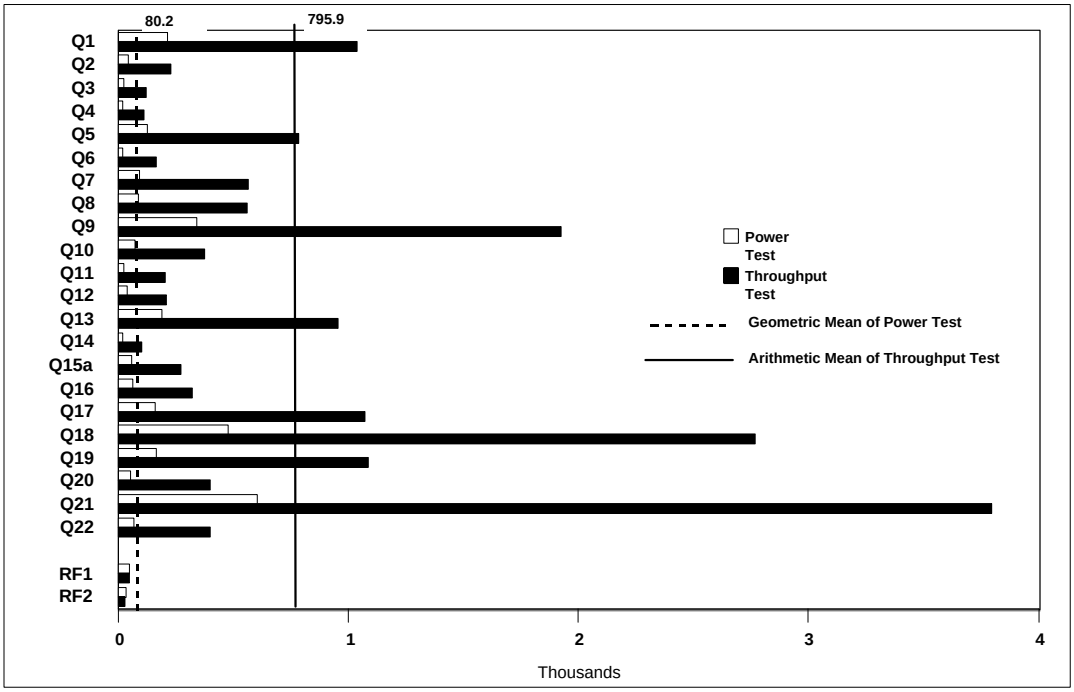
Notes

[1] GHz only measures microprocessor internal clock speed, not application performance. Many factors affect application performance.
[2] When referring to hard disk capacity, one GB equals one billion bytes. Total user-accessible capacity may be less.

IBM Corporation	IBM® System x™ 3650 with IBM DB2® UDB 8.2	TPC-H Rev 2.3.0
		Report Date: Oct. 6, 2006

Total System Cost	Composite Query-per-Hour Metric	Price/Performance
$156,535 USD	**10,165.4 QphH @ 300GB**	**$15.40 USD per QphH @ 300GB**

Database Size	Database Manager	Operating System	Other Software	Availability Date
300GB	**IBM DB2 UDB 8.2**	**SUSE® Linux® Enterprise Server 9 SP3**	**None**	**Oct. 6, 2006**



Database Load Time: 03:08:12	Load Included Backup: Y	Total Data Storage / Database Size: 12.8
RAID (Base Tables Only): N	**RAID (Base Tables and Auxiliary Data Structures): N**	**RAID (All): N**

Configuration		
Processors/Cores/Threads	**2/4/4**	**Intel® Xeon® Processor 5160 (3.0GHz, 4MB L2 Cache, 1333MHz FSB)**
Memory	**4**	**8GB (2 x 4GB) PC2-5300 ECC DDR2 SDRAM RDIMM Kits**
Disk Controllers	**2**	**IBM MegaRAID 8480 SAS Controller**
Disk Drives	**96**	**36.4GB 15K SAS Drive**
	6	**36.4GB 10K SFF SAS Drive**
	2	**73.4GB 10K SFF SAS Drive**
Total Disk Storage		**3838.8GB**

IBM Corporation	IBM System x3650 with IBM DB2 UDB 8.2	TPC-H 2.3.0 Executive Summary
		Report Date: Oct. 6, 2006

Description	Part Number	Third Party Brand	Third Party Pricing	Unit Price	Quantity	Extended Price	3-Yr. Maint. Price
Server Hardware							
IBM System x3650 with 1 Intel Xeon Processor 5160 (3.0GHz/4MB L2 Cache/1333MHz FSB) 2 x 512MB PC2-5300 DDR2 DIMM	7979-7AU	IBM	1	2,739	1	2,739	
Intel Xeon Processor 5160	40K1236	IBM	1	1,279	1	1,279	
8GB (2x4GB) PC2-5300 DDR2 Memory	39M5797	IBM	1	6,529	4	26,116	
IBM 36.4GB 10K 2.5 inch SAS SFF H/S Drive	40K1051	IBM	1	279	6	1,674	
IBM 73.4GB 10K 2.5 inch SAS SFF H/S Drive	40K1052	IBM	1	349	2	698	
IBM MegaRAID 8480 SAS Controller	39R8850	IBM	1	1,199	2	2,398	
PCI-E Riser Card for the x3650		IBM	1	79	1	79	
Display	49387NU	IBM	1	149	1	149	
IBM Preferred Pro USB Keyboard	40K9584	IBM	1	29	1	29	
ScrollPoint 800 DPI Optical Mouse - USB & PS/2	90P0742	IBM	1	25	1	25	
ServicePac for 3-Year 24x7x4 Support (x3650)	21P2078	IBM	1	600	1		600
ServicePac for 3-Year 24x7x4 Support (Display)	30L9183	IBM	1	90	1		90
					Subtotal	35,186	690
Server Storage							
IBM System Storage EXP3000	1727-01X	IBM	1	3,599	8	28,792	
IBM SAS 1m Cable	39R6471	IBM	1	119	4	476	
IBM EXP3000 1m Cable	39R6529	IBM	1	119	4	476	
IBM Hot-Swap 3.5-inch 15K 36.4GB SAS HDD	40K1042	IBM	1	249	96	23,904	
IBM UPS 750TLV	21301TX	IBM	1	299	1	299	
IBM S2 42U Standard Rack	93074RX	IBM	1	1,489	1	1,489	
ServicePac for 3-Year 24x7x4 Support (EXP3000)	41L2768	IBM	1	760	8		6,080
ServicePac for 3-Year 24x7x4 Support (Rack)	41L2760	IBM	1	300	1		300
					Subtotal	55,436	6,380
Server Software							
DB2 UDB ESE 8.2 SW Lic.& Maintenance 12 Months		IBM	2	23,902	2	47,804	
SW Maintenance Renewal - 1 Year		IBM	2	1,138	4		4,552
DPF SW License & Maintenance 12 Months		IBM	2	7,180	2	14,360	
DPF SW Maintenance Renewal - 1 Year		IBM	2	342	4		1,368
SUSE Linux Enterprise Server 9 for x86, AMD64, and Intel EM64T 1-2 Processors w/3-Year Maintenance		Novell	3	4,279	1	4,279	
					Subtotal	66,443	5,920
					Total	$157,065	$12,990
Large Purchase Discount (See Note 1.)			1	13.84%	**Discount**	13,520	

Pricing: 1 - 1-888-Shop-IBM, ext. 5821 ; 2 - IBM; 3 - Novell
Note 1: Pricing is for this system or one of similar size.

Audited by Francois Raab, InfoSizing, Inc. (www.sizing.com)

Three-Year Cost of Ownership USD:	$156,535
QphH@300GB:	**10,165.4**
$ USD/QphH@300GB:	**$15.40**

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

IBM Corporation	IBM System x3650 with IBM DB2 UDB 8.2	TPC-H Rev 2.3.0
		Report Date: Oct. 6, 2006

Measurement Results:

Database Scale Factor	300
Total Data Storage/Database Size	12.8
Start of Database Load	17:03:57
End of Database Load	20:12:09
Database Load Time	03:08:12
Query Streams for Throughput Test	6
TPC-H Power	13,467.1
TPC-H Throughput	7,673.2
TPC-H Composite Query-per-Hour (QphH@300GB)	10,165.4
Total System Price over 3 Years	$156,535 USD
TPC-H Price/Performance Metric ($/QphH@300GB)	$15.40 USD

Measurement Interval:

Measurement Interval in Throughput Test (Ts) = 18,579

Duration of Stream Execution:

	Seed	Query Start Date/Time Query End Date/Time	RF1 Start Date/Time RF1 End Date/Time	RF2 Start Date/Time RF2 End Date/Time	Duration
Stream 00	929201209	09/30/06 02:39:09 09/30/06 03:31:00	09/30/06 02:39:09 09/30/06 02:40:01	09/30/06 03:30:25 09/30/06 03:31:00	00:51:51
Stream 01	929201210	09/30/06 03:31:03 09/30/06 08:31:26	09/30/06 08:32:42 09/30/06 08:33:33	09/30/06 08:33:33 09/30/06 08:34:07	05:00:22
Stream 02	929201211	09/30/06 03:31:03 09/30/06 08:28:31	09/30/06 08:34:07 09/30/06 08:34:54	09/30/06 08:34:54 09/30/06 08:35:23	04:57:28
Stream 03	929201212	09/30/06 03:31:03 09/30/06 08:32:43	09/30/06 08:35:23 09/30/06 08:36:10	09/30/06 08:36:10 09/30/06 08:36:40	05:01:40
Stream 04	929201213	09/30/06 03:31:03 09/30/06 08:02:16	09/30/06 08:36:40 09/30/06 08:37:33	09/30/06 08:37:33 09/30/06 08:38:09	04:31:14
Stream 05	929201214	09/30/06 03:31:03 09/30/06 08:23:17	09/30/06 08:38:09 09/30/06 08:38:56	09/30/06 08:38:56 09/30/06 08:39:26	04:52:13
Stream 06	929201215	09/30/06 03:31:03 09/30/06 08:19:11	09/30/06 08:39:26 09/30/06 08:40:12	09/30/06 08:40:12 09/30/06 08:40:42	04:48:08

IBM Corporation	IBM System x3650 with IBM DB2 UDB 8.2	TPC-H Rev 2.3.0
		Report Date: Oct. 6, 2006

TPC-H Timing Intervals (in seconds):

Query	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Stream 00	215.8	48.7	24.6	24.0	126.9	24.2	93.6	92.0	342.0	76.9	28.7	41.3
Stream 01	395.2	257.9	178.1	31.4	1,354.9	181.9	522.4	614.8	1,920.1	313.9	190.8	338.0
Stream 02	1,144.9	223.4	127.4	138.6	848.6	206.3	464.4	421.5	2,134.8	386.3	183.2	219.1
Stream 03	1,344.4	202.2	23.7	123.4	648.7	154.6	608.2	614.8	1,793.2	374.2	240.2	41.3
Stream 04	1,186.8	258.0	142.3	94.3	594.6	96.9	540.1	506.9	1,894.0	370.6	202.4	162.8
Stream 05	1,158.8	198.3	151.2	148.5	506.9	190.8	653.6	626.4	1,726.1	345.2	208.4	273.0
Stream 06	999.7	255.0	121.2	148.3	771.5	172.8	611.4	582.6	2,091.8	475.9	216.2	226.7
Minimum	395.2	198.3	23.7	31.4	506.9	96.9	464.4	421.5	1,726.1	313.9	183.2	41.3
Average	1,038.3	232.5	124.0	114.1	787.5	167.2	566.7	561.2	1,926.7	377.7	206.9	210.2
Maximum	1,344.4	258.0	178.1	148.5	1,354.9	206.3	653.6	626.4	2,134.8	475.9	240.2	338.0

Stream ID	Q13	Q14	Q15a	Q16	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream 00	192.3	22.5	60.5	63.4	165.4	478.4	170.0	55.1	604.8	72.8	51.2	35.3
Stream 01	1,029.8	89.4	222.1	195.4	1,023.1	3,257.0	1,198.3	407.6	4,050.2	250.2	50.0	33.9
Stream 02	1,081.8	94.0	412.7	211.1	1,208.6	3,235.3	1,215.0	426.5	3,061.3	403.2	47.2	29.1
Stream 03	1,083.3	111.4	271.8	130.0	1,198.7	3,559.2	1,163.2	341.6	3,669.9	401.7	46.6	29.8
Stream 04	752.2	88.7	283.0	264.7	768.3	2,144.1	1,061.2	432.1	4,131.2	298.3	53.4	35.9
Stream 05	1,006.1	87.0	240.3	846.6	1,193.7	1,960.0	780.1	429.9	4,073.8	728.5	46.6	30.0
Stream 06	800.3	146.7	224.3	287.3	1,059.9	2,469.8	1,122.6	382.1	3,804.7	317.4	46.3	30.1
Minimum	752.2	87.0	222.1	130.0	768.3	1,960.0	780.1	341.6	3,061.3	250.2	46.3	29.1
Average	958.9	102.9	275.7	322.5	1,075.4	2,770.9	1,090.1	403.3	3,798.5	399.9	48.4	31.5
Maximum	1,083.3	146.7	412.7	846.6	1,208.6	3,559.2	1,215.0	432.1	4,131.2	728.5	53.4	35.9

Table of Contents

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 2.3.0) August 11, 2002. This is the full diclosure report for benchmark testing of the IBM System x3650 according to the TPC Benchmark H Standard Specification.

The TPC Benchmark H is a decision support benchmark. It consists of a suite of business-oriented ad hoc queries and concurrent data modifications. The queries and the data populating the database have been chosen to have broad industrywide 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 set 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 with 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, exeat 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 ongoing refresh functions, which batch together a number of modifications impacting some part of the decision support database.
- Due to the worldwide 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 10 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 metrics 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.

Benchmarks 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 specific customer application benchmarking when critical capacity planning and/or product evaluation decisions are contemplated.

1 General Items

1.1 Benchmark Sponsor

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

IBM Corporation sponsored this TPC-H benchmark.

1.2 Parameter Settings

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

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

Appendix A, “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.

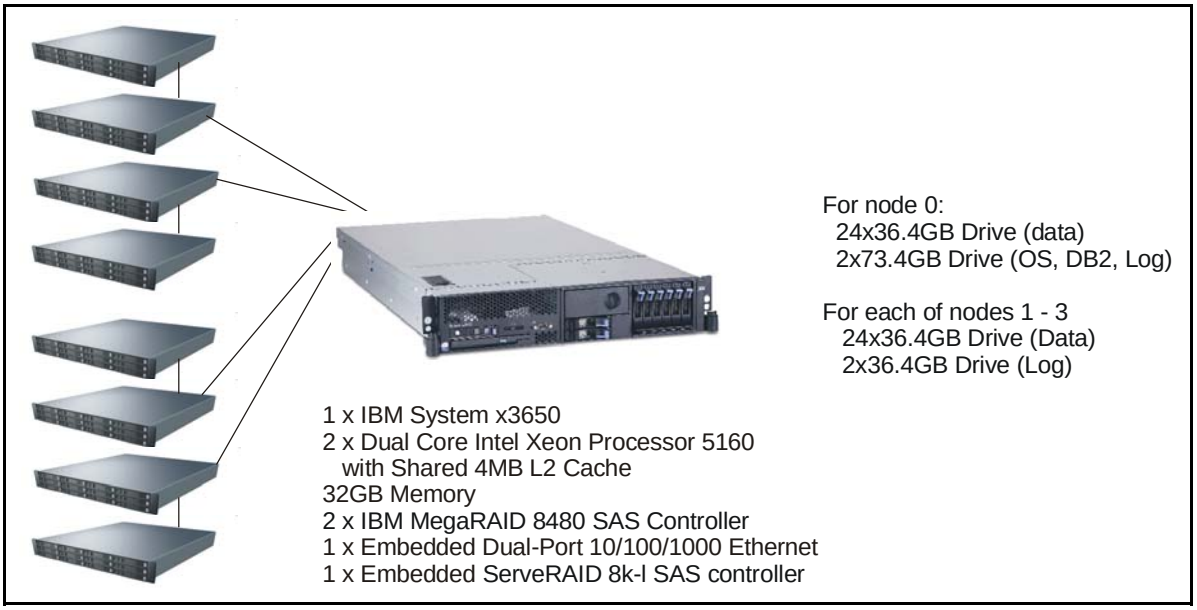
1.3 Configuration Diagrams

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 their protocol type*
- *Number of LAN (e.g., Ethernet) connections, including routers, workstations, terminals, etc., that were physically used in the test or are incorporated into the pricing structure*
- *Type and run-time execution location of software components (e.g., DBMS, query processing tools/languages, middleware components, software drivers, etc.).*

The configuration diagram for the tested and priced system is provided on the following page.

1.3.1 Priced and Measured Configurations



The priced configuration for the x3650 contained:

- Two Intel Xeon Processor 5160 (3.0GHz, 4MB L2 cache, 1333MHz FSB)
- Eight 4GB PC2-5300 DDR2 Fully-Buffered Memory DIMMs
- One embedded dual-port Gigabit Ethernet interface
- One embedded ServeRAID 8k-l controller (attached to 8 internal disks for OS, DB2, Swap and database logs)
- Two IBM MegaRAID 8480 SAS controllers (attached to database data disks)
- Ninety-six (96) 36.4GB 3.5inch 15K SAS disk drives
- Eight (8) EXP3000 Storage Expansion Enclosures
- Six (6) 36.4GB 2.5 inch 10K SAS disk drives
- Two (2) 73.4GB 2.5 inch 10K SAS disk drives

The measured configuration and the priced configuration were identical.

2 Clause 1: Logical Database Design Related Items

2.1 Database Table Definitions

Listings must be provided for all table definition statements and all other statements used to set up the test and qualification databases. (8.1.2.1)

Appendix B contains the scripts that were used to set up the TPC-H test and qualification databases.

2.2 Database Organization

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

Appendix B contains the scripts that were used to create the indexes on the test and qualification databases.

2.3 Horizontal Partitioning

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

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).

Replication was not used.

3 Clause 2: Queries and Update Functions Related Items

3.1 Query Language

The query language used to implement the queries must be identified.

SQL was the query language used.

3.2 Random Number Generation

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

The TPC-supplied DBGEN version 2.3.0 and QGEN version 2.3.0 were used to generate all database populations.

3.3 Substitution Parameters Generation

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

The supplied QGEN version 2.3.0 was used to generate the substitution parameters.

3.4 Query Text and Output Data from Database

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

Appendix C contains the output for each of the qualification queries. The functional query definitions and variants used in this disclosure use the following minor query modifications:

- Table names and view names are fully qualified. For example, the nation table is referred to as "TPCD.NATION."
- The standard IBM SQL date syntax is used for date arithmetic. For example, DATE('1996-01-01')+3 MONTHS.
- The semicolon (;) is used as a command delimiter.

3.5 Query Substitution Parameters and Seeds Used

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 C contains the seed and query substitution parameters used.

3.6 Query Isolation Level

The isolation level used to run the queries must be disclosed. If the isolation level does not map closely to 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 Function Implementation

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 Clause 3: Database System Properties Related Items

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 specifications. The Atomicity, Isolation, Consistency and Durability tests were performed on the IBM System x3650 server. Appendix E contains the ACID transaction source code.

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 Transactions

Perform the ACID transactions 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 randomly selected order key. The number of records in the HISTORY table was also retrieved.
2. The ACID Transaction T1 was executed for the order key used in step 1.
3. The total price and extended price were retrieved for the same order key 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 transactions 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 the aborted ACID transaction:

1. The ACID application is passed a parameter to execute a rollback of the transaction instead of performing the commit.
2. The total price from the ORDER table and the extended price from the LINEITEM table were retrieved for a random order key. The number of records in the HISTORY table was also retrieved.
3. The ACID transaction was executed for the orderkey used in step 2. The transaction was rolled back.
4. The total price and the extended price were retrieved for the same orderkey used in step 2 and step 3. It was verified that the extended price and the total price were the same as in step 2.

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,20,2)),20,3)*

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

FROM TPCD.LINEITEM WHERE L_ORDEYKEY=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 was in a consistent condition.

After executing 7 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. First 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. Second session: Start an ACID query for the same O_KEY as in the ACID transaction.
3. Second session: The ACID query attempts to read the file but is locked out by the ACID transaction waiting to complete.
4. First 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. Second 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. First 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. Second 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. First session: The ACID transaction is released and the Rollback is executed, releasing the read.

4. Second 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. First 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. Second 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. First 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. First 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. Second 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. First 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. First 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. First 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. First session: A transaction T1, which executes modified TPC-H query 1 with DELTA=0, was started.
2. Second session: Before T1 completed, an ACID transaction T2, with randomly selected values of O_KEY, L_KEY and DELTA, was started.
3. Third 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 Failure of a Durable Medium

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

The database log was stored on RAID-1 protected storage. The tables for the database were stored on RAID-0 storage, with the exception of the Nation and Region tables, which were stored on internal RAID-1 protected storage. A backup of the database was taken to a separate software RAID-5 protected array of drives than those used for the database data being backed up.

The tests were conducted on the qualification database. The steps performed are shown below.

1. The complete database was backed up once to Linux software RAID-5 protected storage . The backup of the data was not on the same drive as the data itself.
2. Seven streams of ACID transactions were started. Each stream executed a minimum of 100 transactions.
3. One physical drive of a RAID-0 data volume was removed.
4. The seven streams of ACID transactions failed and recorded their number of committed transactions in success files.
5. The failed disk was replaced with a new drive. The database data partitions containing the failed disk were recreated. The system was rebooted.
6. A database restore was issued using the backup taken at the beginning of this test.
7. A command was issued causing the database to run through its roll-forward recovery.
8. The counts in the success files and the HISTORY table count were compared and were found to match.
9. The software RAID-5 backup partition which lost a member during the loss of data test was restored to its RAID-5 protected status.

4.4.2 Loss of Log and System Crash

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

1. Seven streams of ACID transactions were started. Each stream executed a minimum of 100 transactions.
2. While the test was running, one of the disks from the database RAID-1 log on Logical Node 2 was removed.
3. The test continued running for approximately an additional 30 transactions per stream.

4. Then the system was powered off.
5. When the power was restored, the system rebooted and the database was restarted.
6. The database went through a recovery period.
7. The success file and the HISTORY table counts were compared and found to match.
8. The database log disk was replaced and a rebuild function was initiated to restore the RAID-1 log array to its protected status. The rebuild completed successfully.

4.4.3 System Crash

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

This test was combined with the Loss of Log test. See the previous section.

4.4.4 Memory Failure

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

See the previous section.

5 Clause 4: Scaling and Database Population Related Items

5.1 Cardinality of Tables

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

Table Name	Rows
Order	450,000,000
Lineitem	1,799,989,091
Customer	45,000,000
Part	60,000,000
Supplier	3,000,000
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 described.

The following series of tables shows the distribution of tables and logs across all media.

Controller	Drives	Logical Node/Partition	Size	Use
ServeRAID 8k-I (Embedded RAID controller)	2 - 73.4GB RAID-1	/dev/sda1	100MB	Boot Sector
		/dev/sda2	16GB	Linux Swap
		/dev/sda3	30GB	OS; DB; Kit; small_data
		/dev/sda4	22.1GB	LN0 DB log
			5850KB	unused
	2 36.4GB RAID-1	/dev/sdb1	22.1GB	LN1 DB log
		/dev/sdb2	11.12GB	unused
	2 36.4GB RAID-1	/dev/sdc1	22.1GB	LN2 DB log
		/dev/sdc2	11.12GB	Unused
	2 36.4GB RAID-1	/dev/sdd1	22.1GB	LN3 DB log
		/dev/sdd2	11.12GB	Unused
MegaRAID-1	6 - 36.4GB RAID-0	Logical Node 0		
		/dev/sde1	195.6GB	Extended Partition
		/dev/sde5	12GB	Temp Tables
		/dev/sde6	3.1GB	Lineitem Indexes
		/dev/sde7	17.5GB	Lineitem Data
		/dev/sde8	8GB	Other Tables
		/dev/sde9	1.8GB	Other Indexes
		/dev/sde10	34.3GB	Temp Tables
		/dev/sde11	72GB	/backup3; OS RAID-5 (Bkup of LN3)
			46.8GB	Unused
	6 - 36.4GB RAID-0	Logical Node 0		
		/dev/sdf1	195.6GB	Extended Partition
		/dev/sdf5	12GB	Temp Tables
		/dev/sdf6	3.1GB	Lineitem Indexes
		/dev/sdf7	17.5GB	Lineitem Data
		/dev/sdf8	8GB	Other Tables
		/dev/sdf9	1.8GB	Other Indexes
		/dev/sdf10	34.3GB	Temp Tables
		/dev/sdf11	72GB	/backup3; OS RAID-5 (Bkup of LN3)
			46.8GB	Unused
	6 - 36.4GB RAID-0	Logical Node 0		
		/dev/sdg1	195.6GB	Extended Partition
		/dev/sdg5	12GB	Temp Tables
		/dev/sdg6	3.1GB	Lineitem Indexes
		/dev/sdg7	17.5GB	Lineitem Data
		/dev/sdg8	8GB	Other Tables
		/dev/sdg9	1.8GB	Other Indexes
		/dev/sdg10	34.3GB	Temp Tables
		/dev/sdg11	72GB	/backup3; SW RAID-5 (Bkup of LN3)
			46.8GB	Unused
	6 - 36.4GB RAID-0	Logical Node 0		
		/dev/sdh1	195.6GB	Extended Partition
		/dev/sdh5	12GB	Temp Tables
		/dev/sdh6	3.1GB	Lineitem Indexes
		/dev/sdh7	17.5GB	Lineitem Data
		/dev/sdh8	8GB	Other Tables
		/dev/sdh9	1.8GB	Other Indexes
		/dev/sdh10	34.3GB	Temp Tables
		/dev/sdh11	72GB	/dev/md4;OS RAID-0;Flatfile data
			46.8GB	Unused

Controller	Drives	Logical Node/Partition	Size	Use
MegaRAID-1	6 - 36.4GB RAID-0	Logical Node 1 /dev/sdi1 /dev/sdi5 /dev/sdi6 /dev/sdi7 /dev/sdi8 /dev/sdi9 /dev/sdi10 /dev/sdi11	195.6GB 12GB 3.1GB 17.5 8GB 1.8GB 34.3GB 72GB 46.8GB	Extended Partition Temp Tables Lineitem Indexes Lineitem Data Other Tables Other Indexes Temp Tables /backup2; OS RAID-5 (Bkup of LN2) Unused
	6 - 36.4GB RAID-0	Logical Node 1 /dev/sdj1 /dev/sdj5 /dev/sdj6 /dev/sdj7 /dev/sdj8 /dev/sdj9 /dev/sdj10 /dev/sdj11	195.6GB 12GB 3.1GB 17.5GB 8GB 1.8GB 34.3GB 72GB 46.8GB	Extended Partition Temp Tables Lineitem Indexes Lineitem Data Other Tables Other Indexes Temp Tables /backup2; OS RAID-5 (Bkup of LN2) Unused
	6 - 36.4GB RAID-0	Logical Node 1 /dev/sdk1 /dev/sdk5 /dev/sdk6 /dev/sdk7 /dev/sdk8 /dev/sdk9 /dev/sdk10 /dev/sdk11	195.6GB 12GB 3.1GB 17.5GB 8GB 1.8GB 34.3GB 72GB 46.8GB	Extended Partition Temp Tables Lineitem Indexes Lineitem Data Other Tables Other Indexes Temp Tables /backup2; SW RAID-5 (Bkup of LN2) Unused
	6 - 36.4GB RAID-0	Logical Node 1 /dev/sdl1 /dev/sdl5 /dev/sdl6 /dev/sdl7 /dev/sdl8 /dev/sdl9 /dev/sdl10 /dev/sdl11	195.6GB 12GB 3.1GB 17.5GB 8GB 1.8GB 34.3GB 72GB 46.8GB	Extended Partition Temp Tables Lineitem Indexes Lineitem Data Other Tables Other Indexes Temp Tables /dev/md4;OS RAID-0;Flatfile data Unused

Controller	Drives	Logical Node/Partition	Size	Use
MegaRAID-2	6 - 36.4GB	Logical Node 2		
	RAID-0	/dev/sdm1	195.6GB	Extended Partition
		/dev/sdm5	12GB	Temp Tables
		/dev/sdm6	3.1GB	Lineitem Indexes
		/dev/sdm7	17.5GB	Lineitem Data
		/dev/sdm8	8GB	Other Tables
		/dev/sdm9	1.8GB	Other Indexes
		/dev/sdm10	34.3GB	Temp Tables
		/dev/sdm11	72GB	/backup0; OS RAID-5 (Bkup of LN0)
			46.8GB	Unused
	6 - 36.4GB	Logical Node 2		
	RAID-0	/dev/sdn1	195.6GB	Extended Partition
		/dev/sdn5	12GB	Temp Tables
		/dev/sdn6	3.1GB	Lineitem Indexes
		/dev/sdn7	17.5GB	Lineitem Data
		/dev/sdn8	8GB	Other Tables
		/dev/sdn9	1.8GB	Other Indexes
		/dev/sdn10	34.3GB	Temp Tables
		/dev/sdn11	72GB	/backup0; OS RAID-5 (Bkup of LN0)
			46.8GB	Unused
	6 - 36.4GB	Logical Node 2		
	RAID-0	/dev/sdo1	195.6GB	Extended Partition
		/dev/sdo5	12GB	Temp Tables
		/dev/sdo6	3.1GB	Lineitem Indexes
		/dev/sdo7	17.5GB	Lineitem Data
		/dev/sdo8	8GB	Other Tables
		/dev/sdo9	1.8GB	Other Indexes
		/dev/sdo10	34.3GB	Temp Tables
		/dev/sdo11	72GB	/backup0; SW RAID-5 (Bkup of LN0)
			46.8GB	Unused
	6 - 36.4GB	Logical Node 2		
	RAID-0	/dev/sdp1	195.6GB	Extended Partition
		/dev/sdp5	12GB	Temp Tables
		/dev/sdp6	3.1GB	Lineitem Indexes
		/dev/sdp7	17.5GB	Lineitem Data
		/dev/sdp8	8GB	Other Tables
		/dev/sdp9	1.8GB	Other Indexes
		/dev/sdp10	34.3GB	Temp Tables
		/dev/sdp11	72GB	/dev/md4;OS RAID-0;Flatfile data
			46.8GB	Unused

Controller	Drives	Logical Node/Partition	Size	Use
MegaRAID-2	6 - 36.4GB RAID-0	Logical Node 3 /dev/sdq1 /dev/sdq5 /dev/sdq6 /dev/sdq7 /dev/sdq8 /dev/sdq9 /dev/sdq10 /dev/sdq11	195.6GB 12GB 3.1GB 17.5 8GB 1.8GB 34.3GB 72GB 46.8GB	Extended Partition Temp Tables Lineitem Indexes Lineitem Data Other Tables Other Indexes Temp Tables /backup1; OS RAID-5 (Bkup of LN1) Unused
	6 - 36.4GB RAID-0	Logical Node 3 /dev/sdr1 /dev/sdr5 /dev/sdr6 /dev/sdr7 /dev/sdr8 /dev/sdr9 /dev/sdr10 /dev/sdr11	195.6GB 12GB 3.1GB 17.5GB 8GB 1.8GB 34.3GB 72GB 46.8GB	Extended Partition Temp Tables Lineitem Indexes Lineitem Data Other Tables Other Indexes Temp Tables /backup1; OS RAID-5 (Bkup of LN1) Unused
	6 - 36.4GB RAID-0	Logical Node 3 /dev/sds1 /dev/sds5 /dev/sds6 /dev/sds7 /dev/sds8 /dev/sds9 /dev/sds10 /dev/.sds11	195.6GB 12GB 3.1GB 17.5GB 8GB 1.8GB 34.3GB 72GB 46.8GB	Extended Partition Temp Tables Lineitem Indexes Lineitem Data Other Tables Other Indexes Temp Tables /backup1; SW RAID-5 (Bkup of LN1) Unused
	6 - 36.4GB RAID-0	Logical Node 3 /dev/sdt1 /dev/sdt5 /dev/sdt6 /dev/sdt7 /dev/sdt8 /dev/sdt9 /dev/sdt10	195.6GB 12GB 3.1GB 17.5GB 8GB 1.8GB 34.3GB 123GB	Extended Partition Temp Tables Lineitem Indexes Lineitem Data Other Tables Other Indexes Temp Tables Unused

The priced configuration used 104 disks. The 300GB database flatfiles were stored on a software RAID-0 partition stiped across the database data drives.

5.3 Database Partition / Replication Mapping

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

The database was not replicated. The database was logically partitioned into four logical nodes.

5.4 RAID Implementation

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

RAID-1 was used for log disks and the Operating System/Database install disk. RAID-0 was used for all other database disks, the temporary tablespace, and the OS swap space. The Nation and Region tables were placed on the Operating System/Database install disk (dev/sda3).

5.5 DBGEN Modifications

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 DBGEN version 2.3.0 was used for database population. No modifications were made.

5.6 Database Load Time

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

See the Executive Summary at the beginning of this report.

5.7 Data Storage Ratio

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

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

Disk Type	Number of Disks	Formatted Space per Disk	Total Disk Space	Scale Factor	Storage Ratio
36.4GB 15K Ultra320 SAS Drive	96	36.2GB	3475.2GB		
36.4GB 10K 2.5 inch SAS SFF H/S Drive	6	36.2GB	217.2GB		
73.4GB 10K 2.5 inch SAS SFF H/S Drive	2	73.2GB	146.4GB		
Total			3838.8GB	300GB	12.8

The data storage ratio is 12.80, derived by dividing 3838.8GB by the database size of 300GB.

5.8 Database Load Mechanism Details and Illustration

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

Flat files for each of the tables were created using DBGEN.

The NATION and REGION tables were created in /data/small_tables and then loaded from dbgen output. The other tables were loaded on the four logical nodes.

The tables were loaded as depicted in Figure 4-1.

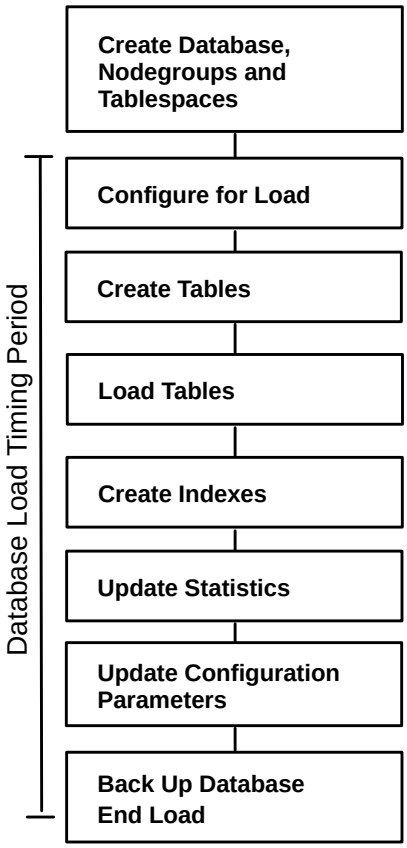


Figure 4-1. Database Load Procedure

5.9 Qualification Database Configuration

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

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

6 Clause 5: Performance Metrics and Execution Rules Related Items

6.1 System Activity between Load and Performance Tests

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

The auditor requested that queries be run against the database to verify the 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 reboot, database restart) must be disclosed.

The following steps were used to implement the power test:

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

6.3 Timing Intervals for Each Query and Refresh Function

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

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report.

6.4 Number of Streams for the Throughput Test

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

Six streams were used for the throughput test.

6.5 Start and End Date/Times for Each Query Stream

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

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report.

6.6 Total Elapsed Time for the Measurement Interval

The total elapsed time for the measurement interval must be reported for the throughput test.

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report..

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

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

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report.

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

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

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report.

6.9 Performance Metrics

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

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report.

6.10 Performance Metric and Numerical Quantities from Both Runs

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

Two consecutive runs of the TPC-H benchmark were performed. The following table contains the results for both runs.

	QppH @ 300GB	QthH @ 300GB	QphH @ 300GB
Run1	13,478.3	7,672.8	10,169.4
Run2	13,467.1	7,673.2	10,165.4

6.11 System Activity between Tests

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

DB2 was restarted between runs.

7 Clause 6: SUT and Driver Implementation Related Items

7.1 Driver

A detailed textual description of how the driver performs its functions, how its various components interact and any product functionality or environmental setting on which it relies must be provided. All related source code, scripts and configurations must be disclosed. The information provided should be sufficient for 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, an indication that the refresh functions must be run, and the SQL file that contains the power stream queries.

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 n sets of temporary tables (one each for lineitem and orders). Then tpcdbatch forks off n children to do an insert with subselect 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

Profile-directed optimization was not used.

8 Clause 7: Pricing Related Items

8.1 Hardware and Software Components

A detailed list of the 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 either general availability status or committed delivery date. If package-pricing is used, contents of the package must be disclosed. Pricing source(s) and effective date(s) must also be reported.

A detailed list of all hardware and software, including the 3-year price, is provided in the Executive Summary at the front of this report. The price quotations are included in Appendix F.

8.2 Three-Year Cost of System Configuration

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

A detailed list of all hardware and software, including the 3-year price, is provided in the Executive Summary at the front of this report. The price quotations are included in Appendix F.

8.3 Availability Dates

The committed delivery date for general availability (availability date) of products used in the price calculations must be reported. When the priced system includes products with different availability dates, availability date reported on the Executive Summary must be the date by which all components are committed to being available. The Full Disclosure Report must report availability dates individually for at least each of the categories for which a pricing subtotal must be provided (see Clause 7.3.1.3).

The System Availability Date is October 6, 2006.

8.4 Country-Specific Pricing

Additional Clause 7 related items may be included in the Full Disclosure Report for each country-specific priced configuration. Country-specific pricing is subject to Clause 7.1.7.

The configuration is priced for the United States of America.

Clause 9: Audit Related Items

9.1 Auditor's Report

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

This implementation of the TPC Benchmark H was audited by Francois Raab of InfoSizing, Inc. (www.sizing.com). For a copy of this disclosure, go to www.tpc.org.

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

Benchmark Sponsor: Celia Schreiber
Manager, System x Performance
Analysis and Benchmarking
3039 Cornwallis Road
Research Triangle Park NC, 27709

October 5, 2006

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

Platform: **IBM® System x™ 3650**

Database Manager: **IBM DB2® UDB 8.2**

Operating System: **SUSE® Linux® Enterprise Server 9 SP3**

The results were:

CPU (Speed)	Memory	Disks	**QphH@300GB**
IBM® System x™ 3650			
2 x Intel Xeon 5160 (3.0 GHz)	4 MB Cache/CPU 32 GB Main	96 x 36.4 GB 15k 6 x 36.4 GB 10k 2 x 73.4 GB 10k	**10,165.4**

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

- The database records were defined with the proper layout and size
- The database population was generated using DBGEN
- The database was properly scaled to 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 one 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:

None.

Respectfully Yours,

François Raab
President

Appendix A: Tunable Parameters and System Configuration

DB2 UDB 8.2 Database Manager Configuration

```
*** Database Manager Config ***get dbm cfg

        Database Manager Configuration

    Node type = Enterprise Server Edition with local and remote clients

 Database manager configuration release level            = 0x0a00

 CPU speed (millisec/instruction)             (CPUSPEED) = 1.889377e-07
 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-amd64-142

 Diagnostic error capture level             (DIAGLEVEL) = 0
 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) = 1250000

 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) = 256
 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) = RESTART

 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) = db2c_db2inst1
 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) = 30000
 Number of FCM request blocks             (FCM_NUM_RQB) = AUTOMATIC
 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
```

DB2 UDB 8.2 Database Configuration (Nodes 0-3)

Node 0

```
*** Database Configuration for Node:  0
get db cfg 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) =
```

```
Database page size                          = 4096

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) = 363900
Catalog cache size (4KB)              (CATALOGCACHE_SZ) = 386
Log buffer size (4KB)                      (LOGBUFSZ) = 2048
Utilities heap size (4KB)                (UTIL_HEAP_SZ) = 40000
Buffer pool size (pages)                   (BUFFPAGE) = 128000
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) = 16384

Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 2048
Percent of mem for appl. group heap    (GROUPHEAP_RATIO) = 70
Max appl. control heap size (4KB)      (APP_CTL_HEAP_SZ) = 2048

Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = (SHEAPTHRES)
Sort list heap (4KB)                       (SORTHEAP) = 18432
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) = 25
Lock timeout (sec)                         (LOCKTIMEOUT) = -1

Changed pages threshold                 (CHNGPGS_THRESH) = 60
Number of asynchronous page cleaners    (NUM_IOCLEANERS) = 1
Number of I/O servers                   (NUM_IOSERVERS) = 4
Index sort flag                            (INDEXSORT) = YES
Sequential detect flag                     (SEQDETECT) = YES
Default prefetch size (pages)           (DFT_PREFETCH_SZ) = AUTOMATIC

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) = 30
Number of secondary log files              (LOGSECOND) = 2
Changed path to log files                  (NEWLOGPATH) =
Path to log files                           = /dev/sda4
Overflow log path                          (OVERFLOWLOGPATH) =
Mirror log path                            (MIRRORLOGPATH) =
First active log file                       = S0000066.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) = 1600
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) = SYSTEM (RESTART)
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


valiant: db2 get db cfg for tpcd completed ok
```

Node 1

```
*** Database Configuration for Node:  1
get db cfg 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) =
Database page size                           = 4096

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) = 363900
Catalog cache size (4KB)             (CATALOGCACHE_SZ) = 386
Log buffer size (4KB)                    (LOGBUFSZ) = 2048
Utilities heap size (4KB)              (UTIL_HEAP_SZ) = 40000
Buffer pool size (pages)                 (BUFFPAGE) = 128000
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) = 16384

Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 2048
Percent of mem for appl. group heap   (GROUPHEAP_RATIO) = 70
Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 2048

Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = (SHEAPTHRES)
Sort list heap (4KB)                     (SORTHEAP) = 18432
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) = 25
Lock timeout (sec)                      (LOCKTIMEOUT) = -1

Changed pages threshold                (CHNGPGS_THRESH) = 60
Number of asynchronous page cleaners   (NUM_IOCLEANERS) = 1
Number of I/O servers                   (NUM_IOSERVERS) = 4
Index sort flag                           (INDEXSORT) = YES
Sequential detect flag                    (SEQDETECT) = YES
Default prefetch size (pages)         (DFT_PREFETCH_SZ) = AUTOMATIC

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) = 30
Number of secondary log files            (LOGSECOND) = 2
Changed path to log files                (NEWLOGPATH) =
Path to log files                            = /dev/sdb1
Overflow log path                   (OVERFLOWLOGPATH) =
Mirror log path                          (MIRRORLOGPATH) =
First active log file                        = S0000068.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) = 1600
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) = SYSTEM (RESTART)
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


valiant: db2 get db cfg for tpcd completed ok
```

Node 2

```
*** Database Configuration for Node:  2
get db cfg 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) =
Database page size                               = 4096

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) = 363900
Catalog cache size (4KB)              (CATALOGCACHE_SZ) = 386
Log buffer size (4KB)                        (LOGBUFSZ) = 2048
Utilities heap size (4KB)                (UTIL_HEAP_SZ) = 40000
Buffer pool size (pages)                     (BUFFPAGE) = 128000
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) = 16384

Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 2048
Percent of mem for appl. group heap   (GROUPHEAP_RATIO) = 70
Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 2048

Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = (SHEAPTHRES)
Sort list heap (4KB)                         (SORTHEAP) = 18432
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) = 25
Lock timeout (sec)                      (LOCKTIMEOUT) = -1

Changed pages threshold                 (CHNGPGS_THRESH) = 60
Number of asynchronous page cleaners   (NUM_IOCLEANERS) = 1
Number of I/O servers                   (NUM_IOSERVERS) = 4
Index sort flag                            (INDEXSORT) = YES
Sequential detect flag                     (SEQDETECT) = YES
Default prefetch size (pages)         (DFT_PREFETCH_SZ) = AUTOMATIC

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) = 30
Number of secondary log files              (LOGSECOND) = 2
Changed path to log files                  (NEWLOGPATH) =
Path to log files                                = /dev/sdc1
Overflow log path                       (OVERFLOWLOGPATH) =
Mirror log path                         (MIRRORLOGPATH) =
First active log file                            = S0000069.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) = 1600
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) = SYSTEM (RESTART)
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



valiant: db2 get db cfg for tpcd completed ok
```

Node 3

```
*** Database Configuration for Node:  3
get db cfg 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) =
Database page size                         = 4096

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) = 363900
Catalog cache size (4KB)             (CATALOGCACHE_SZ) = 386
Log buffer size (4KB)                     (LOGBUFSZ) = 2048
Utilities heap size (4KB)              (UTIL_HEAP_SZ) = 40000
Buffer pool size (pages)                  (BUFFPAGE) = 128000
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) = 16384

Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 2048
Percent of mem for appl. group heap   (GROUPHEAP_RATIO) = 70
Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 2048

Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = (SHEAPTHRES)
Sort list heap (4KB)                       (SORTHEAP) = 18432
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) = 25
Lock timeout (sec)                      (LOCKTIMEOUT) = -1

Changed pages threshold                (CHNGPGS_THRESH) = 60
Number of asynchronous page cleaners   (NUM_IOCLEANERS) = 1
Number of I/O servers                  (NUM_IOSERVERS) = 4
Index sort flag                           (INDEXSORT) = YES
Sequential detect flag                    (SEQDETECT) = YES
Default prefetch size (pages)         (DFT_PREFETCH_SZ) = AUTOMATIC

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) = 30
Number of secondary log files            (LOGSECOND) = 2
Changed path to log files               (NEWLOGPATH) =
Path to log files                          = /dev/sdd1
Overflow log path                    (OVERFLOWLOGPATH) =
Mirror log path                      (MIRRORLOGPATH) =
First active log file                      = S0000068.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) = 1600
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) = SYSTEM (RESTART)
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


valiant: db2 get db cfg for tpcd completed ok
```

DB2 Registry Variables

```
*** DB2 Variables

DB2LINUXAIO=TRUE
DB2_LGPAGE_BP=yes
DB2_EXTENDED_OPTIMIZATION=Y
DB2_ANTIJOIN=Y
DB2_LIKE_VARCHAR=Y,Y
DB2BPVARS=/home/db2inst1/tpcd/custom/bpvars.txt
DB2_FORCE_FCM_BP=ON
```

```
DB2OPTIONS=-t -v +c
DB2COMM=tcpip
DB2_PARALLEL_IO=*
DB2AUTOSTART=FALSE
```

DB2 Version

Database Version:

```
DB21085I  Instance "db2inst1" uses "64" bits and DB2 code release "SQL08025"
with level identifier "03060106".
Informational tokens are "DB2 v8.1.3.112", "s060429", "MI00159", and FixPak
"12".
Product is installed at "/opt/IBM/db2/V8.1".
```

SUSE Linux Version

Kernel Version:

```
Linux valiant 2.6.5-7.244-smp #1 SMP Mon Dec 12 18:32:25 UTC 2005 x86_64
x86_64 x86_64 GNU/Linux
```

SUSE Linux Configuration Parameters

/etc/sysctl.conf

Contents of /etc/sysctl.conf:

```
# Disable response to broadcasts.
# You don't want yourself becoming a Smurf amplifier.
net.ipv4.icmp_echo_ignore_broadcasts = 1
# enable route verification on all interfaces
net.ipv4.conf.all.rp_filter = 1
# enable ipV6 forwarding
#net.ipv6.conf.all.forwarding = 1
kernel.shmall=7549696
kernel.shmmax=27917287424
kernel.msgmni=16384
vm.nr_hugepages=11264
```

Contents of /etc/init.d/boot.local:

```
#! /bin/sh
#
# Copyright (c) 2002 SuSE Linux AG Nuernberg, Germany.  All rights reserved.
#
# Author: Werner Fink <werner@suse.de>, 1996
#         Burchard Steinbild, 1996
#
# /etc/init.d/boot.local
#
# script with local commands to be executed from init on system startup
#
# Here you should add things, that should happen directly after booting
# before we're going to the first run level.
#
/sbin/sysctl -p
```

Backup Devices

```
Personalities : [raid0] [raid5]
md4 : active raid0 sdp11[2] sdl11[1] sdh11[0]
      373703616 blocks 32k chunks

md3 : active raid5 sds11[2] sdr11[1] sdq11[0]
      151010560 blocks level 5, 128k chunk, algorithm 2 [3/3] [UUU]

md2 : active raid5 sdo11[2] sdn11[1] sdm11[0]
      151010560 blocks level 5, 128k chunk, algorithm 2 [3/3] [UUU]

md1 : active raid5 sdk11[2] sdj11[1] sdi11[0]
      151010560 blocks level 5, 128k chunk, algorithm 2 [3/3] [UUU]

md0 : active raid5 sdg11[2] sdf11[1] sde11[0]
      151010560 blocks level 5, 128k chunk, algorithm 2 [3/3] [UUU]

unused devices: <none>
/dev/sda3 on / type ext3 (rw,acl,user_xattr)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
tmpfs on /dev/shm type tmpfs (rw)
devpts on /dev/pts type devpts (rw,mode=0620,gid=5)
/dev/sda1 on /boot type ext3 (rw,acl,user_xattr)
/dev/hdc on /media/cdrecorder type subfs
(ro,nosuid,nodev,fs=cdfss,procuid,iocharset=utf8)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/md4 on /flatfiles type ext3 (rw)
/dev/md2 on /backup0 type ext3 (rw)
/dev/md3 on /backup1 type ext3 (rw)
/dev/md1 on /backup2 type ext3 (rw)
/dev/md0 on /backup3 type ext3 (rw)
```

Appendix B: Database Build Scripts

bpvars

```
NUMPREFETCHQUEUES=2
PREFETCHQUEUESIZE=200
```

backupdb.pl

```
#!/usr/bin/perl
# backup the database The system has been configured such that each node
# is backed up on the subsequent node.  (Node 0 backed up to Node 1,
# Node 1 backed up to Node 2 etc.
# Make output unbuffered.
select(STDOUT);
$| = 1 ;
$dbname="tpcd";

system("rm /backup0/TPCD.*");
system("rm /backup1/TPCD.*");
system("rm /backup2/TPCD.*");
system("rm /backup3/TPCD.*");
for ($node=0; $node <= 3; $node=$node+1)
{
  system("db2_all \"<<+$node <db2 backup database tpcd to /backup$node WITH
16 BUFFERS PARALLELISM 4 without prompting\" ");
}

$ret;
```

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 qualification
         database (sf = .1 = 100MB)\n";

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

# get TPC-D specific environment variables
#-------------------------------------------------------------------------#
# 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 "getvars";
require "macro.pl";
require "tpcdmacro.pl";
require "version";

# Make output unbuffered.
select(STDOUT);
$| = 1 ;
#-------------------------------------------------------------------------#
# verify that necessary environment variables for building the database     #
# are present.  Default those that aren't necessary                         #
#-------------------------------------------------------------------------#

# variables that must be specified for script to run
@reqVars     = ("TPCD_PLATFORM",
                "TPCD_PRODUCT",
                "TPCD_VERSION",
                "TPCD_DBNAME",
                "TPCD_MODE",
                "TPCD_SF",
                "TPCD_DDLPATH",
                "TPCD_AUDIT",
                "TPCD_AUDIT_DIR",
                "TPCD_BUILD_STAGE");

# variables default to 'NULL' if unspecified
@defNullVars = ("TPCD_LOAD_SCRIPT",
          "TPCD_LOAD_SCRIPT_QUAL",
          "TPCD_INPUT",
          "TPCD_QUAL_INPUT",
          "TPCD_DBGEN",
          "TPCD_LOGPRIMARY",
                "TPCD_LOGSECOND",
                "TPCD_LOGFILSIZ",
                "TPCD_LOG_DIR",
                "TPCD_MACHINE",
                "TPCD_AGENTPRI",
                "TPCD_STAGING_TABLE_DDL",
                "TPCD_PRELOAD_STAGING_TABLE_SCRIPT",
                "TPCD_DELETE_STAGING_TABLE_SQL",
                "TPCD_RUNSTATSHORT",
                "TPCD_ADD_RI",
                "TPCD_AST",
                "TPCD_DBM_CONFIG",
                "TPCD_EXPLAIN_DDL",
                "TPCD_NODEGROUP_DEF",
                "TPCD_BUFFERPOOL_DEF",
                "TPCD_LOAD_DB2SET_SCRIPT",
                "TPCD_DB2SET_SCRIPT",
                "TPCD_LOG_DIR_SETUP_SCRIPT",
                "TPCD_LOAD_CONFIGFILE",
                "TPCD_LOAD_DBM_CONFIGFILE",
                "TPCD_TEMP");


&setVar(@reqVars, "ERROR");
&setVar(@defNullVars, "NULL");


if ( $qual eq "QUAL" ){
   @reqQualVars =    ("TPCD_QUAL_DBNAME",
                      "TPCD_QUAL_DDL",
                      "TPCD_QUAL_TBSP_DDL",
                      "TPCD_QUALCONFIGFILE",
                      "TPCD_DBM_QUALCONFIG",
                      "TPCD_LOAD_QUALCONFIGFILE",
                      "TPCD_LOAD_DBM_QUALCONFIGFILE");

   &setVar(@reqQualVars, "ERROR");

   if ( ($ENV{"TPCD_QUAL_INPUT"}) eq "NULL" ){
      if ((($ENV{"TPCD_DBGEN"}) eq "NULL") ||
            (($ENV{"TPCD_TEMP"}) eq "NULL")){
         die "TPCD_DBGEN and TPCD_TEMP must be set if flatfiles are not
provided.\n";
      }
   }
}

$platform=$ENV{"TPCD_PLATFORM"};


if (length($ENV{"TPCD_DBPATH"}) <= 0){
   # 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 '$platform' not supported yet\n";
   }
}
if ( ($ENV{"TPCD_INPUT"}) eq "NULL" ){
   if ((($ENV{"TPCD_DBGEN"}) eq "NULL") ||
      (($ENV{"TPCD_TEMP"}) eq "NULL")){
      die "TPCD_DBGEN and TPCD_TEMP must be set if flatfiles are not
provided.\n";
   }
}
#-------------------------------------------------------------------------#
# ddl script files found under custom directory                           #
#-------------------------------------------------------------------------#

if (length($ENV{"TPCD_DDL"}) <= 0){
   $ENV{"TPCD_DDL"} = "dss.ddl";
}
if (length($ENV{"TPCD_TBSP_DDL"}) <= 0){
   $ENV{"TPCD_TBSP_DDL"} = "dss.tbsp.ddl";
}
if (length($ENV{"TPCD_INDEXDDL"}) <= 0){
   $ENV{"TPCD_INDEXDDL"} = "dss.index.H";
}
if (length($ENV{"TPCD_RUNSTATS"}) <= 0){
   $ENV{"TPCD_RUNSTATS"} = "dss.runstats";
}
if (length($ENV{"TPCD_CONFIGFILE"}) <= 0){
   $ENV{"TPCD_CONFIGFILE"} = "dbcfg_for_run";
}



#-------------------------------------------------------------------------#
#  other settings                                                         #
#-------------------------------------------------------------------------#

if (length($ENV{"TPCD_BACKUP_DIR"}) <= 0){
   $ENV{"TPCD_BACKUP_DIR"} = "${delim}dev${delim}null";
}
if (length($ENV{"TPCD_COPY_DIR"}) <= 0){
   $ENV{"TPCD_COPY_DIR"} = "${delim}dev${delim}null";
}
if (length($ENV{"TPCD_TEMP"}) <= 1){
   $ENV{"TPCD_TEMP"} = "/u/$instance/sqllib/tmp";
}
if (length($ENV{"TPCD_PHYS_NODE"}) <= 0){
   $ENV{"TPCD_NODEGROUP_DEF"}="NULL"
}
if (length($ENV{"TPCD_GENERATE_SEED_FILE"}) <= 0){
  $ENV{"TPCD_GENERATE_SEED_FILE"} = "no";
}
if (length($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_FASTPARSE"}) <= 0){
   $ENV{"TPCD_FASTPARSE"} = "no";
}
if (length($ENV{"TPCD_LOG"}) <= 0){
   $ENV{"TPCD_LOG"} = "no";
}
if (length($ENV{"TPCD_SMPDEGREE"}) <= 0 ){
   $ENV{"TPCD_SMPDEGREE"} = 1;
}
if (length($ENV{"TPCD_ACTIVATE"}) <= 0){
   $ENV{"TPCD_ACTIVATE"} = "no";
}
if (length($ENV{"TPCD_APPEND_ON"}) <= 0){
   $ENV{"TPCD_APPEND_ON"}="yes"
}
if (length($ENV{"TPCD_GENERATE_SEED_FILE"}) <= 0){
   $ENV{"TPCD_GENERATE_SEED_FILE"}="no";
}

#setup global variables
$tpcdVersion=           $ENV{"TPCD_VERSION"};
$buildStage=            $ENV{"TPCD_BUILD_STAGE"};
$mode=                  $ENV{"TPCD_MODE"};
$delim =        $ENV{"TPCD_PATH_DELIM"};
$sep =                  $ENV{"COMMAND_SEP"};
$ddlpath=       $ENV{"TPCD_DDLPATH"};
$extraindex=            $ENV{"TPCD_EXTRAINDEX"};
$earlyindex=            $ENV{"TPCD_EARLYINDEX"};
$loadstats=             $ENV{"TPCD_LOADSTATS"};
$addRI=                 $ENV{"TPCD_ADD_RI"};
$astFile=       $ENV{"TPCD_AST"};
$genSeed=       $ENV{"TPCD_GENERATE_SEED_FILE"};
$log=                   $ENV{"TPCD_LOG"};
$activate=      $ENV{"TPCD_ACTIVATE"};
$RealAudit=             $ENV{"TPCD_AUDIT"};
$auditDir=      $ENV{"TPCD_AUDIT_DIR"};
$loadsetScript=         $ENV{"TPCD_LOAD_DB2SET_SCRIPT"};
$user=                  $ENV{"USER"};
$logDirScript=          $ENV{"TPCD_LOG_DIR_SETUP_SCRIPT"};
$logprimary=            $ENV{"TPCD_LOGPRIMARY"};
$logsecond=             $ENV{"TPCD_LOGSECOND"};
$logfilsiz=     $ENV{"TPCD_LOGFILSIZ"};
$dbpath =       $ENV{"TPCD_DBPATH"};
$explainDDL=            $ENV{"TPCD_EXPLAIN_DDL"};
$platform=      $ENV{"TPCD_PLATFORM"};
$buffpooldef=           $ENV{"TPCD_BUFFERPOOL_DEF"};
$stagingTbl =           $ENV{"TPCD_STAGING_TABLE_DDL"};
$preloadSampleUF=
$ENV{"TPCD_PRELOAD_STAGING_TABLE_SCRIPT"};
$deleteSampleUF=  $ENV{"TPCD_DELETE_STAGING_TABLE_SQL"};
$machine=       $ENV{"TPCD_MACHINE"};
$runstatShort =         $ENV{"TPCD_RUNSTATSHORT"};
$runstats =     $ENV{"TPCD_RUNSTATS"};
$smpdegree =            $ENV{"TPCD_SMPDEGREE"};
$agentpri =             $ENV{"TPCD_AGENTPRI"};
$setScript =            $ENV{"TPCD_DB2SET_SCRIPT"};
$backupdir =            $ENV{"TPCD_BACKUP_DIR"};
$nodegroupdef=          $ENV{"TPCD_NODEGROUP_DEF"};
$dbgen=                 $ENV{"TPCD_DBGEN"};
$appendOn=              $ENV{"TPCD_APPEND_ON"};
$indexddl=              $ENV{"TPCD_INDEXDDL"};

if($qual eq "QUAL"){
        $logDir=  $ENV{"TPCD_LOG_QUAL_DIR"};
        $dbname= $ENV{"TPCD_QUAL_DBNAME"};
        $input=         $ENV{"TPCD_QUAL_INPUT"};
        $sf=            $ENV{"TPCD_QUAL_SF"};
        $loadconfigfile=$ENV{"TPCD_LOAD_QUALCONFIGFILE"};
        $loadDBMconfig=
$ENV{"TPCD_LOAD_DBM_QUALCONFIGFILE"};
        $loadscript =   $ENV{"TPCD_LOAD_SCRIPT_QUAL"};
        $configfile =   $ENV{"TPCD_QUALCONFIGFILE"};
        $dbmconfig =    $ENV{"TPCD_DBM_QUALCONFIG"};
        $ddl=           $ENV{"TPCD_QUAL_DDL"};
        $tbspddl=  $ENV{"TPCD_QUAL_TBSP_DDL"};
}else{
        $logDir=  $ENV{"TPCD_LOG_DIR"};
        $dbname= $ENV{"TPCD_DBNAME"};
        $input=         $ENV{"TPCD_INPUT"};
        $sf=            $ENV{"TPCD_SF"};
        $loadconfigfile=$ENV{"TPCD_LOAD_CONFIGFILE"};
        $loadDBMconfig=
$ENV{"TPCD_LOAD_DBM_CONFIGFILE"};
        $loadscript =   $ENV{"TPCD_LOAD_SCRIPT"};
        $configfile =   $ENV{"TPCD_CONFIGFILE"};
        $dbmconfig =    $ENV{"TPCD_DBM_CONFIG"};
        $ddl=           $ENV{"TPCD_DDL"};
```

```
        $tbspddl=  $ENV{"TPCD_TBSP_DDL"};
}


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";
}

#-------------------------------------------------------------------------#
#  echo parameter settings to acknowledge what is being built             #
#  and set db2set options for database load                               #
#-------------------------------------------------------------------------#

&printSummary;

print "\nSleeping for 15 seconds to give you a chance to reconsider...\n";
sleep 15;

if ( $platform eq "nt" ){
   if (($mode eq "uni") || ($mode eq "smp")){
      #spaces required for NT
      $rc=&dodb_noconn("db2set DB2OPTIONS=\"  -t -v +c\";db2set
DB2NTNOCACHE=ON",$all_ln);
   }
   else{
      $rc=&dodb_noconn("db2set DB2OPTIONS=\\\"  -t -v +c\\\";db2set
DB2NTNOCACHE=ON",$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 : rc = $rc\n";
}

#-------------------------------------------------------------------------#
# set the db2 env vars for loading, from the TPCD_LOAD_DB2SET_SCRIPT
script #
#-------------------------------------------------------------------------#

if ( $loadsetScript ne "NULL" )
{
   if ( $platform eq "nt" ){
      if (( $mode eq "uni" ) || ( $mode eq "smp" )){
         $rc=system("${ddlpath}${delim}$loadsetScript");
      }
      else{
         $rc=system(" rah \" cd ${ddlpath} & $loadsetScript\" ");
      }
   }
   else{
      $rc=system("${ddlpath}${delim}$loadsetScript");
   }
   ($rc == 0) || die "failure loading db2set parms from $loadsetScript \n";
}

!&stopStart || die;
#-------------------------------------------------------------------------#
# Begin complete build: TPCD_BUILDSTAGE = ALL                            #
#-------------------------------------------------------------------------#
if($buildStage eq "ALL") {
        #create the database
        $rc = &createDb;
        ($rc == 0) || die "ERROR:[$0] create database failed. rc = $rc\n ";
        &setLog;
};

$rc = &setLoadConfig;

#-------------------------------------------------------------------------#
# Begin build from CreateTablespace or early Indexes                     #
#-------------------------------------------------------------------------#

if( $buildStage eq "ALL" ||
   $buildStage eq "CRTTBSP" ||
   ($buildStage eq "INDEX" && $earlyindex eq "yes")){
        !&createNodegroups ||  print "ERROR:[$0] create nodegroups
failed.\n";
        !&createBufferPools || print "ERROR:[$0] create bufferpools
failed.\n";
        &outtime("*** Start of audited Load Time - starting to create tables");
        !&createTablespaces || print "WARNING:[$0] create tablespaces
error.\n";
        !&createExplainTbls ||  print "ERROR:[$0] create EXPLAIN tables
failed.\n";
        !&createTables || print "ERROR:[$0] create tables failed.\n";

        mkdir("${delim}tmp${delim}$instance",0777);

        # if earlyindex requested, create indexes
        if ( $earlyindex eq "yes" ){
                !&createIndexes("early") || die "ERROR:[$0] create early
indexes failed.\n";
        }
        # start the dbgen and load.....call the specific mode for loading
(uni,smp,mln)
        !&loadData || die "ERROR:[$0] failure during load data\n";

        # remove the update.pair.num file so when setupDir runs, it doesn't
        # hang waiting for an answer on nt
        &rm("$auditDir${delim}$dbname.$user.update.pair.num");
        # 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");

};
#-------------------------------------------------------------------------#
#  Begin build from Index or Load                                        #
#-------------------------------------------------------------------------#
if( $buildStage eq "ALL" ||
   $buildStage eq "CRTTBSP" ||
   $buildStage eq "LOAD" ||
   $buildStage eq "INDEX"){

        # if indexes haven't been created, do so now
        if ( $earlyindex ne "yes" ){
                !&createIndexes("normal") || die "ERROR:[$0] create
indexes failed.\n";
        }
        if ( $extraindex ne "no" ){
                !&createIndexes("extra") || die "ERROR:[$0] create extra
indexes failed.\n";
        }

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

```
#-------------------------------------------------------------------------#
#  Begin build from runstats                                              #
#-------------------------------------------------------------------------#

if( $buildStage eq "ALL" ||
   $buildStage eq "CRTTBSP" ||
   $buildStage eq "LOAD" ||
   $buildStage eq "INDEX" ||
   $buildStage eq "RUNSTATS"){
            # 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" )){
                        &doRunStats;
            }
};

#-------------------------------------------------------------------------#
# End build phase: all/load/index/runstats                                #
#-------------------------------------------------------------------------#
# Add RI/AST, set run configuration                                       #
#-------------------------------------------------------------------------#

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");

&setConfiguration;
# if logging is enabled, we must take a backup of the database
if ( $log eq "yes" ){
            &createBackup;
}

# stop and restart the database to get config parms recognized
!&stopStart || die;

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

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


#-------------------------------------------------------------------------#
#  Call buildptpcdbatch to compile tpcdbatch                              #
#-------------------------------------------------------------------------#
#  - 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 be prompted #
#    doing setupDir.                                                      #
#  - before we stop the database for the final time                       #
#    if we are in the real audit mode then run dbtables and dbcheck before #
#    we print out the notice that we're ready to run performance tests    #
#    if we are building the qualification database then we'll bind to both #
#    the dbname database and the qualification database                   #
#-------------------------------------------------------------------------#

$rc = system("perl buildtpcdbatch $qual");
($rc == 0 ) || die "buildtpcdbatch failed rc=$rc\n";

if ( $RealAudit eq "yes" ){
            &rm("$auditDir${delim}tools${delim}tpcd.runsetup");
            system("perl setupRun");
            if ( $qual eq "QUAL" ){
                        $verifyType="q";
            }
            else{
                        $verifyType="t";
            }
            system("perl tablesdb $verifyType");

&dodb2file($dbname,"$auditDir${delim}tools${delim}first10rows.sql",$once);
}


#-------------------------------------------------------------------------#
#  Create Catalog info                                                    #
#-------------------------------------------------------------------------#
$rc = system("perl catinfo.pl b");
($rc == 0 ) || warn "catinfo failed!!!  rc = $rc\n";

$rc=system("db2stop");
($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
   $rc=system("db2start");
   ($rc == 0 ) || die "failure during db2start rc = $rc \n";
   if ( $activate eq "yes" ){
      &dodb_noconn("activate database $dbname",$once);
   }
}

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



#-------------------------------------------------------------------------#
# Function: setLog                                                        #
#-------------------------------------------------------------------------#
sub setLog{
            # update the log information first
            # set up the log directory before we do any index creation
            my $rc;
            my $setLogs;
            my $setLogString;

            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";
	$rc = &dodb_noconn("$setLogString",$all_ln);

}

#-------------------------------------------------------------------------#
# Function: createDb                                                      #
#-------------------------------------------------------------------------#
sub createDb{
	&outtime("*** Starting to create the database");
	# setup required variables
	my $rc;
	$rc = &dodb_noconn("db2 \"create database $dbname on $dbpath
collate using identity with 'TPC-D $sf GB'\"",$once);
	($rc == 0) || return($rc);
	# 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 $sep
 \
db2 commit",$once);
 &dodb_conn($dbname,"db2 grant connect on database to public $sep      \
     db2 commit",$once);
&dodb_noconn("db2 reset database manager configuration",$once);
}

#-------------------------------------------------------------------------#
# Function: createNodegroups                                              #
#-------------------------------------------------------------------------#
sub createNodegroups{
	&outtime("*** Creating the nodegroups.");
	my $rc;
	if ( $nodegroupdef ne "NULL"){
		$rc =
&dodb2file($dbname,"$ddlpath${delim}$nodegroupdef",$once);
	}
}

#-------------------------------------------------------------------------#
# Function: createExplainTbls                                             #
#-------------------------------------------------------------------------#
sub createExplainTbls{
	&outtime("*** Creating the EXPLAIN tables.");
	my $rc;
	my $explnPathFile;
	my $home;
	my $sqlpath;

	if ( $explainDDL ne "NULL" ){
		$explnPathFile="$explainDDL";
	}
	else{
		if ( $platform eq "ptx" ){
			$home=$ENV{"HOME"};
			$sqlpath="$home${delim}sqllib";
		}
		if ( $platform ne "nt" ){
			$home=$ENV{"HOME"};
			$sqlpath="$home${delim}sqllib";
		}
		else{
			$sqlpath=$ENV{"DB2PATH"};
		}

$explnPathFile="$sqlpath${delim}misc${delim}EXPLAIN.DDL";
	}
	$rc = &dodb_conn($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);
}
#-------------------------------------------------------------------------#
# Function: createBufferPools                                             #
#-------------------------------------------------------------------------#
sub createBufferPools{
	my $rc;
	&outtime("*** Creating the bufferpools");
	if ( $buffpooldef ne "NULL" ){
		#run the create bufferpool ddl
		$rc =
&dodb2file($dbname,"$ddlpath${delim}$buffpooldef",$once);
	}
}

#-------------------------------------------------------------------------#
# Function: createTablespaces                                             #
#-------------------------------------------------------------------------#
sub createTablespaces{
	&outtime("*** Ready to start creating the tablespaces");
	# setup required variables
	my $rc;
	$rc = &dodb2file($dbname,"$ddlpath${delim}$tbspddl",$once);
	($rc == 0) || return $rc;
	# 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
database
		$rc =
&dodb2file($dbname,"$ddlpath${delim}$stagingTbl",$once);
		($rc == 0) || return $rc;
		if ( $qual ne "QUAL" ){
			if ( $preloadSampleUF ne "NULL" ){
				# preload the sample UF data for
statistics gathering
				$rc = system ("perl
$ddlpath${delim}$preloadSampleUF");
				#($rc == 0) || return $rc;
			}
			if ( $deleteSampleUF ne "NULL" ){
				# delete the sample rows now that
stats have been gathered
				$rc =
&dodb2file($dbname,"$ddlpath${delim}$deleteSampleUF",$once);
				#($rc == 0) || return $rc;
			}
```

```
                        }
                }
}

#-------------------------------------------------------------------------#
# Function: createTables                                                  #
#-------------------------------------------------------------------------#
sub createTables{
        my $rc;
        $rc = &dodb2file($dbname,"$ddlpath${delim}$ddl",$once);
        ($rc == 0) || return $rc;
        # update the locksize on the non-updated tables to be table level
locking
        # update the tables for appendmode
        if ($appendOn eq "yes"){
                $rc = &dodb_conn($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);
        }
        else{
                $rc = &dodb_conn($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);
        }
}

#-------------------------------------------------------------------------#
# Function: createIndexes                                                 #
#-------------------------------------------------------------------------#
sub createIndexes{
        # setup required variables
        local @args = @_;
        my $indexType = @args[0];
        my $rc;
        &outtime("*** Starting to create $indexType indexes");
        if( $indexType eq "extra"){
                $rc =
&dodb2file($dbname,"$ddlpath${delim}$extraindex",$once);
        }elsif ($indexType eq "early" || $indexType eq "normal"){
                $rc =
&dodb2file($dbname,"$ddlpath${delim}$indexddl",$once);
        }
        &outtime("*** Create $indexType index completed");
        return $rc;
}



#-------------------------------------------------------------------------#
# Function: setLoadConfig                                                 #
#-------------------------------------------------------------------------#
sub setLoadConfig{

        &outtime("*** Setting LOAD configuration.");
        my $rc;
        my $buffpage;
        my $sortheap;
        my $sheapthres;
        my $util_heap_sz;
        my $ioservers;
        my $ioclnrs=                    1;
        my $chngpgs=                    60;

    if($loadDBMconfig ne "NULL"){
        $rc =
&dodb2file_noconn("$ddlpath${delim}$loadDBMconfig",$once);
    }
    else{
        $rc = &dodb_noconn("db2 update dbm cfg using sheapthres
$sheapthres",$once);
    }

        if (($loadconfigfile eq "") || ($loadconfigfile eq "NULL")){
                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;
                }
                else {
                        die "Neither a LOAD config filename nor a
valid machine size has \
                        been specified!\n";
                }
                $rc = &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);
        }
        else{
                $rc =
&dodb2file_noconn("$ddlpath${delim}$loadconfigfile",$all_ln);
        }
#       ($rc == 0) || return $rc;
#       if($loadDBMconfig ne "") || ($loadDBMconfig ne "NULL"){
#               $rc =
&dodb2file_noconn("$ddlpath${delim}$loadDBMconfig",$once);
#       }
#       else{
#               $rc = &dodb_noconn("db2 update dbm cfg using
sheapthres $sheapthres",$once);
#       }
        ($rc == 0) || return $rc;
        &dodb_noconn("db2 terminate",$once);
        $rc = &stopStart;
        return $rc;
}
#-------------------------------------------------------------------------#
# Function: loadData                                                      #
#-------------------------------------------------------------------------#
sub loadData{
        # start the dbgen and load.....call the specific mode for loading
(uni,smp,mln)
        my $rc;
        if (( $mode eq "uni" ) || ( $mode eq "smp" )){
```

```
                        &outtime("*** Starting the load");
                        # call the appropriate dbgen/load for uni/smp
                        if ( $loadscript eq "NULL"){
                                $rc = system("perl genloaduni $qual");
                                ($rc == 0) || print "ERROR:[$0] genloaduni
failed rc = $rc\n";
                        }
                        else{
                                $rc =
&dodb2file_noconn("$ddlpath${delim}$loadscript",$once);
                                ($rc == 0) || print "ERROR:[$0] load script:
$loadscript failed. rc = $rc\n";
                        }
                }
                elsif ( $mode eq "mln" ) {
                        &outtime("*** Starting the load");
                        # call the appropriate dbgen/split/(sort)/load for mln
                        if ( $loadscript eq "NULL"){
                                $rc = system("perl genloadmln $qual");
                                ($rc == 0) || print "ERROR:[$0] genloadmln
failed. rc = $rc\n";
                        }
                        else{
                         #      system("$ddlpath${delim}$loadscript");
                        #       $rc =
&dodb2file_noconn("$ddlpath${delim}$loadscript $sf");
                                $rc =
&dodb2file_noconn("$ddlpath${delim}$loadscript",$once);
                                ($rc == 0) || print "ERROR:[$0] load script
$loadscript failed. rc = $rc\n";
                        }
                }

        elsif ( $mode eq "mpp" ){
                &outtime("*** Starting the load");
                # call the appropriate dbgen/split/(sort)/load for mpp
                if ( $loadscript eq "NULL"){
                        $rc = system("perl genloadmpp $qual");
                        ($rc == 0) || print "ERROR:[$0] genloadmpp failed. rc = $rc\n";
                }
                else{
                        #system("$ddlpath${delim}$loadscript");
                        #$rc = &dodb2file_noconn("$ddlpath${delim}$loadscript $sf");
                        $rc = &dodb2file_noconn("$ddlpath${delim}$loadscript");
                        ($rc == 0) || print "ERROR:[$0] load script $loadscript failed. rc =
$rc\n";
                }
        }
                else{
                        print "TPCD_MODE not set to one of uni, smp, mln or
mpp\n";
                        $rc = 1;
                }
                ($rc == 0) || &outtime("*** Load complete");
                return $rc;
}

#-------------------------------------------------------------------------#
# Function: doRunStats                                                    #
#-------------------------------------------------------------------------#
sub doRunStats{
        # 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");
}

#-------------------------------------------------------------------------#
# Function: setConfiguration                                              #
#-------------------------------------------------------------------------#
sub setConfiguration{
                my $ret = 0;
&dodb_noconn("db2 update database manager configuration using
max_querydegree $smpdegree",$once);
                &dodb_noconn("db2 update database configuration for $dbname using
dft_degree $smpdegree",$all_ln);
#mujib          &dodb_noconn("db2 update database manager configuration using
max_querydegree $smpdegree",$once);
&dodb2file_noconn("${ddlpath}${delim}$dbmconfig",$once);
                &dodb2file_noconn("${ddlpath}${delim}$configfile",$all_ln);
#mujib          &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} & $setScript\" ");
                      }
                   }
                   #($ret == 0 ) || die "failure setting runtime db2set parms from
$setScript \n";
                }
        &outtime("*** Setting up 4K-Temp");
       # Added by Tricia
                $ret=&dodb2file($dbname,"$ddlpath${delim}temp_4k",$once);
        &outtime("*** Config Completed.");
}

#-------------------------------------------------------------------------#
# Function: createBackup                                                  #
#-------------------------------------------------------------------------#
sub createBackup{
                my $rc;
                &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.
            # PST - Commented out the next line cause it results in an
        # unexpected system error on Linux. Can't have one and continue.
        #&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 $backupdir without prompting\" \' ");
                           #($rc == 0 ) || print "ERROR: backup of catalog node
failed rc = $rc\n";
                           # back up remaining nodes
                           #$rc=system("db2_all \'||}]<<-000< db2 backup database
$dbname to $backupdir without prompting\' ");
                           #($rc == 0 )|| print "ERROR: backup of remaining nodes
failed rc = $rc\n";
                        }
                        else{
                          #$rc = &dodb_noconn("db2 backup database $dbname to
$backupdir",$once);
                        }
            system ("perl $ddlpath${delim}backupdb.pl");
            $rc=0;
                        ($rc == 0) || &outtime("*** Finished the backup");
              }
              else{
            #PST - Changed invocation of backup
            &outtime("*** Starting the backup");
                        # 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
database; however
                        # all other mechanisms that guarantee durability of the
qualification
                        # database must be enabled in the same way for the test
database".
                        # According to this clause we do need to keep the backup
of the database.
                        #$rc = &dodb_noconn("db2dart $dbname /CHST /WHAT
DBBP OFF",$all_ln);
            system ("perl $ddlpath${delim}backupdb.pl");
            $rc=0;
            ($rc == 0) || &outtime("*** Finished the backup");
              }
              return $rc;
}

#-------------------------------------------------------------------------#
# Function: printSummary                                                  #
#-------------------------------------------------------------------------#
sub printSummary{
              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}dbcfg_for_run${sfReal}GB\n";
                 $configfile="dbcfg_for_run${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 "") || ($loadconfigfile eq "NULL")){
            print "   Machine size set to $machine so the following
configuration\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";
          }
}

1;
```

create_bufferpools

```
alter  bufferpool ibmdefaultbp size 250;
create bufferpool BP32K size 10000 pagesize 32k;
```

create_indexes

```
values(current timestamp);
CREATE UNIQUE INDEX "TPCD  "."R_RK" ON "TPCD    "."REGION"
          ("R_REGIONKEY" ASC)
          PCTFREE 0 ;
commit work;

values(current timestamp);
CREATE UNIQUE INDEX "TPCD  "."N_NK" ON "TPCD    "."NATION"
          ("N_NATIONKEY" ASC)
          PCTFREE 0 ;
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;

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

values(current timestamp);
CREATE UNIQUE INDEX "TPCD    "."P_PK" ON "TPCD    "."PART"
          ("P_PARTKEY" ASC)
          PCTFREE 0 ;
commit work;
values(current timestamp);

values(current timestamp);
CREATE       INDEX "TPCD    "."PS_SK" ON "TPCD    "."PARTSUPP"
          ("PS_SUPPKEY" ASC)
          PCTFREE 0 ;
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_PKSK" ON "TPCD    "."PARTSUPP"
          ("PS_PARTKEY" ASC,
           "PS_SUPPKEY" 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 UNIQUE INDEX "TPCD  "."C_CK" ON "TPCD    "."CUSTOMER"
          ("C_CUSTKEY" ASC)
          PCTFREE 0 ;
```

```
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;

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_ok_ln on tpcd.lineitem (L_orderkey, l_linenumber)
pctfree 3;
commit work;
values(current timestamp);

-- values(current timestamp);
-- create unique index tpcd.l_ok on tpcd.lineitem (L_orderkey) pctfree 3;
-- commit work;
-- values(current timestamp);

alter table tpcd.orders add primary key(o_orderkey);
alter table tpcd.lineitem add primary key(l_orderkey,l_linenumber);
commit work;
```

create_nodegroups

```
-- Create nodegroups.
create nodegroup catalog_node on node(0);
create nodegroup all_nodes;
```

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 WITH
DEFAULT)
     IN SMALL_TABLES;


CREATE TABLE TPCD.REGION  ( R_REGIONKEY  INTEGER NOT NULL,
                  R_NAME      CHAR(25) NOT NULL,
                  R_COMMENT    VARCHAR(152) NOT NULL WITH
DEFAULT)
     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 WITH
DEFAULT )
     IN OTHER_STUFF
     INDEX IN OTHER_INDEX
     PARTITIONING KEY(P_PARTKEY) USING HASHING;

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 WITH
DEFAULT)
     IN OTHER_STUFF
     INDEX IN OTHER_INDEX
     PARTITIONING KEY(S_SUPPKEY);

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 WITH
DEFAULT)
     IN OTHER_STUFF
     INDEX IN OTHER_INDEX
     PARTITIONING KEY(PS_PARTKEY) USING HASHING;

CREATE TABLE TPCD.CUSTOMER ( C_CUSTKEY     INTEGER NOT
NULL,
                  C_NAME       CHAR(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 WITH
DEFAULT)
     IN OTHER_STUFF
     INDEX IN OTHER_INDEX
     PARTITIONING KEY(C_CUSTKEY);

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 WITH
DEFAULT
--                PRIMARY KEY (O_ORDERKEY)
                )
     IN OTHER_STUFF
     INDEX IN OTHER_INDEX
     PARTITIONING KEY(O_ORDERKEY) USING HASHING
     ORGANIZE BY ( ( O_ORDERDATE ) );

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 WITH
DEFAULT
--                    PRIMARY KEY (L_ORDERKEY, L_LINENUMBER)
                    )
      IN LINEITEM_TABLE
      INDEX IN LINEITEM_INDEXES
      PARTITIONING KEY(L_ORDERKEY) USING HASHING
      ORGANIZE BY (  (L_SHIPDATE) );

COMMIT WORK;
```

create_tablespaces

```
drop tablespace small_tables;
drop tablespace lineitem_table;
drop tablespace lineitem_indexes;
drop tablespace other_stuff;
drop tablespace other_index;
drop tablespace temp_tables;
drop tablespace temp2_tables;
commit;

create regular tablespace small_tables
  in nodegroup catalog_node
  managed by system
  using ('/data/small_tables')
  on node (0)
  overhead      5.0
  transferrate  0.22;

create regular tablespace LINEITEM_TABLE
  pagesize 32K
  managed by database
  using ( device '/dev/sde7'      17728M,
          device '/dev/sdf7'      17728M,
          device '/dev/sdg7'      17728M,
          device '/dev/sdh7'      17728M) on node(0)
  using ( device '/dev/sdi7'      17728M,
          device '/dev/sdj7'      17728M,
          device '/dev/sdk7'      17728M,
          device '/dev/sdl7'      17728M) on node(1)
  using ( device '/dev/sdm7'      17728M,
          device '/dev/sdn7'      17728M,
          device '/dev/sdo7'      17728M,
          device '/dev/sdp7'      17728M) on node(2)
  using ( device '/dev/sdq7'      17728M,
          device '/dev/sdr7'      17728M,
          device '/dev/sds7'      17728M,
          device '/dev/sdt7'      17728M) on node(3)
  bufferpool   BP32K
  extentsize     24
  prefetchsize   576
  overhead      25.0
  transferrate  0.10;

create regular tablespace LINEITEM_INDEXES
  pagesize 32K
  managed by database
  using ( device '/dev/sde6'     3136M,
          device '/dev/sdf6'      3136M,
          device '/dev/sdg6'      3136M,
          device '/dev/sdh6'     3136M) on node(0)
  using ( device '/dev/sdi6'     3136M,
          device '/dev/sdj6'      3136M,
          device '/dev/sdk6'      3136M,
          device '/dev/sdl6'     3136M) on node(1)
  using ( device '/dev/sdm6'     3136M,
          device '/dev/sdn6'      3136M,
          device '/dev/sdo6'      3136M,
          device '/dev/sdp6'     3136M) on node(2)
  using ( device '/dev/sdq6'     3136M,
          device '/dev/sdr6'      3136M,
          device '/dev/sds6'      3136M,
          device '/dev/sdt6'     3136M) on node(3)
  bufferpool   BP32K
  extentsize     24
  prefetchsize   576
  overhead      25.0
  transferrate  0.10;


create regular tablespace OTHER_STUFF
  pagesize 32K
  managed by database
  using ( device '/dev/sde8'      8064M,
          device '/dev/sdf8'      8064M,
          device '/dev/sdg8'      8064M,
          device '/dev/sdh8'      8064M) on node(0)
  using ( device '/dev/sdi8'      8064M,
          device '/dev/sdj8'      8064M,
          device '/dev/sdk8'      8064M,
          device '/dev/sdl8'      8064M) on node(1)
  using ( device '/dev/sdm8'      8064M,
          device '/dev/sdn8'      8064M,
          device '/dev/sdo8'     8064M,
          device '/dev/sdp8'      8064M) on node(2)
  using ( device '/dev/sdq8'      8064M,
          device '/dev/sdr8'      8064M,
          device '/dev/sds8'      8064M,
          device '/dev/sdt8'      8064M) on node(3)
  bufferpool   BP32K
  extentsize    24
  prefetchsize  576
  overhead      25.0
  transferrate  0.10;

create regular tablespace OTHER_INDEX
  pagesize 32K
  managed by database
  using ( device '/dev/sde9'      1792M,
          device '/dev/sdf9'      1792M,
          device '/dev/sdg9'      1792M,
          device '/dev/sdh9'      1792M) on node(0)
  using ( device '/dev/sdi9'      1792M,
          device '/dev/sdj9'      1792M,
          device '/dev/sdk9'      1792M,
          device '/dev/sdl9'      1792M) on node(1)
  using ( device '/dev/sdm9'      1792M,
          device '/dev/sdn9'      1792M,
          device '/dev/sdo9'      1792M,
          device '/dev/sdp9'      1792M) on node(2)
  using ( device '/dev/sdq9'      1792M,
          device '/dev/sdr9'      1792M,
          device '/dev/sds9'      1792M,
          device '/dev/sdt9'      1792M) on node(3)
  bufferpool   BP32K
  extentsize     24
  prefetchsize   576
  overhead      25.0
  transferrate  0.10;

create temporary tablespace TEMP_TABLES
  pagesize 32K
  managed by database
  using ( device '/dev/sde10'      34752M,
          device '/dev/sdf10'      34752M,
          device '/dev/sdg10'      34752M,
          device '/dev/sdh10'      34752M) on node(0)
  using ( device '/dev/sdi10'      34752M,
          device '/dev/sdj10'      34752M,
          device '/dev/sdk10'      34752M,
          device '/dev/sdl10'      34752M) on node(1)
  using ( device '/dev/sdm10'      34752M,
          device '/dev/sdn10'      34752M,
          device '/dev/sdo10'      34752M,
          device '/dev/sdp10'      34752M) on node(2)
  using ( device '/dev/sdq10'      34752M,
```

```
        device '/dev/sdr10'     34752M,
        device '/dev/sds10'     34752M,
        device '/dev/sdt10'     34752M) on node(3)
  bufferpool   BP32K
  extentsize    24
  prefetchsize  576
  overhead     25.0
  transferrate  0.10;

create temporary tablespace TEMP2_TABLES
  pagesize 4K
  managed by database
  using ( device '/dev/sde5'      12160M,
        device '/dev/sdf5'      12160M,
        device '/dev/sdg5'      12160M,
        device '/dev/sdh5'      12160M) on node(0)
  using ( device '/dev/sdi5'      12160M,
        device '/dev/sdj5'      12160M,
        device '/dev/sdk5'      12160M,
        device '/dev/sdl5'      12160M) on node(1)
  using ( device '/dev/sdm5'      12160M,
        device '/dev/sdn5'      12160M,
        device '/dev/sdo5'      12160M,
        device '/dev/sdp5'      12160M) on node(2)
  using ( device '/dev/sdq5'      12160M,
        device '/dev/sdr5'      12160M,
        device '/dev/sds5'      12160M,
        device '/dev/sdt5'      12160M) on node(3)
  bufferpool   IBMDEFAULTBP
  extentsize    192
  prefetchsize  768
  overhead     5.0
  transferrate  0.40;

commit work;
```

createmseedme.pl

```
#!/usr/bin/perl

push(@INC, split(':', $ENV{'PATH'}));

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

$seedTime;  #holds timestamp which all seeds are created from
$numSeeds;  #number of seeds to create
$seedFile;  #filename of seedfile

#create base seed
$seedTime = (localtime)[4]; #gets month
$seedTime++;  #Months start at 0, not 1, so increment month so that april is 4 and
not 3
# ensures a standard length of 9 or 10 (depending on the month) for
mm/dd/hh/mm/ss
# ie 404040404 instead of 44444 for april 4 04:04:04.  A '0' is not necessary for a
# month < 10 though.
# (localtime)[3] gets day, [2] gets hour, [1] gets minute, and [0] gets second.
for ($i = 3; $i > -1 ; $i--){
        $t = (localtime)[$i];
        if ($t < 10){
                $t = "0".$t; #inserts a '0' infront of single digit number
        }
        $seedTime = $seedTime.$t
}

print "****Createmseedme base timestamp is: $seedTime\n";

#set # of seeds and seed filename
if (@ARGV eq 1){
        $numSeeds = int($ARGV[0]);
        if ($numSeeds eq 0){
                $numSeeds = 1000;
        }
}
else{
        $numSeeds = 1000; #default value
}

if (length($ENV{"TPCD_AUDIT_DIR"}) <= 0)
{
  die "TPCD_AUDIT_DIR environment variable not set\n";
}

$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$seedFile = "$auditDir${delim}auditruns${delim}mseedme";

#create seed file and populate it, with each new seed incremented by 1.
open(SEEDFILE, ">$seedFile") || warn ("Can not open the file $seedFile!\n");
for ($i = 0; $i < $numSeeds; $i++)
{
        print SEEDFILE $seedTime++."\n";
}
close SEEDFILE || warn ("Can not close the file $seedFile!\n");
1;
```

createUFtables

```
connect to tpcd;

drop table TPCDTEMP.ORDERS_NEW;
drop table TPCDTEMP.ORDERS_DEL;
drop table TPCDTEMP.LINEITEM_NEW;

commit;

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 OTHER_STUFF
      INDEX IN OTHER_INDEX
      PARTITIONING KEY(O_ORDERKEY) USING HASHING;

CREATE INDEX "TPCDTEMP"."I_ORDERS_NEW" ON
"TPCDTEMP"."ORDERS_NEW"
           ("APP_ID" ASC,
            "O_ORDERKEY" ASC,
            "O_CUSTKEY" ASC,
            "O_ORDERSTATUS" ASC,
            "O_TOTALPRICE" ASC,
            "O_ORDERDATE" ASC,
            "O_ORDERPRIORITY" ASC,
            "O_CLERK" ASC,
            "O_SHIPPRIORITY" ASC,
            "O_COMMENT" ASC) PCTFREE 0;


CREATE TABLE TPCDTEMP.ORDERS_DEL ( APP_ID     INTEGER NOT
NULL,
                        O_ORDERKEY  INTEGER NOT NULL)
      IN OTHER_STUFF
      INDEX IN OTHER_INDEX
      PARTITIONING KEY(O_ORDERKEY) USING HASHING;

CREATE UNIQUE INDEX "TPCDTEMP"."I_ORDERS_DEL" ON
"TPCDTEMP"."ORDERS_DEL"
           ("APP_ID" ASC,
```

```
        "O_ORDERKEY" ASC) PCTFREE 0;



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 LINEITEM_TABLE
      INDEX IN LINEITEM_INDEXES
      PARTITIONING KEY(L_ORDERKEY);

CREATE INDEX "TPCDTEMP"."I_LINEITEM_NEW" ON
"TPCDTEMP"."LINEITEM_NEW"
           ("APP_ID" ASC) PCTFREE 0;

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;

connect reset;
```

db2nodes.cfg

```
0 valiant 0
1 valiant 1
2 valiant 2
3 valiant 3
```

deletedummyufdata.sql

```
connect to tpcd;
delete from TPCDTEMP.ORDERS_NEW;
delete from TPCDTEMP.LINEITEM_NEW;
delete from TPCDTEMP.ORDERS_DEL;
commit work;
connect reset;
```

drop_tables

```


DROP TABLE TPCD.NATION;

DROP TABLE TPCD.REGION;
DROP TABLE TPCD.PART;
DROP TABLE TPCD.SUPPLIER;

DROP TABLE TPCD.PARTSUPP;

DROP TABLE TPCD.CUSTOMER;
DROP TABLE TPCD.ORDERS;
DROP TABLE TPCD.LINEITEM;
```

dss.runstats

```
values (current timestamp, 'TS*** runstats nation START like ');
RUNSTATS ON TABLE TPCD.NATION    WITH DISTRIBUTION on all
columns
  and columns (
    n_name like statistics,
    n_comment like statistics )
  AND detailed INDEXES ALL;
commit;
values (current timestamp, 'TS*** runstats done nation ');
RUNSTATS ON TABLE TPCD.REGION    WITH DISTRIBUTION on all
columns
  and columns (
    r_name like statistics,
    r_comment like statistics )
  AND detailed 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 detailed INDEXES ALL;
commit;
values (current timestamp, 'TS*** runstats done part ');
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 detailed INDEXES ALL;
commit;
values (current timestamp, 'TS*** runstats done partsupp ');
RUNSTATS ON TABLE TPCD.PARTSUPP  WITH DISTRIBUTION on all
columns
  and columns (
    ps_comment like statistics)
  AND detailed INDEXES ALL;
commit;
values (current timestamp, 'TS*** runstats done customer ');

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 detailed  INDEXES ALL;
commit;
values (current timestamp, 'TS*** runstats done orders ');
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 detailed INDEXES ALL;
commit;
values (current timestamp, 'TS*** runstats done lineitem ');
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 detailed INDEXES ALL;
COMMIT WORK;
values (current timestamp, 'TS*** runstats END like');
```

load.dbcfg.sql

```
update database configuration for tpcd using
num_freqvalues 0
num_quantiles 600
buffpage    10000
catalogcache_sz 386
chngpgs_thresh 15
dbheap 15000
locklist 16384
logbufsz 2048
logfilsiz 16384
logsecond 10
logprimary 10
maxappls 40
maxlocks 60
mincommit 1
num_iocleaners 4
num_ioservers  4
pckcachesz 320
softmax 750
sortheap 25000
stat_heap_sz 16000
stmtheap 4096
util_heap_sz 128000
applheapsz 768
app_ctl_heap_sz 1024
dft_queryopt 7
dft_degree  1;
get database configuration for tpcd;
```

load_dbmcfg.sql

```
update database manager configuration using
cpuspeed 7.951129e-7
sheapthres 430000
intra_parallel NO
max_querydegree -1
diaglevel 3
svcename db2c_db2inst1;

get database manager configuration;

--connect reset;
```

load_all.sql

```
connect to tpcd;

values(current timestamp, 'TS*** Load Supplier Started ');
load from supplier.tbl of del
   MODIFIED BY COLDEL|
   FASTPARSE ANYORDER MESSAGES
/home/db2inst1/tpcd/temp/supplier.msg
   REPLACE INTO TPCD.SUPPLIER NONRECOVERABLE DATA BUFFER
256 CPU_PARALLELISM 1
   partitioned db config mode load_only output_dbpartnums (0,1,2,3)
   part_file_location /flatfiles/300GB_flatfiles;
commit work;

values(current timestamp, 'TS*** Load done partsupp ');
load from orders.tbl of del
  MODIFIED BY COLDEL| FASTPARSE ANYORDER MESSAGES
/home/db2inst1/tpcd/temp/orders.msg
  REPLACE INTO TPCD.orders NONRECOVERABLE  DATA BUFFER 256
CPU_PARALLELISM 1
  partitioned db config mode load_only output_dbpartnums (0,1,2,3)
  part_file_location /flatfiles/300GB_flatfiles;
commit work;

values(current timestamp, 'TS*** Load done orders ');
load from lineitem.tbl of del
    MODIFIED BY COLDEL| FASTPARSE ANYORDER MESSAGES
/home/db2inst1/tpcd/temp/lineitem.msg
    REPLACE INTO TPCD.lineitem NONRECOVERABLE DATA BUFFER
256 CPU_PARALLELISM 1
    partitioned db config mode load_only output_dbpartnums (0,1,2,3)
    part_file_location /flatfiles/300GB_flatfiles;
commit work;

values(current timestamp, 'TS*** Load done supplier ');
load from customer.tbl of del
   MODIFIED BY COLDEL|
   FASTPARSE ANYORDER MESSAGES
/home/db2inst1/tpcd/temp/customer.msg
   REPLACE INTO TPCD.customer NONRECOVERABLE DATA BUFFER
256 CPU_PARALLELISM 1
   partitioned db config mode load_only output_dbpartnums (0,1,2,3)
   part_file_location /flatfiles/300GB_flatfiles;
commit work;

values(current timestamp, 'TS*** Load done customer ');
load from part.tbl of del
   MODIFIED BY COLDEL| FASTPARSE ANYORDER MESSAGES
/home/db2inst1/tpcd/temp/part.msg
   REPLACE INTO TPCD.part NONRECOVERABLE DATA BUFFER 256
CPU_PARALLELISM 1
   partitioned db config mode load_only output_dbpartnums (0,1,2,3)
   part_file_location /flatfiles/300GB_flatfiles;
commit work;

values(current timestamp, 'TS*** Load done part ');
load from partsupp.tbl of del
       MODIFIED BY COLDEL| FASTPARSE ANYORDER MESSAGES
/home/db2inst1/tpcd/temp/partsupp.msg
       REPLACE INTO TPCD.partsupp NONRECOVERABLE DATA BUFFER
256 CPU_PARALLELISM 1
       partitioned db config mode load_only output_dbpartnums (0,1,2,3)
       part_file_location /flatfiles/300GB_flatfiles;
commit work;
values(current timestamp, 'TS*** Load done lineitem ');
LOAD FROM /flatfiles/300GB_flatfiles/region.tbl OF DEL
    MODIFIED BY COLDEL| FASTPARSE ANYORDER MESSAGES
/home/db2inst1/tpcd/temp/region.msg
    REPLACE INTO TPCD.REGION STATISTICS NO NONRECOVERABLE;
commit work;

values(current timestamp, 'TS*** Load done region ');
LOAD FROM /flatfiles/300GB_flatfiles/nation.tbl OF DEL
    MODIFIED BY COLDEL| FASTPARSE ANYORDER MESSAGES
/home/db2inst1/tpcd/temp/nation.msg
    REPLACE INTO TPCD.NATION STATISTICS NO NONRECOVERABLE;
values(current timestamp, 'TS*** Load done nation ');
commit work;
connect reset;
```

run.dbcfg_4mln.sql

```
update database configuration for tpcd using
buffpage  128000
database_memory 363900
catalogcache_sz 386
dbheap 20000
locklist 16384
maxlocks 25
maxappls 40
mincommit 1
```

```
num_iocleaners 1
num_ioservers  4
DLCHKTIME 5000
pckcachesz 640
logfilsiz 16384
logprimary 30
logsecond 2
softmax 1600
sortheap   18432
stat_heap_sz 10000
stmtheap 20000
util_heap_sz 40000
applheapsz 16000
logbufsz 2048
APPGROUP_MEM_SZ 2048
app_ctl_heap_sz 2048
dft_degree 1
dft_queryopt 7
maxfilop 1024
chngpgs_thresh 60
NUM_QUANTILES 300;

get database configuration for tpcd;

--connect reset;
```

run.dbmcfg_4mln.sql

```
update database manager configuration using
cpuspeed 1.889377e-07
comm_bandwidth 0.1
sheapthres  1250000
aslheapsz 15
rqrioblk 32767
intra_parallel no
max_querydegree -1
maxagents 256
num_poolagents 64
num_initagents 4
fcm_num_buffers 30000
numdb 1
svcename db2c_db2inst1
DFT_MON_TIMESTAMP OFF
HEALTH_MON OFF
diaglevel 0;
get database manager configuration;
```

run_db2set.pl

```
#!/usr/bin/perl


@SETTINGS=(
"DB2_LGPAGE_BP=yes",
"DB2LINUXAIO=TRUE",
"DB2_LIKE_VARCHAR=Y,Y",
"DB2COMM=tcpip",
"DB2AUTOSTART=FALSE",
"DB2_EXTENDED_OPTIMIZATION=Y",
"DB2_ANTIJOIN=Y",
"DB2OPTIONS=\"-t -v +c\"",
"DB2BPVARS=/home/db2inst1/tpcd/custom/bpvars.txt",
"DB2_PARALLEL_IO=*");

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

setlogs.pl

```
#!/usr/bin/perl
# Set the new log path for the database
# Make output unbuffered.
select(STDOUT);
$| = 1 ;
$dbname="tpcd";



  system("db2_all \"<<+000 <db2 update db cfg for tpcd using newlogpath
/dev/sda4\" ");
  system("db2_all \"<<+001 <db2 update db cfg for tpcd using newlogpath
/dev/sdb1\" ");
  system("db2_all \"<<+002 <db2 update db cfg for tpcd using newlogpath
/dev/sdc1\" ");
  system("db2_all \"<<+003 <db2 update db cfg for tpcd using newlogpath
/dev/sdd1\" ");

$ret;
```

verifytpcdbatch.clp

```
connect to TPCD;
select name,creator,valid,last_bind_time,isolation from sysibm.sysplan where
name like 'TPCD%';
connect reset;
terminate;
```

temp_4K

```
alter bufferpool ibmdefaultbp size 317440;
alter bufferpool ibmdefaultbp numblockpages 0;
alter bufferpool BP32K size 128000;
alter bufferpool BP32K numblockpages 16768;
alter bufferpool BP32K blocksize 24;
alter bufferpool BP32K size -1;
commit;
connect reset;
db2stop;
db2start;
connect to tpcd;
drop tablespace userspace1;
commit;
drop tablespace tempspace1;
commit;
connect reset;
db2stop force;
Db2start;
```

tpcd.setup

```
# NOTE: ALL variable defitions must have a comment at the end.
TPCD_PLATFORM=linux                  # aix, nt, sun ....
TPCD_VERSION=2                  # 1 or 2 (Version of tpcd). Default 1
TPCD_DBNAME=TPCD                     # name to create database under
TPCD_WORKLOAD=H                      # TPC version (R for TPCR, H for
TPCH)
TPCD_AUDIT_DIR=/home/db2inst1/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=mln                   # uni/smp/mln/mpp
TPCD_PHYS_NODE=1                     # number of physical nodes
TPCD_LN_PER_PN=4                     # 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=/home/db2inst1/tpcd          # path for database (defaults to
home)
TPCD_DDLPATH=/home/db2inst1/tpcd/custom  # path for all ddl files and
customized
                        # scripts (load script), config files,etc
TPCD_BUFFERPOOL_DEF=create_bufferpools # name of file 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=create_tablespaces       # ddl file for tablespaces
TPCD_DDL=create_tables                 # ddl file for tables
TPCD_QUAL_TBSP_DDL=create_tablespaces  # ddl file for tablespaces for qual
TPCD_QUAL_DDL=create_tables            # ddl file for qualification
database
                        # tablespaces and tables should be identical
                        # to regular ddl except container names
TPCD_INDEXDDL=create_indexes           # 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
TPCD_ADD_RI=NULL                       # file name that contains any RI
                        # constraints to add after index creation
                        # set to NULL (default) if unused
                        # indices to create after the load.
TPCD_AST=NULL                          # file name that contains complete AST
                        # definition including connection to
                        # the database, summary table creation,
                        # population, indexing and runstats.
TPCD_RUNSTATS=dss.runstats             # 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                 # NOTE!! THIS IS BUGGY....I
can't get it to
                        # work on UNI successfully
                        # 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=/home/db2inst1/tpcd/appendix.v230/dbgen # path name to data
gen code
                        # Parameters used to specify source of
                        # data for load scripts
TPCD_INPUT=/flatfiles/300GB_flatfiles  # NULL - use dbgen generated data
OR
                        # path name -  to the pre-generated
                        #            flat files
                        # /gwl/dss/12MB - path for pregenerated 12MB
                        # /gwl/dss/100MB - path for pregen'd 100MB
                        #
TPCD_QUAL_INPUT=/flatfiles/1GB_QUAL_FLATFILES   # NULL - use
dbgen generated data  OR
                        # path name -  to the pre-generated
                        #            flat files
TPCD_TAILOR_DIR=/home/db2inst1/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.dbcfg.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.dbmcfg.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.dbcfg.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.dbmcfg.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=/tmp/tpch                    # 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=\backupdir             # directory where backup files are
placed
TPCD_LOGPRIMARY=NULL                   # NULL/value = how many primary
log files
                        # to configure.  If NULL is specified then
                        # the default is not changed.
TPCD_LOGFILSIZ=NULL                    # NULL/value = how 4KB pages to use
for
                        # logfilsiz db cfg parameter.  If NULL is
                        # specified then the default is not changed
```

```
TPCD_LOGSECOND=NULL                # NULL/value = how many secondary log files
                                   # to configure.  If NULL is specified then
                                   # the default is not changed.
TPCD_LOG_DIR=/home/db2inst1/logs   # directory where log files stored..
                                   # NULL leaves them in the dbpath
TPCD_LOG_DIR_SETUP_SCRIPT=setlogs.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=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.dbcfg_4mln.sql  # name of configuration file in ddl path
                                   # that will be used for the benchmark run
TPCD_DBM_CONFIG=run.dbmcfg_4mln.sql  # name of config file for database manager
                                   # cfg parms
TPCD_QUALCONFIGFILE=run.dbcfg_qual_4mln.sql # name of database cfg file in ddl path
                                   # for qualification database
TPCD_DBM_QUALCONFIG=run.dbmcfg_4mln.sql # name of config file for database
                                   # manager cfg parms

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=yes                  # activate the database upon build
                                   # completion
                                   # run specific parameters
TPCD_AUDIT=yes                     # 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/tpch             # place to put temp working files
TPCD_SHARED_TEMP_FULL_PATHNAME=d:\sqllib\tmp # just added
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=/300GB_4mln_UF_flatfiles # where to generate flat files
                                   # for update functions
TPCD_STAGING_TABLE_DDL=createUFtables  # script 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 containing
                                   # 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=DELETEDUMMYUFDATA.SQL # 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=256             # number of chunks to split the update
                                   # function into.
TPCD_CONCURRENT_INSERTS=32         # number of insert chunks that are run
                                   # concurrently.  TPCD_SPLIT_UPDATES
                                   # should be evenly divisible by this number
TPCD_CONCURRENT_INSERTS_LOAD=16    # number of insert chunks that are loaded
                                   # concurrently.  TPCD_SPLIT_UPDATES should
                                   # be evenly divisible by this number.
                                   # this controls the load portion of the
                                   # insert routine for partitioned databases
TPCD_SPLIT_DELETES=256             # number of portions to split the delete
                                   # function into.
                                   # this variable is only valid in UNI/SMP
                                   # mode.
TPCD_CONCURRENT_DELETES=32         # number of DELETE chunks that are run
                                   # concurrently.  TPCD_SPLIT_UPDATES
                                   # should be evenly divisible by this number
TPCD_GEN_UPDATEPAIRS=18            # 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=NO      # pe V1.2 only - will we be running each
                                   # query stream of throughput starting at
                                   # different nodes or from same node
TPCD_STATS_INTERVAL=3              # timing interval for vmstats/iostats
TPCD_STATS_THRU_INT=300            # timing interval for vmstats/iostats for
                                   # throughput run
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=load_all.sql          # script to run for loading tables
                        # in TPCD_DDLPATH directory under mln/mpp
                        # leave as NULL if using default genloaduni
TPCD_LOAD_SCRIPT_QUAL=load_all_qualdb.sql  # 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)
                        #  acid test specific information
TPCD_DB2LOG=d:\sqllib\db2              # directory where the db2diag.log can
                        # be found for the durability tests
TPCD_APPEND_ON=no                   # set to no if the cluster indexes are used
```

Appendix C: Qualification Query Output

Qualification Queries

Query 1

Start timestamp 09/15/06 10:19:48.960621

-- 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      56586554400.729
53758257134.869      55909065222.828              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     111701729697.743
106118230307.606     110367043872.499              25.502           38249.118
        0.050     2920374
R           F                37719753.000      56568041380.899
53741292684.604      55889619119.832              25.506           38250.855
     0.050     1478870


Number of rows retrieved is:     4
---------------------------------------------
```

Stop timestamp 09/15/06 10:20:04.928542
Query Time = 16.0 secs

Query 2

Start timestamp 09/15/06 10:18:29.250100

-- 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

  ...87 lines deleted

        7843.520  Supplier#000006683        FRANCE                    11680
Manufacturer#4           2Z0JGkiv01Y00oCFwUGfviIbhzCdy
16-464-517-8943  carefully bold accounts doub



Number of rows retrieved is:    100
------------------------------------------------


Stop timestamp 09/15/06 10:18:29.478152
Query Time =            0.2 secs
```

Query 3

```
Start timestamp 09/15/06 10:19:25.365522

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

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


Stop timestamp 09/15/06 10:19:42.521730
Query Time =            17.2 secs
```

Query 4

```
Start timestamp 09/15/06 10:19:43.240626

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

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

```
Stop timestamp 09/15/06 10:19:48.581930
Query Time =            5.3 secs
```

Query 5

```
Start timestamp 09/15/06 10:20:15.386180

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

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

```
Stop timestamp 09/15/06 10:20:23.342191
Query Time =            8.0 secs
```

Query 6

```
Start timestamp 09/15/06 10:18:44.943740

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

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

```
Stop timestamp 09/15/06 10:18:46.807947
Query Time =            1.9 secs
```

Query 7

```
Start timestamp 09/15/06 10:20:23.342191

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

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


Stop timestamp 09/15/06 10:20:28.911270
Query Time =            5.6 secs
```

Query 8

```
Start timestamp 09/15/06 10:18:57.002771

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

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


Stop timestamp 09/15/06 10:19:07.919691
Query Time =           10.9 secs
```

Query 9

```
Start timestamp 09/15/06 10:18:29.478152

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

-- 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       31342867.235
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

... 161 lines deleted

VIETNAM                     1993       50953919.152
VIETNAM                     1992       49613838.315


Number of rows retrieved is:    175
------------------------------------------------


Stop timestamp 09/15/06 10:18:42.700477
Query Time =            13.2 secs
```

Query 10

```
Start timestamp 09/15/06 10:20:04.928542

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

-- 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               734235.245            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

  ... 8 lines deleted

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

Stop timestamp 09/15/06 10:20:10.330676
Query Time = 5.4 secs

Query 11

Start timestamp 09/15/06 10:19:48.581930

-- 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

... 1033 lines deleted

     51968         7879102.210
     72073         7877736.110
      5182        7874521.730


Number of rows retrieved is:  1048
-----------------------------------------------
```

Stop timestamp 09/15/06 10:19:48.708039
Query Time = 0.1 secs

Query 12

Start timestamp 09/15/06 10:20:28.911270

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

Stop timestamp 09/15/06 10:20:34.263602
Query Time = 5.4 secs

Query 13

Start timestamp 09/15/06 10:19:24.022860

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

```
Stop timestamp 09/15/06 10:19:25.365522
Query Time =              1.3 secs
```

Query 14

```
Start timestamp 09/15/06 10:18:27.846300

-----------------------------------------------
--#SET ROWS_OUT -1 ROWS_FETCH -1

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


Stop timestamp 09/15/06 10:18:29.250100
Query Time =              1.4 secs
```

Query 15

```
Start timestamp 09/15/06 10:19:48.708039

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

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


Stop timestamp 09/15/06 10:19:48.960621
Query Time =             0.3 secs
```

Query 16

```
Start timestamp 09/15/06 10:19:42.769234

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

-- 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

... 18299 lines deleted


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


Stop timestamp 09/15/06 10:19:43.240626
Query Time =             0.5 secs
```

Query 17

```
Start timestamp 09/15/06 10:18:46.807947

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

-- 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
```

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


Stop timestamp 09/15/06 10:18:51.884225
Query Time =            5.1 secs
```

Query 18

```
Start timestamp 09/15/06 10:18:51.884225

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

-- 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

...  43 lines deleted

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


Stop timestamp 09/15/06 10:18:57.002771
Query Time =            5.1 secs
```

Query 19

```
Start timestamp 09/15/06 10:20:10.330676

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

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


Stop timestamp 09/15/06 10:20:15.386180
Query Time =              5.1 secs
```

Query 20

```
Start timestamp 09/15/06 10:18:42.700477

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

-- 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

... 189 lines delted

Supplier#000009869
ucLqxzrpBTRMewGSM29t0rNTM30g1Tu3Xgg3mKag
Supplier#000009899        7XdpAHrzr1t,UQFZE
Supplier#000009974        7wJ,J5DKcxSU4Kp1cQLpbcAvB5AsvKT


Number of rows retrieved is:    204
-----------------------------------------------


Stop timestamp 09/15/06 10:18:44.943740
Query Time =              2.2 secs
```

Query 21

```
Start timestamp 09/15/06 10:19:07.919691

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

-- 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

...  85 lines deleted

Supplier#000002039                12
Supplier#000002357                12
Supplier#000002483                12


Number of rows retrieved is:    100
------------------------------------------------


Stop timestamp 09/15/06 10:19:24.022860
Query Time =            16.1 secs
```

Query 22

```
Start timestamp 09/15/06 10:19:42.521730

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

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


Stop timestamp 09/15/06 10:19:42.769234
Query Time =            0.2 secs
```

First 10 Rows of the Database

```
connect to TPCD

   Database Connection Information

 Database server        = DB2/LINUXX8664 8.2.5
 SQL authorization ID   = DB2INST1
 Local database alias   = TPCD


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 integrate 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 fluffily 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 requests

       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
---------- ---------------------------------------------------------- -------------------------
---------- ------------------------- ----------- ----------- ----------------------
---------------------
        10 floral moccasin royal powder burnished                Manufacturer#5
 Brand#54   LARGE BURNISHED STEEL             44 LG CAN
+9.10010000000000E+002 bold, ironic
        11 chocolate turquoise sandy snow misty                  Manufacturer#2
Brand#25   STANDARD BURNISHED NICKEL         43 WRAP BOX
+9.11010000000000E+002 furiousl
        14 linen seashell burnished blue gainsboro               Manufacturer#1
Brand#13   SMALL POLISHED STEEL              28 JUMBO BOX
+9.14010000000000E+002 special, regular
        37 turquoise ivory orange sandy maroon                   Manufacturer#4
Brand#45   LARGE POLISHED TIN               48 JUMBO BOX
+9.37030000000000E+002 blithely regular
        39 rose dodger lace peru floral                          Manufacturer#5
Brand#53   SMALL POLISHED TIN               43 JUMBO JAR
+9.39030000000000E+002 quickly bo
        43 medium khaki chocolate rosy blush                     Manufacturer#4
Brand#44   PROMO POLISHED STEEL              5 WRAP CASE
+9.43040000000000E+002 carefully iro
        47 sky firebrick red linen dim                           Manufacturer#4
Brand#45   LARGE BURNISHED BRASS             14 JUMBO PACK
+9.47040000000000E+002 bold, unusual a
        49 blue tan cornsilk burlywood beige                     Manufacturer#2
Brand#24   SMALL BURNISHED TIN               31 MED DRUM
+9.49040000000000E+002 carefu
        50 yellow cornflower royal blush almond                  Manufacturer#3
Brand#33   LARGE ANODIZED TIN                25 WRAP PKG
+9.50050000000000E+002 regular dinos ar
        55 antique cream pale tomato rose                        Manufacturer#2
Brand#23   ECONOMY BRUSHED COPPER            9 MED BAG
+9.55050000000000E+002 furiously

  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

----------- ------------------------- ---------------------------------------- -----------
--------------- ------------------------
--------------------------------------------------------------------------------------------------------------------------------
--
        10 Supplier#000000010        Saygah3gYWMp72i PY                      24
34-852-489-8585   +3.89191000000000E+003 ironic deposits poach quickly furiously final accounts. carefull
        11 Supplier#000000011        JfwTs,LZrV, M,9C                        18
28-613-996-1505   +3.39308000000000E+003 quickly bold asymptotes mold carefully unusual pearls. requests boost at the blith
        14 Supplier#000000014        EXsnO5pTNj4iZRm                         15
25-656-247-5058   +9.18982000000000E+003 enticingly bold platelets wake against the
        37 Supplier#000000037        cqjyB5h1nV                              0
10-470-144-1330   +3.01747000000000E+003 ironic requests alongside of the blithely express accounts wake fluffily above the carefully express
        39 Supplier#000000039        ZM, nSYpEPWr1yAFHaC91qjFcijjeU5eH
         8 18-851-856-5633   +6.11565000000000E+003 ironic, express instructions can sleep bl
        43 Supplier#000000043
Z5mLuAoTUEeKY5v22VnnA4D87Ao6jF2LvMYnlX8h          12
22-421-568-4862   +7.77341000000000E+003 slyly final accounts wake blithely slyly regular requests. sl
        47 Supplier#000000047        3XM1x,Pcxqw,HK4XNlgbnZMbLhBHLA
          14 24-810-354-4471   +2.95809000000000E+003 regular deposits engage slyly with the
        49 Supplier#000000049        Nvq 6macF4GtJvz                         24
34-211-567-6800   +9.91524000000000E+003 blithely silent pinto beans hang slyly. blithely eve
        50 Supplier#000000050        rGobqSMMYz0ErrPhCGS
9 19-561-560-7437   +4.51587000000000E+003 quickly regular theodolites wake within the slyly even accounts. furiously express accounts
        55 Supplier#000000055        OqdYSiOQeG4eGi636Tj                     24
34-876-912-6007   +7.16215000000000E+003 special, express deposits against the bold, fluffy pinto beans are even, re

  10 record(s) selected.


SELECT * FROM TPCD.PARTSUPP FETCH FIRST 10 ROWS ONLY

PS_PARTKEY  PS_SUPPKEY  PS_AVAILQTY PS_SUPPLYCOST
PS_COMMENT

----------- ----------- ----------- ----------------------
--------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
-
        10         11        2952   +9.96120000000000E+002 blithely even foxes nag furiously about the quickly ex

        10      750011        3335   +6.73270000000000E+002 final, regular foxes cajole carefully about the blithely express accounts. carefully regular platelets
```

against the silent pinto beans sleep carefully among the blithely regular foxes. final r
10 1500011 5691 +1.64000000000000E+002 carefully express accounts wake ruthlessly. carefully ironic frets haggle furi

10 2250011 841 +3.74020000000000E+002 pending, pending requests may haggle sometimes. silent pinto beans are blithe

11 12 4540 +7.09870000000000E+002 final packages mold after the carefully unusual requests. quickly fi

11 750012 4729 +8.94900000000000E+002 regular packages sleep carefully fluffily ironic ac

11 1500012 3708 +8.18740000000000E+002 slyly pending theodolites wake quickly unusual, express accounts. fluffily regular requests cajole furiously quickly even dugouts. slyly bold platelets

11 2250012 3213 +4.71980000000000E+002 ideas nag regular instructions. regular, thin pinto beans unwind furiously ironic accounts. quickly express platele
14 15 5278 +6.50070000000000E+002 quickly even deposits doze quickly pending, bold deposits. carefully regular packages sublate carefully

14 750015 5334 +8.89500000000000E+002 express instructions affix quickly. slyly bold requests use. special, express foxes haggle fluffily express deposits. silently even pinto beans throughout the blithely iron

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

---

10 Customer#000000010 6LrEaV6KR6PLVcgl2ArL Q3rqzLzcT1 v2 5 15-741-346-9870 +2.75354000000000E+003 HOUSEHOLD bold, final frays sleep carefully special ideas. carefully final asymptotes sleep furiously against the even i
11 Customer#000000011 PkWS 3HlXqwTuzrKg633BEi 23 33-464-151-3439 -2.72600000000000E+002 BUILDING furiously express packages are. regular courts play deposits. silent, ironic packages engage. furiously regular a
14 Customer#000000014 KXkletMlL2JQEA 1 11-845-129-3851 +5.26630000000000E+003 FURNITURE never regular pinto beans boost bl
37 Customer#000000037 7EV4Pwh,3SboctTWt 8 18-385-235-7162 -9.17750000000000E+002 FURNITURE ironic, special instructions detect quickly above the even theodolites. quickly unusual requests maintain
39 Customer#000000039 nnbRg,Pvy33dfkorYE FdeZ60 2 12-387-467-6509 +6.26431000000000E+003 AUTOMOBILE theodolites sleep ironic foxes. fluffily ironic requests detect carefu

43 Customer#000000043 ouSbjHk8lh5fKX3zGso3ZSIj9Aa3PoaFd 19 29-316-665-2897 +9.90428000000000E+003 MACHINERY idly regular sentiments affix. slyly pending foxes around the deposits haggle according to th
47 Customer#000000047 b0UgocSqEW5 gdVbhNT 2 12-427-271-9466 +2.74580000000000E+002 BUILDING blithely final deposits about the furiously special packages play even idea

49 Customer#000000049 cNgAeX7Fqrdf7HQN9EwjUa4nxT,68L FKAxzl 10 20-908-631-4424 +4.57394000000000E+003 FURNITURE thin ideas are foxes. furiously regular t

50 Customer#000000050 9SzDYlkzxByyJ1QeTI o 6 16-658-112-3221 +4.26613000000000E+003 MACHINERY express, bold frets boost among the express ideas. final, express requests boost platelets.

55 Customer#000000055 zIRBR4KNEl HzaiV3a i9n6elrxzDEh8r8pDom 10 20-180-440-8525 +4.57211000000000E+003 MACHINERY carefully regular requests against the fluffily special accounts cajole slyly slyly

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

59718 7424651 F +1.12649120000000E+005 01/01/1992 5-LOW Clerk#000269045 0 carefully regular pinto beans across the even grouches dete
183488 34807951 F +2.50627220000000E+005 01/01/1992 3-MEDIUM Clerk#000234624 0 slyly regular dependencies sublate furiously. ironic pinto beans wake accor
249668 34734230 F +3.44531700000000E+004 01/01/1992 2-HIGH Clerk#000027034 0 quickly ironic requests sleep after the enticin
302497 40041712 F +2.76518630000000E+005 01/01/1992 1-URGENT Clerk#000181747 0 fluffily final packages haggle. carefully r
334181 8600005 F +9.40402300000000E+004 01/01/1992 2-HIGH Clerk#000061971 0 fluffily pending foxes haggle carefully furiously final pinto beans. s
360261 28122308 F +1.49784400000000E+004 01/01/1992 5-LOW Clerk#000038000 0 express tithes doze stealthily around the final requ
368004 34593859 F +9.69309000000000E+004 01/01/1992 2-HIGH Clerk#000193336 0 slyly unusual theodolites snooze pending instructions! q
391429 38916928 F +2.21903540000000E+005 01/01/1992 5-LOW Clerk#000147604 0 carefully ironic requests affix carefully pend
414725 10491310 F +2.65019800000000E+004 01/01/1992 5-LOW Clerk#000257604 0 sly accounts detect along the slyly express
463072 16276666 F +3.31911040000000E+005 01/01/1992 1-URGENT Clerk#000030587 0 furiously ironic pearls along the slyly bol

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+001
+4.69273500000000E+004  +3.00000000000000E-002
+8.00000000000000E-002 R          F          01/02/1992 02/05/1992  01/15/1992
  NONE                  MAIL      even instructions kindle furio
   5018977   24611600    611601          1  +2.00000000000000E+001
+3.02074000000000E+004  +0.00000000000000E+000
+0.00000000000000E+000 A          F          01/02/1992 03/19/1992
01/15/1992  NONE                  SHIP      quickly ironic excu
   5431079   23660609   1910631          1  +1.20000000000000E+001
+1.88210400000000E+004  +3.00000000000000E-002
+8.00000000000000E-002 R          F          01/02/1992 03/29/1992  02/01/1992
  NONE                  MAIL      permanent, final requests sleep along th
  12493984   27378769   2628797          4  +6.00000000000000E+000
+1.10784000000000E+004  +2.00000000000000E-002
+8.00000000000000E-002 A          F          01/02/1992 02/13/1992  01/29/1992
  TAKE BACK RETURN        REG AIR   regular packages haggle. quickly

  14168833   46737982    988028          3  +4.90000000000000E+001
+9.88648500000000E+004  +9.00000000000000E-002
+5.00000000000000E-002 R          F          01/02/1992 02/01/1992  01/28/1992
  TAKE BACK RETURN        SHIP      furiously bold courts are careful
  15413986   53978084   2978085          4  +1.40000000000000E+001
+1.62314600000000E+004  +5.00000000000000E-002
+5.00000000000000E-002 A          F          01/02/1992 01/31/1992  01/04/1992
  COLLECT COD             TRUCK     final, bold dolphins
  16579717    7957247   1207254          1  +2.00000000000000E+000
+2.60770000000000E+003  +9.00000000000000E-002
+3.00000000000000E-002 R          F          01/02/1992 03/27/1992  01/17/1992
  TAKE BACK RETURN        MAIL      theodolites affix furiousl
  17035334   53238361   2238362          6  +2.00000000000000E+001
+2.59340000000000E+004  +5.00000000000000E-002
+1.00000000000000E-002 R          F          01/02/1992 02/06/1992  01/05/1992
  NONE                  SHIP      furiously ironic dependencies
  17294055   31936046   1186077          5  +8.00000000000000E+000
+8.64360000000000E+003  +6.00000000000000E-002
+5.00000000000000E-002 R          F          01/02/1992 02/02/1992  01/04/1992
  NONE                  TRUCK     bold pinto beans by the fluffily final
  18436929   39951321   2451348          1  +5.00000000000000E+000
+6.85165000000000E+003  +0.00000000000000E+000
+7.00000000000000E-002 A          F          01/02/1992 02/27/1992  01/26/1992
  TAKE BACK RETURN        SHIP      even, regular platelets na

 10 record(s) selected.
```

Seeds and Query Substitution Parameters

```
Power stream    Seed = 929201209
-- TPC TPC-H Parameter Substitution (Version 2.3.0 )
-- using 929201209 as a seed to the RNG
Q1  DELTA       107
Q2  SIZE        46
   TYPE        STEEL
   REGION      AFRICA
Q3  SEGMENT     AUTOMOBILE
   DATE        1995-03-04
Q4  DATE        1993-08-01
Q5  REGION      EUROPE
   DATE        1994-01-01
Q6  DATE        1994-01-01
   DISCOUNT    0.07
   QUANTITY    25
Q7  NATION1     MOZAMBIQUE
   NATION2     KENYA
Q8  NATION      KENYA
   REGION      AFRICA
   TYPE        PROMO POLISHED BRASS
Q9  COLOR       turquoise
Q10 DATE        1993-03-01
Q11 NATION      RUSSIA
   FRACTION    0.0000003333
Q12 SHIPMODE1   AIR
   SHIPMODE2   SHIP
   DATE        1997-01-01
Q13 WORD1       express
   WORD2       deposits
Q14 DATE        1997-09-01
Q15 DATE        1993-05-01
Q16 BRAND       Brand#15
   TYPE        SMALL BURNISHED
   SIZE1       12
   SIZE2       11
   SIZE3       19
   SIZE4       16
   SIZE5       24
   SIZE6       34
   SIZE7       30
   SIZE8       9
Q17 BRAND       Brand#52
   CONTAINER   WRAP PACK
Q18 QUANTITY    312
Q19 BRAND1      Brand#52
   BRAND2      Brand#44
   BRAND3      Brand#44
   QUANTITY1   2
   QUANTITY2   17
   QUANTITY3   23
Q20 COLOUR      cream
   DATE        1996-01-01
   NATION      MOROCCO
Q21 NATION      MOROCCO
Q22 I1          20
   I2          10
   I3          13
   I4          24
   I5          21
   I6          18
   I7          16

Throughput Stream = 1    Seed = 929201210
-- TPC TPC-H Parameter Substitution (Version 2.3.0 )
-- using 929201210 as a seed to the RNG
Q1  DELTA       115
Q2  SIZE        34
   TYPE        BRASS
   REGION      EUROPE
Q3  SEGMENT     FURNITURE
   DATE        1995-03-21
Q4  DATE        1996-03-01
Q5  REGION      MIDDLE EAST
   DATE        1994-01-01
Q6  DATE        1994-01-01
   DISCOUNT    0.04
   QUANTITY    24
Q7  NATION1     INDIA
   NATION2     FRANCE
Q8  NATION      FRANCE
   REGION      EUROPE
   TYPE        PROMO BURNISHED BRASS
Q9  COLOR       snow
Q10 DATE        1993-12-01
Q11 NATION      IRAN
   FRACTION    0.0000003333
```

Q12 SHIPMODE1 REG AIR
SHIPMODE2 SHIP
DATE 1997-01-01
Q13 WORD1 express
WORD2 deposits
Q14 DATE 1997-12-01
Q15 DATE 1995-11-01
Q16 BRAND Brand#45
TYPE LARGE POLISHED
SIZE1 9
SIZE2 23
SIZE3 17
SIZE4 2
SIZE5 13
SIZE6 27
SIZE7 36
SIZE8 21
Q17 BRAND Brand#54
CONTAINER WRAP DRUM
Q18 QUANTITY 314
Q19 BRAND1 Brand#54
BRAND2 Brand#22
BRAND3 Brand#43
QUANTITY1 8
QUANTITY2 18
QUANTITY3 20
Q20 COLOUR olive
DATE 1994-01-01
NATION ETHIOPIA
Q21 NATION INDIA
Q22 I1 10
I2 28
I3 12
I4 27
I5 18
I6 15
I7 11

Throughput Stream = 2 Seed = 929201211
-- TPC TPC-H Parameter Substitution (Version 2.3.0)
-- using 929201211 as a seed to the RNG
Q1 DELTA 62
Q2 SIZE 22
TYPE NICKEL
REGION AFRICA
Q3 SEGMENT AUTOMOBILE
DATE 1995-03-06
Q4 DATE 1993-12-01
Q5 REGION AFRICA
DATE 1994-01-01
Q6 DATE 1994-01-01
DISCOUNT 0.02
QUANTITY 25
Q7 NATION1 ALGERIA
NATION2 UNITED KINGDOM
Q8 NATION UNITED KINGDOM
REGION EUROPE
TYPE ECONOMY BRUSHED STEEL
Q9 COLOR sandy
Q10 DATE 1994-09-01
Q11 NATION UNITED KINGDOM
FRACTION 0.0000003333
Q12 SHIPMODE1 SHIP
SHIPMODE2 REG AIR
DATE 1993-01-01
Q13 WORD1 express
WORD2 packages
Q14 DATE 1993-03-01
Q15 DATE 1993-08-01
Q16 BRAND Brand#35
TYPE PROMO BRUSHED
SIZE1 2
SIZE2 1
SIZE3 3
SIZE4 40
SIZE5 20
SIZE6 19
SIZE7 13
SIZE8 44
Q17 BRAND Brand#55
CONTAINER SM BOX
Q18 QUANTITY 315
Q19 BRAND1 Brand#11
BRAND2 Brand#14
BRAND3 Brand#42
QUANTITY1 3
QUANTITY2 19
QUANTITY3 27
Q20 COLOUR azure
DATE 1993-01-01
NATION ROMANIA
Q21 NATION ALGERIA
Q22 I1 29
I2 26
I3 18
I4 16
I5 14
I6 11
I7 27

Throughput Stream = 3 Seed = 929201212
-- TPC TPC-H Parameter Substitution (Version 2.3.0)
-- using 929201212 as a seed to the RNG
Q1 DELTA 70
Q2 SIZE 9
TYPE TIN
REGION EUROPE
Q3 SEGMENT FURNITURE
DATE 1995-03-23
Q4 DATE 1996-07-01
Q5 REGION AMERICA
DATE 1995-01-01
Q6 DATE 1995-01-01
DISCOUNT 0.07
QUANTITY 25
Q7 NATION1 PERU
NATION2 MOROCCO
Q8 NATION MOROCCO
REGION AFRICA
TYPE ECONOMY PLATED STEEL
Q9 COLOR red
Q10 DATE 1993-06-01
Q11 NATION IRAQ
FRACTION 0.0000003333
Q12 SHIPMODE1 FOB
SHIPMODE2 REG AIR
DATE 1993-01-01
Q13 WORD1 special
WORD2 packages
Q14 DATE 1993-06-01
Q15 DATE 1996-03-01
Q16 BRAND Brand#15
TYPE MEDIUM BURNISHED
SIZE1 40
SIZE2 18
SIZE3 5

SIZE4 28
SIZE5 30
SIZE6 11
SIZE7 10
SIZE8 48
Q17 BRAND Brand#52
CONTAINER SM PACK
Q18 QUANTITY 313
Q19 BRAND1 Brand#13
BRAND2 Brand#42
BRAND3 Brand#32
QUANTITY1 8
QUANTITY2 20
QUANTITY3 23
Q20 COLOUR lawn
DATE 1996-01-01
NATION INDONESIA
Q21 NATION PERU
Q22 I1 16
I2 24
I3 15
I4 29
I5 33
I6 12
I7 21

Throughput Stream = 4 Seed = 929201213
-- TPC TPC-H Parameter Substitution (Version 2.3.0)
-- using 929201213 as a seed to the RNG
Q1 DELTA 78
Q2 SIZE 47
TYPE STEEL
REGION AMERICA
Q3 SEGMENT MACHINERY
DATE 1995-03-08
Q4 DATE 1994-04-01
Q5 REGION ASIA
DATE 1995-01-01
Q6 DATE 1995-01-01
DISCOUNT 0.05
QUANTITY 24
Q7 NATION1 INDONESIA
NATION2 GERMANY
Q8 NATION GERMANY
REGION EUROPE
TYPE ECONOMY ANODIZED STEEL
Q9 COLOR peru
Q10 DATE 1994-04-01
Q11 NATION UNITED STATES
FRACTION 0.0000003333
Q12 SHIPMODE1 TRUCK
SHIPMODE2 REG AIR
DATE 1993-01-01
Q13 WORD1 special
WORD2 packages
Q14 DATE 1993-09-01
Q15 DATE 1993-12-01
Q16 BRAND Brand#45
TYPE ECONOMY PLATED
SIZE1 10
SIZE2 29
SIZE3 8
SIZE4 19
SIZE5 27
SIZE6 49
SIZE7 15
SIZE8 22
Q17 BRAND Brand#54
CONTAINER SM DRUM
Q18 QUANTITY 315
Q19 BRAND1 Brand#15
BRAND2 Brand#25
BRAND3 Brand#31
QUANTITY1 3
QUANTITY2 10
QUANTITY3 30
Q20 COLOUR smoke
DATE 1994-01-01
NATION UNITED STATES
Q21 NATION INDONESIA
Q22 I1 22
I2 25
I3 23
I4 28
I5 12
I6 30
I7 15

Throughput Stream = 5 Seed = 929201214
-- TPC TPC-H Parameter Substitution (Version 2.3.0)
-- using 929201214 as a seed to the RNG
Q1 DELTA 86
Q2 SIZE 35
TYPE BRASS
REGION EUROPE
Q3 SEGMENT BUILDING
DATE 1995-03-25
Q4 DATE 1996-11-01
Q5 REGION EUROPE
DATE 1995-01-01
Q6 DATE 1995-01-01
DISCOUNT 0.02
QUANTITY 25
Q7 NATION1 ARGENTINA
NATION2 UNITED STATES
Q8 NATION UNITED STATES
REGION AMERICA
TYPE LARGE POLISHED STEEL
Q9 COLOR olive
Q10 DATE 1995-01-01
Q11 NATION JAPAN
FRACTION 0.0000003333
Q12 SHIPMODE1 RAIL
SHIPMODE2 REG AIR
DATE 1993-01-01
Q13 WORD1 special
WORD2 packages
Q14 DATE 1994-01-01
Q15 DATE 1996-06-01
Q16 BRAND Brand#35
TYPE STANDARD BRUSHED
SIZE1 33
SIZE2 15
SIZE3 4
SIZE4 49
SIZE5 23
SIZE6 46
SIZE7 9
SIZE8 42
Q17 BRAND Brand#51
CONTAINER LG BOX
Q18 QUANTITY 312
Q19 BRAND1 Brand#22
BRAND2 Brand#13
BRAND3 Brand#35
QUANTITY1 9

```
   QUANTITY2    11
   QUANTITY3    26
Q20 COLOUR      floral
   DATE         1993-01-01
   NATION       KENYA
Q21 NATION      ARGENTINA
Q22 I1          24
   I2           21
   I3           32
   I4           14
   I5           30
   I6           25
   I7           20

Throughput Stream = 6    Seed = 929201215
-- TPC TPC-H Parameter Substitution (Version 2.3.0 )
-- using 929201215 as a seed to the RNG
Q1  DELTA       94
Q2  SIZE        23
   TYPE         NICKEL
   REGION       AMERICA
Q3  SEGMENT     MACHINERY
   DATE         1995-03-10
Q4  DATE        1994-08-01
Q5  REGION      MIDDLE EAST
   DATE         1995-01-01
Q6  DATE        1995-01-01
   DISCOUNT     0.07
   QUANTITY     25
Q7  NATION1     CHINA
   NATION2      MOZAMBIQUE
Q8  NATION      MOZAMBIQUE
   REGION       AFRICA
   TYPE         LARGE BURNISHED COPPER
Q9  COLOR       midnight
Q10 DATE        1993-10-01
Q11 NATION      ALGERIA
   FRACTION     0.0000003333
Q12 SHIPMODE1   AIR
   SHIPMODE2    REG AIR
   DATE         1994-01-01
Q13 WORD1       special
   WORD2        packages
Q14 DATE        1994-04-01
Q15 DATE        1994-03-01
Q16 BRAND       Brand#15
   TYPE         LARGE ANODIZED
   SIZE1        16
   SIZE2        35
   SIZE3        38
   SIZE4        8
   SIZE5        20
   SIZE6        39
   SIZE7        18
   SIZE8        12
Q17 BRAND       Brand#53
   CONTAINER    LG PACK
Q18 QUANTITY    314
Q19 BRAND1      Brand#24
   BRAND2       Brand#41
   BRAND3       Brand#24
   QUANTITY1    4
   QUANTITY2    12
   QUANTITY3    23
Q20 COLOUR      plum
   DATE         1996-01-01
   NATION       CANADA
Q21 NATION      CHINA
Q22 I1          13
   I2           19
   I3           28
   I4           18
   I5           31
   I6           16
   I7           32
```

Appendix D: Driver Source Code

Load_line_uf

```
#!/bin/ksh
RFpair=$1;
db2 connect to tpcd
db2 "load from lineitem.tbl.new.u$RFpair of del modified by coldel| fastparse
messages /home/db2inst1/tpcd/temp/liUF.msg replace into
TPCDTEMP.LINEITEM_new nonrecoverable data buffer 256  partitioned db
config mode load_only output_dbpartnums (0,1,2,3) part_file_location
/300GB_4mln_UF_flatfiles;"
db2 commit;
db2 connect reset
db2 terminate
```

load_orders_uf

```
#!/bin/ksh
RFpair=$1;
db2 connect to tpcd
db2 "load from orders.tbl.new.u$RFpair of del modified by coldel| fastparse
messages /home/db2inst1/tpcd/temp/orUF.msg replace into
TPCDTEMP.ORDERS_new nonrecoverable data buffer 256 partitioned db config
mode load_only output_dbpartnums (0,1,2,3) part_file_location
/300GB_4mln_UF_flatfiles;"
db2 commit;
db2 connect reset
db2 terminate
```

loadSampleUf.sh

```
#!/usr/bin/ksh
# Please indicate which pair you want to use for the preloading of the
# update functions. This pair can NOT be used for the actual benchmark.
# In generat we pre-load the pair which the largest number. For example
# if we created 12 pairs, we'll preload pair 12. This number is the parameter
# passed to ploaduf1 and ploaduf2 to load the data.

toolsDir=/home/db2inst1/tpcd/tools
##toolsDir=${TPCD_AUDIT_DIR}/tools

echo " about to call ploaduf1"

#${toolsDir}/ploaduf1 21
#${toolsDir}/ploaduf2 21
/home/db2inst1/tpcd/tools/ploaduf1 18
/home/db2inst1/tpcd/tools/ploaduf2 18
##${toolsDir}/ploaduf1 28
##${toolsDir}/ploaduf2 28

db2 "connect to tpcd"
db2 "RUNSTATS ON TABLE TPCDTEMP.LINEITEM_NEW WITH
DISTRIBUTION AND DETAILED INDEXES ALL"
db2 "RUNSTATS ON TABLE TPCDTEMP.ORDERS_NEW WITH
DISTRIBUTION AND DETAILED INDEXES ALL"
db2 "RUNSTATS ON TABLE TPCDTEMP.ORDERS_DEL WITH
DISTRIBUTION AND DETAILED INDEXES ALL"
db2 "terminate"
```

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=$(HOME)/sqllib
COMPILE_FLAGS= -c -DSQLAIX -DLINUX -I$(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= -o $@ -L$(BASE)/lib -ldb2
#LINK_FLAGS= -o $@ -L$(BASE)/lib -Xlinker $(BASE)/lib
$(BASE)/lib/libdb2.so
#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_DBNAME) \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_DBNAME) \n prep tpcdUF.sqc
BINDFILE PACKAGE ISOLATION RS 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
```

ploaduf1

```
#!/bin/ksh
RFpair=$1
/home/db2inst1/tpcd/tools/load_line_uf $RFpair &
/home/db2inst1/tpcd/tools/load_orders_uf $RFpair
```

Ploaduf2

```
#!/bin/ksh
RFpair=$1;
db2 connect to tpcd
db2 "load from delete.new.$RFpair of del modified by coldel| fastparse  messages
/home/db2inst1/tpcd/temp/deleteUF.msg replace into
TPCDTEMP.ORDERS_DEL nonrecoverable data buffer 256 partitioned db
config mode load_only part_file_location /300GB_4mln_UF_flatfiles;"
db2 commit;
db2 connect reset
db2 terminate
```

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"};
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 startLog.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  >> $runDir${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 runthroughtput tests
 system("echo \"The following disks are assigned to the indicated volume
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)
     system("perl getstats p &");
   }
   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 $auditDir${delim}auditruns${delim}tpcdbatch -d
$dbname -f $runDir${delim}qtextpow.sql -r on -b on -s $sf -u p1 -m $inlistmax -n
0 -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");
}
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";
}

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 ( $mode eq "mpp")
{
      $prefix ="rah \";";
      $a ="\\";
}
else {
       $prefix = "";
         $a = "";
}


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")
      {
            $rc= `$prefix perl5 $auditDir${delim}tools${delim}zap $a"-f$a"
$a"^sar$a" `;
```

```
      $rc= `$prefix perl5 $auditDir${delim}tools${delim}zap $a"-f$a"
$a"^sadc$a" `;
      }
     else
      {
      $rc= `$prefix perl5 $auditDir${delim}tools${delim}zap $a"-f$a"
$a"^vmstat$a" `;
      $rc= `$prefix perl5 $auditDir${delim}tools${delim}zap $a"-f$a"
$a"^iostat$a" `;

      $rc= `$prefix iostat$suffix `;

      }
     $rc= `$prefix perl5 $auditDir${delim}tools${delim}zap $a"-f$a"
$a"^getstats$a" `;

   }
}

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  >>
$runDir${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 >
$runDir${delim}dbrun$runNum.snap.$curTs");
   system("db2 get snapshot for database manager >>
$runDir${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 (mpinterX.metrics)
#cd $TPCD_RUN_DIR
&cat("$runDir${delim}mpqinter*","$runDir${delim}mpinter$runNum.metrics");

if ($runUF eq "UF") {
&cat("$runDir${delim}mpufinter*","$runDir${delim}mpinter$runNum.metrics")
;
}

#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 >> $dbtunefile");
  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 configuration taken at
: $timestamp";
     close(OSTUNE);
     system("${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");
}
```

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 differences.
# 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];
  }
}


@reqVars     = ("TPCD_AUDIT_DIR",
              "TPCD_RUN_DIR",
              "TPCD_DBNAME",
              "TPCD_RUNNUMBER",
              "TPCD_SF",
              "TPCD_PLATFORM",
              "TPCD_PATH_DELIM",
              "TPCD_PRODUCT",
              "TPCD_AUDIT",
              "TPCD_PHYS_NODE",
              "TPCD_MODE",
              "TPCD_ROOTPRIV",
              "TPCD_NUMSTREAM");


&setVar(@reqVars, "ERROR");

if (length($ENV{"TPCD_LOG_DIR"}) <= 0)
{
    $ENV{"TPCD_LOG_DIR"} = "NULL";
}

#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"};
$platform=$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" || $platform 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 manager 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 during
# 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" || $platform eq
"hp" || $platform eq "linux")
{
   system("$streamExNT perl watchstreams $streamEx");
}
else
{
   die "platform not supported, can't start watchstreams in background";
}

# 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";
   }
}


  # 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; $loopStream++)
  {
    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 $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");
  }
  else {
    $ret=system("$auditDir${delim}auditruns${delim}tpcdbatch -d $dbname -f
$runDir${delim}quft.sql -r on -b on -s $sf -u t2 -m $inlistmax -n $numStream");
  }
  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");
  }

#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 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");
}

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 >
$runDir${delim}dbTrun$runNum.snap.$curTs");
   system("db2 get snapshot for database manager >>
$runDir${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.metrics");

if ($runUF ne "no") {

&cat("$runDir${delim}mtufinter*","$runDir${delim}mtinter$runNum.metrics");
}

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 >>
$runDir${delim}endLog.Info",$all_ln);
if ( $logDir ne "NULL" )
{
   $rc=&dodb_noconn("$dircmd $logDir >>
$runDir${delim}endLog.Info",$all_ln);
}

#system("db2_all \']}db2 get db cfg for tpcd | grep -i log >>
$runDir${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 endLog.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 >> $dbtunefile");
 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 configuration taken at
: $timestamp";
    close(OSTUNE);
    system("${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");
}
```

tpcdbatch.h

```
/*****************************************************************
*************
*
*   TPCDBATCH.H
*
```

```
*  Revision History:
*
*  27 may 99 bbe from (24 nov 98 jen) fixNTtimestamp - fixed NT timestamp to
print millisecond correctly
*  27 may 99 bbe from (10 dec 98 jen) SUN - added Haider's changes necessary
for SUN
*  17 jun 99 jen  Increased version to 5.1
*  10 aug 99 bbe  Increased version to 5.2
*  13 aug 99 bbe  Increased version to 5.3
*  18 mar 02 ken  Increased version to 5.7
*****************************************************************
**************/

/** 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) || defined(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 "5.7"

#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) || defined (SQLHP)
#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) ||
defined(SQLDOS)
typedef struct timeb Timer_struct;
#elif (defined (SQLUNIX) || defined(SQLAIX) || defined(SQLHP))
/*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   */
```

tpcdbatch.sqc

```
/*****************************************************************
************
*
*   TPCDBATCH.SQC
*
*  Revision History:
*
*  21 Dec 95 jen  Corrected calculation of geometric mean to include in the
*                 count of statements the update functions.
*  03 Jan 96 jen  Corrected calculation of arithmetic mean to not include the
*                 timings for the update functions. (only want query timings
*                 as part of arithmetic mean)
*  15 Jan 96 jen  Added extra timestamps to the update functions.
*  22 Jan 96 jen  Get rid of checking of short_time....we always use the long
*                 timings.
*                 Fixed timings to print query/uf times rounded up to 0.1 seconds
*                 and uses these rounded time values in subsequent calculations
*                 Fixed bug where last seed in mseedme file wasn't getting read
*                 correctly - EOF processing done too soon.
*
*  22 Feb 96 kbs  port to NT
*  26 Mar 96 kbs  Fix to avoid countig UFs as queries for min max
*  27 Jun 97 wlc Temporarily fixed deadlock problems when doing UF1, UF2
*  30 Jul 97 wlc Add in support for load_update and TPCD_SPLIT_DELETES
*  13 Aug 97 wlc fixed UF1 log file formatting problem,
*                using TPCD_TMP_DIR for temp files instead of /tmp,
*                make summary table fit in 80-column,
*                fixed UF2 # of deleted rows reporting problem
*  18 Aug 97 wlc added command line support for inlistmax
*  20 Aug 97 wlc added support for runthroughput without UF
*  27 Aug 97 aph Replaced hardcoded 'tpcdaudit' with
getenv("TPCD_AUDIT_DIR")
*  05 Sep 97 wlc fixing free() problem in NT
*  26 Sep 97 kmw change FLOAT processing in echo_sqlda and print_headings
*  10 oct 97 jen  add lock table in share mode for staging tables
*  21 oct 97 jen added explicit rollback on failure of uf1
*  27 oct 97 jen don't update TPCD.xxxx.update.pair.num if not running UFs in
*                throughput run
*  01 nov 97 jen temp code to do a prep then execute stmt in UFs so we can
*                get timings
*  03 nov 97 jen realligned UF code for readablility
*                pushed UF2 commit into loop for inlistmax
*                fixed UF2 code so rollback performed
*  04 nov 97 jen Added code to handle vldb
*  06 nov 97 jen Commented out temp code for prep then execute stmts using
*                TPCD_PREPARETIME def
*                Updated version number to 2.2
*                send all output during update functiosn to output files, not
*                stderr
*  10 nov 97 jen jenCI Updated version number to 2.3
*                Added handling of TPCD_CONCURRENT_INSERTS. Change
control of
*                chunk processing to use the concurrent_inserts value as the
*                control.  Now the inserts will be run in
TPCD_CONCURRENT_INSERTS
*                sets, each having concurrent_inserts/
*  13 nov 97 jen jen DEADLOCK.  FIxed bug that Alex found where deadlock
count
*                (maxwait) was incremented on every execution of the stmt as
*                opposed to just when deadlock really happened.
*  14 nov 97 jen jenSEM - fix up error reporting on semaphore failure
*                sem_op now returns failure to caller so caller can report where
*                failure has happened.
*                Forced dbname to be upper case, an all other parts of update
*                pair number to be lowercase
*  15 nov 97 jen SEED Reworked code to grab the seed from the seed file.  Now
*                reusing seeds between runs, so power run will always use first
*                seed, throughput will use the 2nd - #stream+1 seeds
*
*  13 jan 98 jen LONG  Increase stmt_str to be able to hold inlists with larger
*                order key numbers
*  04 mar 98 jen IMPORT added support for TPCD_UPDATE_IMPORT to chose
whether
*                using import or load api's for loading data into the staging
*                tables
*  04 mar 98 jen TIMER changed from using gettimer to gettimeofday for unix
*  01 apr 98 jen Fixed IMPORT code to do the proper checking on strcmp (ie
!strcmp)
*  01 apr 98 jen removed code to handle vldb - not needed
*                Upgraded version to 2.4        for ( chunk
*  01 apr 98 jen Fixed up import code on NT so the variable is recognized in the
*                children
*  25 may 98 sks Reworked some of the environment variable code so consolidate
as
*                much as possible.  Not all complete because of differences in
*                the way nt and AIX calls (and starts stuff in background) for UFs
*  29 may 98 jen REUSE_STAGE Changed UF1 so we reuse the same staging
tables
*                instead of having a new set for each update pair
*  06 jul 98 jen Removed locking of staging tables since they are created with
*                locksize table now
*  06 jul 98 jen 912RETRY - added code to retry query execution on 912 as well
*                as 911
*  07 jul 98 jen Fixed summary_table() so 1000x adjustment not based on UF
(setting
*                of max and min pointers
*                Added generic SleepSome function to handle NT vs AIX sleep
differences
*  01 apr 98 djd Added change to permit the use of table functions for UF1.
*                to enable this set TPCD_UPDATE_IMPORT to tf in TPCD.SETUP
file.
*                MERGED this into base copy on Jul 07
*  10 jul 98 jen haider's fix for 'outstream' var for error processing in
*                runUF1_fn and runUF2_fn
*                Updated version to 2.5
*  25 sep 98 jen Added stream number printing into mpqry* files and increases
*                accuracy of timestamp in mpqry (and mts*qry*) files
*  06 oct 98 jen TIME_ACC Added accuracy of timestamp in mpqry (and
mts*qry*)
*                files. Cleaned up misuse of Sleep and flushed buffers on
*                deadlocks
*  19 oct 98 kbs fix UF2_fn to correctly count rows deleted in case of deadlock
*  20 oct 98 kbs rewrite UF2 and UF2_fn for static SQL with staging table
```

```
*  23 oct 98 jen Cleaned up retrying of order/lineitem on lineitem deadlock in
UF1
*  24 oct 98 jen Used load_uf1 and load_uf2 instead of general load_updates
*  26 oct 98 kbs inject the UF1 with a single staging table
*  02 nov 98 jen Fixed processing of multiple chunks in uf2 so don't duplicate
*  21 nov 98 kmw Fixed BIGINT
*  05 dec 98 aph Moved runUF1_fn() and runUF2_fn() into a separate file
tpcdUF.sqc
*             so that it can be bound separately with a different isolation level.
*  21 dec 98 aph Integrated Jennifer's QppD calculation (rounding & adjustment)
fixes.
*  22 dec 98 aph For UFs during Throughput run, defer CONNECT until children
launched.
*  28 dec 98 aph Removed error_check() call after CONNECT RESET
*  29 dec 98 aph For UFs do not COMMIT in tpcdbatch.sqc.  COMMITs happen
in tpcdUF.sqc.
*  18 jan 99 kal replaced header with #include "tpcdbatch.h"
*  27 may 99 bbeaton from (03 mar 99 jen) Fixed SUN fix that wasn't compatible
with
*             NT (using %D %T instead of %x %X for strftime)
*  16 jun 99 jen Added missing LPCTSTR cast of semaphore file name for NT
*  17 jun 99 jen SEMA Changes semaphore file for update functions to look for
tpcd.setup
*             not for the orders.***  update data file
*  21 jul 99 bbeaton Added semaphore control that allows runpower to be run as
two
*             separate streams (update and query).  This involves the use of
*             two semaphores to be used as it executes in three different
*             sections.  The first is the update inserts.  The next is the query
*             stream which is started with the update stream, but waits until
*             the inserts are complete.  The third section is the update deletes
*             which execute after the queries are complete.
*  21 jul 99 bbeaton Added functions to handle semaphore creation, control, etc.
*  21 jul 99 bbeaton Modified output to mp*inter files.  It now only outputs
*             intermediate data that will be calculated by calcmetricp.pl.  This
*             is a result of the runpower being split into two streams and thus
*             tpcdbatch not having access to all data.
*  21 jul 99 bbeaton The start time for runpower UF2 now does not start until after
*             the query stream is complete so that its wait time is not included
*             NOTE: The wait time that the first UF1 in runthroughput still
*             includes the wait period that occurs waiting on queries.
*  18 mar 02 kentond removed the need for list files. Instead of using the *.list
*             files to determine the name of the output files, the tags for the
*             source sql files are used.
*  07 Jan 04 jregier Added Christian's change to the create_semaphore function,
*             simply checks for the existence of the semaphore first and
*             removes it if it wasn't properly cleaned up previously.
**************************************************************************
*************/

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

#include "tpcdbatch.h"

/**************************************************************************
**********/
/* global structure containing elements passed between different functions */
/**************************************************************************
**********/
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 */
/* kjd715 */
/*   char               str_file_name[256];   */ /* output filename    */
/* kjd715 */
  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 functions?*/
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 (default) */
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[])
{
 /* kjd715 */
 /*struct comm_line_opt c_l_opt = { "\0","\0", 0, 1, 0, 0, 0 };*/ /* kjd715 */
 struct comm_line_opt c_l_opt = { "\0", 0, 1, 0, 0, 0 };
 /* kjd715 */
 /* 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 specific 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 */
```

```
  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 statement */
  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 number: %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 number: %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 going */
    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 terminated) */
  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 converted.\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).  Aborting.\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 results...\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 exists */

      /*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         -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 details");
  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 output)");
  printf("\n\n--#TAG      tag           (user specified tag name for sequence#)");
  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;
       /* kjd715 */
       case 'l' :
       case 'L' :    loopvar+=1;
                                       /*

strcpy(g_struct->c_l_opt->str_file_name,argv[++loopvar]);
                                       */
                break;
              /* kjd715 */
       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 updates*/
         if (!strcmp(uppercase(argv[loopvar]),"P"))
           g_struct->c_l_opt->update=4; /* power update without updates*/
         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 command ");
        fprintf(stderr, "line\nnor in environment variable DB2DBDFT.");
        display_usage();
      }
      else
        strcpy(dbname,testptr);

}
/* kjd715 */
                /*
    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();
    }
                */
/* kjd715 */
}
```

```
/*************************************************/
/* 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 ) {
            /* kjd715 */
      fprintf(outstream, "XXThe input file could not be opened.\n\n");
      /* kjd715 */
      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 variable 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 directory */
    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 correct\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 semaphores, */
  /* 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),
lowercase(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 total */
        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).  Aborting.\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*/

  fprintf (outstream, "\n\n\nSummary of Results\n==================\n");
  fprintf (outstream,
          "\nSequence #     Elapsed Time   Adjusted Time Start Timestamp   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 properly
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 seconds, */
    /* 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 +
((double)g_struct->stream_start_time.tv_usec/1000000);
    Ts2 = (double)g_struct->stream_end_time.tv_sec +
((double)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 +
((double)g_struct->stream_start_time.millitm/1000);
    Ts2 = (double)g_struct->stream_end_time.time +
((double)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 database 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;

  if (select_status == TPCDBATCH_SELECT)
    for (loopvar=0; loopvar<sqlda->sqld; loopvar++) {
      free(sqlda->sqlvar[loopvar].sqldata);
      free(sqlda->sqlvar[loopvar].sqlind);
    }

  free(sqlda);
  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, starting\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, updatePair);
#else
  sprintf (statement, "perl %s/tools/ploaduf1 %d 1", sourcedir, updatePair);
#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. Exiting.\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);
  }
  /*semctl(su_semid, 0, IPC_RMID, 0);*/ /*mujib*/
#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, childpid[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", GetLastError());
    /* 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 DELjen */
  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, starting\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 concurrency */

  /* 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, updatePair);
#else
  sprintf (statement, "perl %s/tools/ploaduf2 %d 2", sourcedir, updatePair);
#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 processes */
#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);
  }
  /*semctl(su_semid, 0, IPC_RMID, 0);*/ /*mujib*/
#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, childpid[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 children\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", GetLastError());
    /* 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=0, found=0;                          /* kjd715 */
  char input_ln[256]  = "\0";       /* kjd715 */
  char tag[128]       = "\0";           /* kjd715 */

  strcpy(run_dir,g_struct->run_dir);
  sprintf(delim,"%s",env_tpcd_path_delim);
  strcpy(time_stamp,g_struct->file_time_stamp);
  /* kjd715 */
  if (g_struct->stream_list == NULL)
  {
      if((g_struct->stream_list =
                fopen(g_struct->c_l_opt->infile, READMODE)) == NULL)
            {
      fprintf(stderr,"\nThe input file could not be opened.");
      fprintf(stderr,"Make sure that the filename is correct.\n");
      exit(-1);
    }
  }
  found = 0;
  do {
    fscanf(g_struct->stream_list, "\n%[^\n]\n", input_ln);
    if (strstr(input_ln, "--#TAG") == input_ln)
    {
              found = 1;
              strcpy(tag,(input_ln+sizeof("--#TAG")));
              if(strncmp(tag, "UF", 2) == 0)
            qnum = atoi(tag+2)*(-1);
              else if(strncmp(tag, "Q", 1) == 0 )
              {
                        /* for query 15a the 'a' must be trimmed */
                        /* off before converting to integer      */
                        if(strlen(tag)>3)
                    tag[3] = '\0';
```

```
                        qnum = atoi(tag+1);
                }
    }

    if (feof(g_struct->stream_list))
      found = 1;

    }while (!found);
    /*
      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);
                */
   /* kjd715 */

    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),
currentUpdatePair,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 completed */
      /* 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 statements 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 ) && (sqlcode !=-1415 ))
      {
        max_wait ++;
        fprintf (stderr, "\nAn error has been detected on execute
immediate...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;

  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.  Disconnects
  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 correct\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);

  /* connect reset used to only be done in the semaphore control
    block below, to the best of my knowledge that was incorrect.
    jregier 03/09/30
  */
  EXEC SQL CONNECT RESET;
```

```
  /* 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); /* throughput stream
*/

#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, GetLastError: %d,
quitting\n",
                    GetLastError());
              exit(-1);
            }
            fprintf(stderr,"Semaphores created successfully!\n");
      }
      else
      {
      /* RUNTHROUGHPUT creates semaphores based on the number of query
streams while the number of streams for runpower is constant */
            hSem = CreateSemaphore(NULL, 0,
                        g_struct->c_l_opt->intStreamNum,
                        (LPCTSTR)(g_struct->sem_file));

            if (hSem == NULL)
            {
                fprintf(stderr,
                    "CreateSemaphore (ready semaphore) failed, GetLastError: %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 = semget(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);
            }
            /*semctl(semid,0,IPC_RMID,0);*/ /* mujib */
      }
      else
      /* THROUGHPUT */
      {

                  /* TRY TO CREATE IT USING EXCL MODE */
/* cmgarcia */
                  while ( (semid =
semget(semkey,1,IPC_CREAT|IPC_EXCL|S_IRUSR|S_IWUSR)) < 0)
                  {
                        if (errno == EEXIST)
                        {
                              /* IT ALREADY EXISTS */
                              if (verbose)
                              {
                                    fprintf(stderr,
                                          "Throughput
can't get initial semaphore! semget failed errno = EEXIST...retrying\n");
                              }
                        errno = 0;

                              /* GET THE SEMAPHORE THAT
ALREADY EXISTS */
                              if ( (semid =
semget(semkey,1,S_IRUSR|S_IWUSR)) < 0)
                              {
                                    fprintf(stderr,
                                          "Throughput
can't get (no create) initial semaphore! semget failed errno = %d\n",
                                          errno);
```

```
                                        exit(1);
                                }

                                /* REMOVE THE SEMAPHORE */
                                if (semctl (semid, 1, IPC_RMID) <
0)
                                {
                                        fprintf(stderr,
                                                "Throughput
can't remove initial semaphore! semget failed errno = %d\n",
                                                errno);
                                        exit(1);
                                }
                        }
                        else
                        {
                                fprintf(stderr,
                                        "Throughput can't get
initial semaphore! semget failed errno = %d\n",
                                        errno);
                                exit(1);
                        }
                } /* IF WE COULDN'T TRY AGAIN */


        /* jlr
        if ( (semid = semget(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);
        }
        */


        /* semctl(semid,0,IPC_RMID,0);*/ /* mujib */

        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", GetLastError());
      /* 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 semaphore 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", GetLastError());
            /* 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 semaphore\n");
    else
     fprintf(stderr,"updatestream finished waiting on querystream semaphore\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 LastError: %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 semaphore 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
```

tpcdUF.sqc

```
/*************************************************************************
************
*
```

```
*  TPCDUF.SQC
*
*  Revision History:
*
*  05 dec 98 aph Created tpcdUF.sqc containing runUF1_fn() and runUF2_fn()
*                so that it can be bound separately with a different isolation level.
*  15 may 99 bbe Added cast (short) for type conversion between a long and a
short.
*  16 jun 99 jen Added in proper connect reset code for UF functions (mistakenly
*                removed
*  17 jun 99 jen SEMA Changes semaphore file for update functions to look for
tpcd.setup
*                not for the orders.***  update data file  (AIX  only )
*  21 jul 99 bbe Commented out conditions in SQL statments that searched on
fields
*                other than app_id.
*
*****************************************************************
*************/
#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 functions?*/
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_CO
MMENT
           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_QUANTITY,
          L_EXTENDEDPRICE,L_DISCOUNT,L_TAX,

L_RETURNFLAG,L_LINESTATUS,L_SHIPDATE,L_COMMITDATE,L_REC
EIPTDATE,
          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",
            insertedOrders,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 numChunks, 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: 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)
      {
        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",
                    deletedOrders,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",
                    deletedLineitems,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\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",
maxwait,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      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, "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",
thisConcurrentDelete);
    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 */
}
```

Appendix E: ACID Transaction Source Code

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
```

acid.sqc

```
/*************************************************************************
*******/
/*      File: acid.sqc                                                   */
/*************************************************************************
*******/

/*   changes:
 *
 *  961109  jel    add EXEC SQL CLOSE for each cursor in acidT
 *                 to avoid bug in db2pe v1r2
 *  980225  gav    port to NT
 *  981103  kal    added ast_acidQ  for isolation test 7
 *  981103  kal    changed ast query to be the same as that used in
 *                 consistency tests.  Fixed so the long lEprice is
 *                 cast to a double.  Changed so uses 3 decimal points of
 *                 precision.
 *
 */

#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) ;
   }

/*
** why is this code in here? we don't want to
** commit until the history table has been updated as well
  if (acid->termination == 0) {
     EXEC SQL CLOSE L_CURSOR;
     EXEC SQL CLOSE O_CURSOR;
```

```
    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);
      }
      sqlerror("acidT: COMMIT", &sqlca);
      goto Terror;
    }
  }
*/

  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 */
```

makefile

```
DBNAME =          $(TPCD_QUAL_DBNAME)

INCLUDE =         $(HOME)/sqllib/include

#CFLAGS =         -I$(INCLUDE) -g -Dpascal= -DLINT_ARGS \
#           -Dfar= -D_loadds= -DSQLA_NOLINES -qflag=i:i -qlanglvl=ansi
CFLAGS  =         -m64 -I$(INCLUDE) -g -Dpascal= -DLINT_ARGS  \
          -DSQLA_NOLINES -DLinux

#LFLAGS  =        -lm -lcurses -ls -ll -ly -liconv -lbsd
#CFLAGS =         -I$(INCLUDE) -g -Dpascal= -DLINT_ARGS  \
#           -DSQLA_NOLINES -qflag=i:i -qlanglvl=ansi
# .. sun       -DSQLA_NOLINES

LFLAGS  =         -m64 -lm
#LFLAGS  =        -lm
# sun .... LFLAGS  =  -lm


LIB       =       -L$(HOME)/sqllib/lib -ldb2

CC        =       cc

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: Price Quotations

Oct. 4, 2006

IBM Corporation
Ms. Celia Schreiber
System x Performance

Dear Celia:

The table shown below lists the U.S. pricing for DB2 Universal Database Enterprise Server Edition product that has been used in the TPC-H Benchmark.

All prices shown are in U.S. Dollars.

DB2 Enterprise Server Edition (ESE)	**Qty**	**Reference Price per unit**	**Total Reference price**
SW License & 12 Months Maintenance	2	23,902	47,804
SW Maintenance Renewal - 1 year	4	1,138	4,552
DPF SW License & Maintenance 12 Months	2	7,180	14,360
DPF SW Maintenance Renewal – 1 Year	4	342	1,368
		Sub-total reference price for DB2 ESE:	**68,084**
		TOTAL REFERENCE PRICE:	**68,084**

Any and all prices herein are suggested prices only and are subject to change at IBM's sole discretion. Products listed herein are subject to withdrawal or modification by IBM at any time at IBM's sole discretion.

Sincerely,

Richard Hughes
IBM Sales & Distribution, Software Sales
Americas Sales Executive DB2 and Informix
212-493-2065
rhughes@us.ibm.com

Novell Quote

Novell.

Quote Information

Date: 06−Oct−2006

Quote ID: 11311

Subject: chris king 4

Contract: VLA

Harold Moore
1177 Ave of Americas 35th Floor
New York, New York 10036 USA
Phone: 212−403−7837 Fax: 801−861−1056
Email: Hmoore@novell.com

Client Information

IBM

7708 Audubon Dr.

Raleigh, NC 27615−3403 USA

Chris King
Phone: (919) 543− 8799 Fax: 919− 486− 2327
Email: kchris@us.ibm.com

Products

Item Description	SKU	Retail Price	Qty	Extended Price
SUSE Linux Enterprise Server 9 2−CPU with 24X7 Priority Support and 3036 Training kit Bundle 3−Year Upgrade Protection	874−004926	4,279.00	1	4,279.00
			Number of Items: 1	***Total: USD 4,279.00***

Sales Rep Comments

This Quote is valid for 30 days from 06−Oct−2006. If a pricing discrepancy should arise, the pricing as published in the current Novell Product Price List and/or NPA will prevail. This quote does not include state and/or local sales tax, if any. The prices in this quote are suggested only. To get final VLA pricing, check with your VLA Approved Partner.