Hewlett-Packard Company

TPC Benchmark™ H
Full Disclosure Report

HP ProLiant DL585 2.2GHz 4P DC
using
IBM DB2 UDB 8.2 and
Red Hat Enterprise Linux 4 AS

First Edition
August 2005

First Edition – August 2005

Hewlett Packard Company, the Sponsor of this benchmark test, believes that the information in this document is accurate as of the publication date. The information in this document is subject to change without notice. The Sponsor assumes no responsibility for any errors that August appear in this document.

The pricing information in this document is believed to accurately reflect the current prices as of the publication date. However, the Sponsor provides no warranty of the pricing information in this document.

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

All performance data contained in this report was obtained in a rigorously controlled environment. Results obtained in other operating environments August vary significantly. No warranty of system performance or price/performance is expressed or implied in this report.

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark™ H test conducted on the HP ProLiant DL585 using IBM DB2 UDB 8.2, in conformance with the requirements of the TPC Benchmark™ H Standard Specification, Revision 2.3.0. The operating system used for the benchmark was Red Hat Enterprise Linux 4 AS.

The benchmark results are summarized in the following table.

Hardware	Software	Total System Cost	QppH @ 300GB	QthH @ 300GB	QphH @ 300GB	$ / QphH @ 300GB
HP ProLiant DL585	**IBM DB2 UDB 8.2** **Red Hat Enterprise Linux 4 AS**	**$288,751**	**15504.0**	**9157.2**	**11915.3**	**$24.24 USD**

The TPC Benchmark™ H was developed by the Transaction Processing Performance Council (TPC). The TPC was founded to define transaction processing benchmarks and to disseminate objective, verifiable performance data to the industry.

Copies of this full disclosure report can be obtained from the Transaction Processing Performance Council at www.tpc.org

Standard and Executive Summary Statements

Pages vi-x contains the Executive Summary and Numerical Quantities Summary of the benchmark results for the HP ProLiant DL585.

Auditor

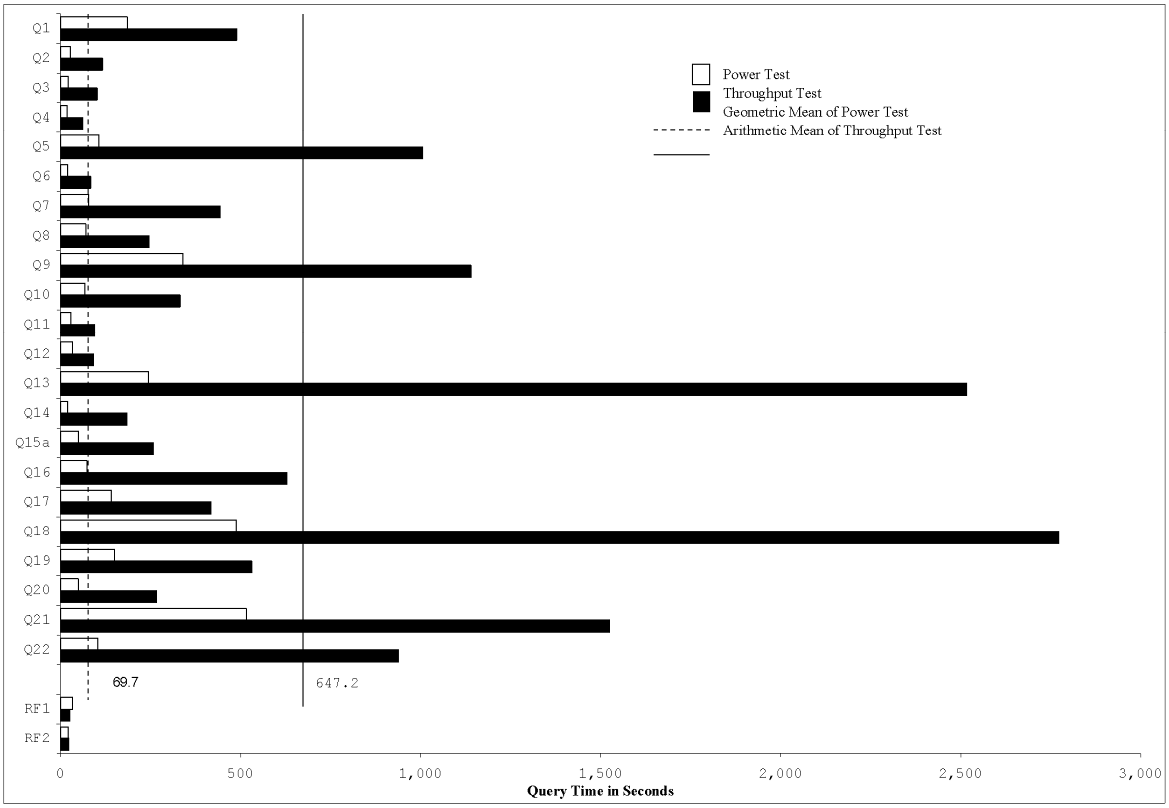
The benchmark configuration, environment and methodology used to produce and validate the test results, and the pricing model used to calculate the cost per QppH and QthH were audited by Lorna Livingtree of Performance Metrics, Inc.. to verify compliance with the relevant TPC specifications. The auditor's letter of attestation is attached in Section 9.1 "Auditors' Report."

Table Of Contents

invent	**HP ProLiant DL585 2.2GHz 4P DC**	TPC-H Rev. 2.3.0 Report Date: **August 31, 2005**

Total System Cost	Composite Query per Hour Metric	Price / Performance
$288,751	**11915.3** **QphH @ 300GB**	**$24.24 USD** **per QphH @ 300GB**

Database Size	Database Manager	Operating System	Other Software	Availability Date
300GB	**IBM DB2 UDB 8.2**	**Red Hat Enterprise Linux 4 AS**	**none**	**Oct 5, 2005**



Database Load Time = 6:16:45	Load Included Backup: Y	Total Data Storage / Database Size = 19.7

RAID (Base tables only): N	RAID (Base tables and auxiliary data structures): N	RAID (All): N

System Configuration :

Processors :	4 x 2.2GHz 1MB L2 cache dual core AMD Opteron 875 Processors
Cores :	8 total
Threads :	8 total
Memory :	64 GB memory
Disk Controllers :	8 x HP Smart Array P600 SAS Controller
Disks :	160 x 36.4GB SAS 10K SFF drives (external)
	2 x 36.4GB 15K U320 drives (internal)
Total Disk Storage:	5896.8 GB

Database Size includes only raw data (e.g., no temp, index, redundant storage space, etc.).

invent	HP ProLiant DL585 2.2GHz 4P DC	TPC-H Rev. 2.3.0 Report Date: 31-Aug-05

Description	Part Number	Third Party		Unit Price	Qty	Extended Price	3 yr. Maint. Price
Server Hardware		Brand	Pricing				
ProLiant DL585 R01 O2.2GHz (1 MB) x 2	383357-001		1	15,999	1	15,999	
- 2 GB PC2700 DDR, Integrated Smart Array Controller 5i,							
- Embedded NC7782 Dual Port PCI-X 10/100/1000T Gigabit NIC							
DL585 875 O2.2GHz/PC2700 8 socket processor option kit	381476-B21		1	4,049	2	8,098	
4 GB PC2700 DDR SDRAM DIMM 2x2048 WW	371049-B21		1	2,269	16	36,304	
HP s7540 17in CRT Monitor	PF997AA#ABA		1	149	1	149	
HP PS/2 Scroll Mouse Carbonite	DG169AV		1	5	1	5	
PS/2 Standard Keyboard	DG170AV#ABA		1	10	1	10	
HP 5642 Unassembled Rack	358254-B21		1	689	1	689	
3YR 24X7 4HR 500 SERIES SVR	U4608E		1	1,575	1		1,575
					Subtotal	**61,254**	**1,575**
Storage							
HP SMART Array P600 3G SAS/SATA RAID Controller	337972-B21		1	999	8	7,992	
HP StorageWorks MSA50 Disk Enclosure	364430-B21		1	1,899	16	30,384	
HP 36GB 15K U320 Pluggable Hard Drive (internal)	286776-B22		1	299	2	598	
HP 36GB 10K SAS Single Port SFF Hard Drive	375859-B21		1	329	160	52,640	
MSA50 Carepaq 3yr 4hr 24x7	U8130E		1	1,827	16		29,232
					Subtotal	**91,614**	**29,232**
Hardware and Maintence Discount							
Large Purchase and Net 30 discount	16.0%		1			($24,459)	($4,929)
					Hardware Subtotal	**128,409**	**25,878**
Software							
DB2 UDB Enterprise Server Edition (ESE) License incl 1-yr Maint		IBM	2	22,608	4	90,432	
DB2 UDB Enterprise Server Edition (ESE) Support - 1-yr/proc		IBM	2	1,077	8		8,616
DB2 UDB Data Partitioning Feature (DPF) License incl 1-yr Maint		IBM	2	6,791	4	27,164	
DB2 UDB Data Partitioning Feature (DPF) Support - 1-yr/proc		IBM	2	323	8		2,584
HP Red Hat Ent Linux 4 AS w/PRM 24x7 3yr sw tech sprt umlim.	392381-B26		1	6,747	1	6,747	included
					Subtotal	**124,343**	**11,200**
HP Software and Maintence Discount							
Large Purchase and Net 30 discount	16.0%		1			($1,080)	
					Software Subtotal	**123,263**	**11,200**
					Total	**$251,673**	**$37,078**

Three-Year Cost of Ownership:	**$288,751**
QphH @ 300GB:	**11915.3**
$ / QphH @ 300GB:	**$24.24 USD**

Pricing: 1=HP Direct: 800-203-6748; 2=IBM

Note:The benchmark results and test methodology were audited by Lorna Livingtree of Performance Metrics, Inc. (www.perfmetrics.com).

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 the stated prices are not available according to these terms, please inform at pricing@tpc.org. Thank you.

invent	HP ProLiant DL585 2.2GHz 4P DC	TPC-H Rev. 2.3.0 Report Date: **August 31, 2005**

Numerical Quantities

Measurement Results:

Database Scale Factor	= 300
Total Data Storage / Database Size	= 19.7
Start of Database Load	= 8/22/2005 14:46:18
End of Database Load	= 8/22/2005 21:03:03
Database Load Time	= 6:16:45
Query Streams for Throughput Test	= 6
TPC-H Power	= 15504.0
TPC-H Throughput	= 9157.2
TPC-H Composite Query-per-Hour Metric (QphH@300GB)	= 11915.3
Total System Price Over 5 Years	= $288,751 USD
TPC-H Price/ Performance Metric ($/QphH@300GB)	= $24.24 USD

Measurement Intervals:

Measurement Interval in Throughput Test (Ts)	= 15568 seconds

Duration of Stream Execution:

	Seed	Query Start Date/Time Query End Date/Time		RF1 Start Date/Time RF1 End Date/Time		RF2 Start Date/Time RF2 End Date/Time		Duration
Stream 00	822210303	08/22/05	23:01:06	08/22/05	23:00:33	08/22/05	23:48:14	0:47:07
		08/22/05	23:48:14	08/22/05	23:01:06	08/22/05	23:48:35	
Stream 01	822210304	08/22/05	23:48:38	08/23/05	3:58:37	08/23/05	4:03:43	4:09:59
		08/23/05	3:58:37	08/23/05	4:03:43	08/23/05	4:04:07	
Stream 02	822210305	08/22/05	23:48:38	08/23/05	3:52:54	08/23/05	4:04:32	4:04:17
		08/23/05	3:52:54	08/23/05	4:04:32	08/23/05	4:04:53	
Stream 03	822210306	08/22/05	23:48:38	08/23/05	3:34:32	08/23/05	4:05:18	3:45:55
		08/23/05	3:34:32	08/23/05	4:05:18	08/23/05	4:05:44	
Stream 04	822210307	08/22/05	23:48:38	08/23/05	3:31:47	08/23/05	4:06:10	3:43:09
		08/23/05	3:31:47	08/23/05	4:06:10	08/23/05	4:06:34	
Stream 05	822210308	08/22/05	23:48:38	08/23/05	4:03:13	08/23/05	4:06:59	4:14:35
		08/23/05	4:03:13	08/23/05	4:06:59	08/23/05	4:07:20	
Stream 06	822210309	08/22/05	23:48:38	08/23/05	3:34:28	08/23/05	4:07:46	3:45:50
		08/23/05	3:34:28	08/23/05	4:07:46	08/23/05	4:08:06	

TPC-H Timing Intervals (in seconds)

Query	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Stream 00	185.0	27.4	20.6	18.8	106.4	19.9	77.7	70.2
Stream 01	245.0	119.1	100.1	23.4	700.0	46.4	562.3	309.5
Stream 02	639.2	171.0	31.5	65.9	1322.6	130.9	464.6	164.4
Stream 03	446.0	62.2	70.9	59.4	1243.0	90.0	550.3	262.8
Stream 04	533.8	106.9	153.2	56.9	1178.3	64.0	348.6	248.1
Stream 05	599.6	121.8	85.0	67.5	770.4	95.5	458.7	261.5
Stream 06	468.0	114.7	167.7	93.9	815.9	72.5	273.9	227.3
minimum	245.0	62.2	31.5	23.4	700.0	46.4	273.9	164.4
average	488.6	116.0	101.4	61.2	1005.0	83.2	443.1	245.6
maximum	639.2	171.0	167.7	93.9	1322.6	130.9	562.3	309.5
Query	**Q9**	**Q10**	**Q11**	**Q12**	**Q13**	**Q14**	**Q15a**	**Q16**
Stream 00	339.2	67.3	29.2	32.8	244.2	19.9	50.0	72.8
Stream 01	655.9	559.1	84.9	97.1	2818.7	228.0	331.1	683.2
Stream 02	1859.4	411.7	43.6	96.3	2163.7	157.7	169.5	694.5
Stream 03	1353.9	194.4	90.2	119.2	1624.5	204.7	220.0	270.1
Stream 04	1258.7	309.0	86.7	96.8	2632.5	215.7	334.7	314.6
Stream 05	328.3	177.5	69.3	54.6	3484.6	192.4	301.1	910.2
Stream 06	1382.3	337.3	189.0	83.5	2376.4	105.7	186.7	896.3
minimum	328.3	177.5	43.6	54.6	1624.5	105.7	169.5	270.1
average	1139.8	331.5	94.0	91.3	2516.7	184.0	257.2	628.2
maximum	1859.4	559.1	189.0	119.2	3484.6	228.0	334.7	910.2
Query	**Q17**	**Q18**	**Q19**	**Q20**	**Q21**	**Q22**	**RF1**	**RF2**
Stream 00	141.2	488.2	149.1	48.7	515.4	103.8	33.6	21.6
Stream 01	423.2	4162.5	691.8	107.8	1568.5	480.6	29.5	24.6
Stream 02	504.1	2923.8	682.2	172.0	1025.0	763.1	25.1	20.8
Stream 03	562.6	3270.9	341.4	81.9	1335.1	1101.2	24.5	26.0
Stream 04	530.5	1480.0	504.7	187.5	1649.0	1098.8	25.6	24.0
Stream 05	239.0	2896.1	491.1	921.3	1565.2	1183.1	25.0	21.7
Stream 06	244.8	1900.6	471.9	130.7	2008.8	1001.5	25.4	20.4
minimum	239.0	1480.0	341.4	81.9	1025.0	480.6	24.5	20.4
average	417.4	2772.3	530.5	266.9	1525.3	938.1	25.9	22.9
maximum	562.6	4162.5	691.8	921.3	2008.8	1183.1	29.5	26.0

1.0 General Items

1.1 Test Sponsor

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

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

1.2 Parameter Settings

Settings must be provided for all customer-tunable parameters and options which have been changed from the defaults found in actual products, including by 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*

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

Appendix A, “Tunable Parameters,” contains a list of all DB2 parameters, operating system parameters and compiler options. Session initialization parameters can be set during or immediately after establishing the connection to the database within the tpchbatch program documented in Appendix D, “Implementation- Specific Layer and Driver Source Code.” This result uses the default session initialization parameters established during preprocessing/binding of the tpchbatch program. The procedure for preprocessing, binding, compiling and linking the tpchbatch program is documented in Appendix A.71, “Compiler Option for TPCDBATCH Driver.”

1.3 Configuration Items

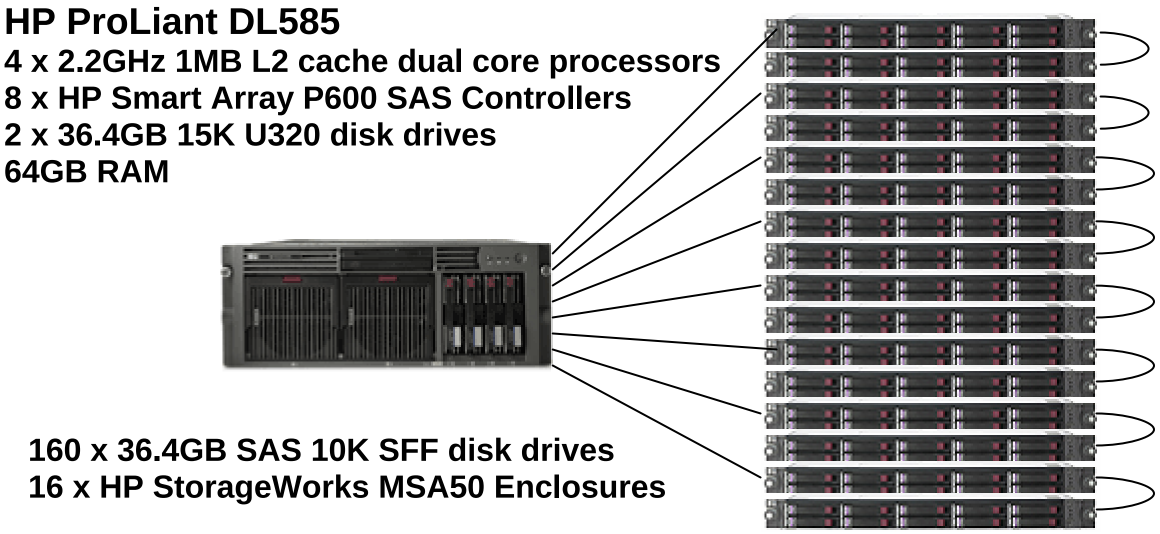
Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:

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

The server System Under Test (SUT), a HP ProLiant DL585, depicted in Figure 1.1, consisted of :
Four AMD Opteron 875 2.2GHz 1MB L2 cache dual core processors

- 64 GB of memory
- 8 x HP Smart Array P600 SAS Controllers
- 16 x HP StorageWorks MSA50 Enclosure
- 160 x 36.4GB SAS 10K SFF Drives

Figure 1.1 Benchmarked and priced configuration


HP ProLiant DL585
4 x 2.2GHz 1MB L2 cache dual core processors
8 x HP Smart Array P600 SAS Controllers
2 x 36.4GB 15K U320 disk drives
64GB RAM
160 x 36.4GB SAS 10K SFF disk drives
16 x HP StorageWorks MSA50 Enclosures

2.0 Clause 1: Logical Database Design

Appendix B, "Database Build Scripts," contains the programs and input files used to load the test and qualification databases. The test and qualification databases are built in exactly the same way in all respects except for the scale factor; they use the same table definitions, indexes and partitioning methods. Thus, the buildtpcd script documented in Appendix B was used for both the qualification and test databases except that different input files were used.

There are two phases for the loading of the database: the generation of the flat data files and the building of the database from them. The buildtpcd script executes DDL and other command scripts to create the database, load the data into the tables, create indexes, gather statistics, and set the configuration. These DDL and other command scripts are documented in Appendix B.

2.1 Table Definitions

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

Appendix B, "Database Build Scripts," contains the table definitions and the program used to load the database.

2.2 Physical Organization of Database

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

Appendix B, "Database Build Scripts," contains the DDL for the index definitions.

2.3 Horizontal Partitioning

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

Horizontal partitioning was used for all tables except for the nation and region tables, see Appendix B "Database Build Scripts".

2.4 Replication

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

No replication was used.

3.0 Clause 2: Queries and Refresh Functions 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 QGEN version 1.3.0 and DBGEN 1.3.0 were used to generate all the database populations. See Appendix B "Database Build Scripts" for details.

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 1.3.0 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.1, "Qualification Queries and Output," contains the output for each of the queries. The functional query definitions and variants used in this disclosure use the following minor query modification:

- Table names are fully qualified. For example, the "NATION" table is referred to as "TPCD.NATION."
- The standard IBM SQL date syntax is used for the 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 the query substitution parameters used during the performance test must be disclosed in tabular format, along with the seeds used to generate these parameters.

Appendix C.3, "Query Substitution Parameters," contains the query substitution parameters used in the performance tests.

3.6 Isolation Level

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

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

3.7 Refresh Functions

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

The refresh functions are part of the implementation specific layer/driver code included in Appendix D, “Implementation Specific Layer and Driver Source Code.”

4.0 Clause 3: Database System Properties

4.1 Atomicity Requirements

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

All ACID tests were conducted according to specification. The Atomicity, Isolation, Consistency and Durability tests were performed on the HP ProLiant DL585. Appendix E, "ACID Transaction Source Code," contains the source code for the ACID transaction and query.

4.1.1 Atomicity of the Completed Transactions

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

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

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

4.1.2 Atomicity of Aborted Transactions

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

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

1. The 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 Orderkey. 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 steps 2 and 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 – L_DISCOUNT) * (1 + L_TAX)*
For each ORDER and LINEITEM defined by (O_ORDERKEY = L_ORDERKEY)

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

```
SELECT DECIMAL (SUM (DECIMAL (INTEGER (INTEGER (DECIMAL (INTEGER (100 * DECIMAL
(L_EXTENDEDPRICE, 20, 3)), 20, 3) * (1 – L_DISCOUNT)) * (1 + L_TAX)), 20, 3) / 100.0) 20, 3) FROM
TPCD.LINEITEM WHERE L_ORDERKEY = okey

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

4.2.2 Consistency Tests

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

The queries defined in section 4.2.1 "Consistency Condition" were run after the initial database build and prior to executing the ACID transaction. The queries showed that the database was in a consistent state.

After executing 7 streams of 100 ACID transactions, the queries defined in 4.2.1 "Consistency Condition" section were run again. The queries showed that the database was still in a consistent state.

4.3 Isolation Requirements

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

4.3.1 Isolation Test 1 - Read-Write Conflict with Commit

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

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 - Read-Write Conflict with Rollback

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

The following steps were performed to satisfy the test of isolation for a 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 by the ACID transaction.
3. First session: The ACID transaction is released and the Rollback is executed, releasing the record.
4. Second session: With the LINEITEM record now released, the ACID query completes.

4.3.3 Isolation Test 3 - Write-Write Conflict with Commit

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

The following steps were performed to verify isolation of two update 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 First 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.EXTENDEDPRICE + (DELTA * (T1.L_EXTENDEDPRICE) / T1.L_QUANTITY)

4.3.4 Isolation Test 4 - Write-Write Conflict with Rollback

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

The following steps were performed to verify isolation of two update 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 First Session. This transaction is forced to wait while the First Session holds a lock on the LINEITEM record requested by the Second Session.
3. First session: Roll back the ACID transaction T1. With the LINEITEM record now released, the ACID transaction T2 completes.
4. Verify that: T2.L_EXTENDEDPRICE = T1.EXTENDEDPRICE

4.3.5 Isolation Test 5 – Concurrent Read and Write Transactions on Different Tables

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

The following steps were performed to successfully conduct this test:

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. Second 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 completes.
4. T1 is allowed to complete.
5. Verify that the appropriate rows in the ORDERS, LINEITEM and HISTORY tables are changed.

4.3.6 Isolation Test 6 – Update Transactions During Continuous Read-Only Query Stream

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

The following steps were performed to successfully conduct this test:

1. First session: A transaction T1, which executes Q1x with DELTA = 0 is started.
2. Second session: Before T1 completes, an ACID transaction T2 with randomly selected values of O_KEY, L_KEY and DELTA, is started.
3. Third session: Before T1 completes, a transaction T3, which executes Q1 with a randomly selected value of DELTA (not equal to 0), is started.
4. T1 completes.
5. T2 completes.
6. T3 completes.
7. Verify that the appropriate rows in the ORDERS, LINEITEM and HISTORY tables are changed.

4.4 Durability Requirements

The tested system must guarantee durability: the ability to preserve the effects of committed transactions and insure database consistency after recovery from any one of the failures listed in Clause 3.5.2.

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

1. The qualification database was backed up to a disk volume other than one used to store the database.
2. The consistency test was verified.
3. The current count of the total number of records in the HISTORY table was determined.
4. A test to run 7 streams of 200 ACID transactions on each execution was started.
5. One of the data disks on logical node 1 containing the DB2 TPCH database data tables was removed after at least 30 transactions per stream had executed.
6. The 7 streams failed and recorded the number of committed transactions in the success file.
7. The failed disk was replaced with a new disk. The database partitions that existed on the failed disks were deleted and recreated. The system was rebooted..
8. A database restore was issued using the backup files from step 1.
9. The command was issued for the database to perform a roll forward recovery operation.
10. The counts in the success file and HISTORY table were compared and found to match.
11. Consistency test was verified.

4.4.2 Loss of Log and Loss of System Power

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

1. The consistency test was verified.
2. The current count of the total number of records in the HISTORY table was determined giving hist1.
3. A test to run 7 streams of 200 ACID transactions on was started.
4. One of the disks containing the DB2 TPCH database transaction log data on logical node 3 was removed after at least 30 ACID transactions had completed.
5. The database consistency was not affected because of log mirroring, and ACID transactions continued to execute successfully.
6. The system was shut down by removing the power cord(s) to the system after at least 30 additional transactions had completed.
7. The system was powered back on and rebooted.
8. The mirrored disk removed in Step 4 was replaced by a new disk, the hardware based RAID recovery process of the array controller reinitialized the disk and cloned it from the other disk in the mirrored pair.
9. Step 2 was performed, giving hist2. It was verified that hist2 – hist1 was equal to or greater than the number of records in the success file.
10. Consistency test was verified.

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.

See section 4.4.2.

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 section 4.4.2

5.0 Clause 4: Scaling and Database Population

5.1 Initial 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 5.1 lists the TPC Benchmark H defined tables and the row count for each table as they existed upon completion of the build.

Table 5. 1: Initial Number of Rows

Table Name	Row Count
Region	5
Nation	25
Supplier	3,000,000
Customer	45,000,000
Part	60,000,000
Partsupp	240,000,000
Orders	450,000,000
Lineitem	1,799,989,091

5.2 Distribution of Tables and Logs Across Media

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

DB2 was configured on the HP Proliant DL585 test system with
8 x HP Smart Array P600 SAS controllers
8 x HP StorageWorks MSA50 disk enclosures
160 x 36.4GB SAS 10K SFF external disk drives
2 x 36.4GB 15K U320 internal disk drives

For each of the 4 logical nodes, 38 disks were used for the table data tablespaces (LINEITEM_TABLE, OTHER_TABLES), the index tablespaces (LINITEM_INDEXES, OTHER_INDEXES) and the temporary tablespaces (TEMP_TABLES, TEMP2_TABLES). These disks were not mirrored.

For each node, the database log was distributed on 2 disks using hardware RAID1. For each mirror, the primary and secondary copies were on separate disks.

The OS and benchmark execution programs were shared by each logical node and resided on the OS disks. The tablespace for SMALL_TABLES (NATION and REGION) also resided on the OS disks. The OS disks were internal to the server and used hardware RAID1. For each mirror, the primary and secondary copies were on separate disks.

A detailed description of distribution of tablespaces and log can be found in Table 5.2.1.

Table 5.2.1: SMART Array Controller Disk Array to Logical Drive Mapping

Controller	Drives	Logical Node/Partition	Size	Use
SA5i (integrated)	2 – 36.4GB RAID1+0	/dev/cciss/c0d0p1	102 MB	Boot sector
		/dev/cciss/c0d0p2	2048 MB	Linux swap
		/dev/cciss/c0d0p3	32578 MB	OS, DB, kit, small_tables
SA-P600-SAS (Slot 1)	18 – 36.4GB RAID0 204800 MB	Logical Node 2		
		/dev/cciss/c3d0p1	38792 MB	temp2_tables
		/dev/cciss/c3d0p2	33951 MB	lineitem_table
		/dev/cciss/c3d0p3	4563 MB	linitem_indexes
		/dev/cciss/c3d0p4	127497 MB	Extended partition
		/dev/cciss/c3d0p5	15655 MB	other_tables
		/dev/cciss/c3d0p6	2746 MB	other_indexes
		/dev/cciss/c3d0p7	109097 MB	temp_tables,
	RAID1+0 209904 MB	/dev/cciss/c3d1p1	55080 MB	/backup_2
		/dev/cciss/c3d1p2	189525 MB	/dev/md4 OS RAID0 stripset for /flatfiles and /flatfiles/ufdata
	2 - 36.4GB RAID1+0 34699 MB	/dev/cciss/c3d2p1	16384 MB	log for node 2
SA-P600-SAS (Slot 2)	18 – 36.4GB RAID0 204800 MB	Logical Node 3		
		/dev/cciss/c4d0p1	38792 MB	temp2_tables
		/dev/cciss/c4d0p2	33951 MB	lineitem_table
		/dev/cciss/c4d0p3	4563 MB	linitem_indexes
		/dev/cciss/c4d0p4	127497 MB	Extended partition
		/dev/cciss/c4d0p5	15655 MB	other_tables
		/dev/cciss/c4d0p6	2746 MB	other_indexes
		/dev/cciss/c4d0p7	109097 MB	temp_tables,
	RAID1+0 209904 MB	/dev/cciss/c4d1p1	55080 MB	/backup_3
		/dev/cciss/c4d1p2	189525 MB	/dev/md4 OS RAID0 stripset for /flatfiles and /flatfiles/ufdata
	2 - 36.4GB RAID1+0 34699 MB	/dev/cciss/c4d2p1	16384 MB	log for node 3

SA-P600-SAS (Slot 3)	20 – 36.4GB	Logical Node 1		
	RAID0 204800 MB	/dev/cciss/c6d0p1	38792 MB	temp2_tables
		/dev/cciss/c6d0p2	33951 MB	lineitem_table
		/dev/cciss/c6d0p3	4563 MB	linitem_indexes
		/dev/cciss/c6d0p4	127497 MB	Extended partition
		/dev/cciss/c6d0p5	15655 MB	other_tables
		/dev/cciss/c6d0p6	2746 MB	other_indexes
		/dev/cciss/c6d0p7	109097 MB	temp_tables,
	RAID1+0 244604 MB	/dev/cciss/c6d1p1	55080 MB	/backup_5
		/dev/cciss/c6d1p2	189525 MB	/dev/md4 OS RAID0 stripset for /flatfiles and /flatfiles/ufdata
SA-P600-SAS (Slot 4)	20 – 36.4GB	Logical Node 0		
	RAID0 204800 MB	/dev/cciss/c5d0p1	38792 MB	temp2_tables
		/dev/cciss/c5d0p2	33951 MB	lineitem_table
		/dev/cciss/c5d0p3	4563 MB	linitem_indexes
		/dev/cciss/c5d0p4	127497 MB	Extended partition
		/dev/cciss/c5d0p5	15655 MB	other_tables
		/dev/cciss/c5d0p6	2746 MB	other_indexes
		/dev/cciss/c5d0p7	109097 MB	temp_tables,
	RAID1+0 244604 MB	/dev/cciss/c5d1p1	55080 MB	/backup_4
		/dev/cciss/c5d1p2	189525 MB	/dev/md4 OS RAID0 stripset for /flatfiles and /flatfiles/ufdata
SA-P600-SAS (Slot 5)	20 – 36.4GB	Logical Node 3		
	RAID0 204800 MB	/dev/cciss/c8d0p1	38792 MB	temp2_tables
		/dev/cciss/c8d0p2	33951 MB	lineitem_table
		/dev/cciss/c8d0p3	4563 MB	linitem_indexes
		/dev/cciss/c8d0p4	127497 MB	Extended partition
		/dev/cciss/c8d0p5	15655 MB	other_tables
		/dev/cciss/c8d0p6	2746 MB	other_indexes
		/dev/cciss/c8d0p7	109097 MB	temp_tables,
	RAID1+0 244604 MB	/dev/cciss/c8d1p1	55080 MB	/backup_7
		/dev/cciss/c8d1p2	189525 MB	/dev/md4 OS RAID0 stripset for /flatfiles and /flatfiles/ufdata

SA-P600-SAS (Slot 6)	20 – 36.4GB	Logical Node 2		
	RAID0 204800 MB	/dev/cciss/c7d0p1	38792 MB	temp2_tables
		/dev/cciss/c7d0p2	33951 MB	lineitem_table
		/dev/cciss/c7d0p3	4563 MB	linitem_indexes
		/dev/cciss/c7d0p4	127497 MB	Extended partition
		/dev/cciss/c7d0p5	15655 MB	other_tables
		/dev/cciss/c7d0p6	2746 MB	other_indexes
		/dev/cciss/c7d0p7	109097 MB	temp_tables,
	RAID1+0 244604 MB	/dev/cciss/c7d1p1	55080 MB	/backup_6
		/dev/cciss/c7d1p2	189525 MB	/dev/md4 OS RAID0 stripset for /flatfiles and /flatfiles/ufdata
SA-P600-SAS (Slot 7)	18 – 36.4GB	Logical Node 1		
	RAID0 204800 MB	/dev/cciss/c2d0p1	38792 MB	temp2_tables
		/dev/cciss/c2d0p2	33951 MB	lineitem_table
		/dev/cciss/c2d0p3	4563 MB	linitem_indexes
		/dev/cciss/c2d0p4	127497 MB	Extended partition
		/dev/cciss/c2d0p5	15655 MB	other_tables
		/dev/cciss/c2d0p6	2746 MB	other_indexes
		/dev/cciss/c2d0p7	109097 MB	temp_tables,
	RAID1+0 209904 MB	/dev/cciss/c2d1p1	55080 MB	/backup_1
		/dev/cciss/c2d1p2	189525 MB	/dev/md4 OS RAID0 stripset for /flatfiles and /flatfiles/ufdata
	2 - 36.4GB			
	RAID1+0 34699 MB	/dev/cciss/c2d2p1	16384 MB	log for node 1
SA-P600-SAS (Slot 8)	18 – 36.4GB	Logical Node 0		
	RAID0 204800 MB	/dev/cciss/c1d0p1	38792 MB	temp2_tables
		/dev/cciss/c1d0p2	33951 MB	lineitem_table
		/dev/cciss/c1d0p3	4563 MB	linitem_indexes
		/dev/cciss/c1d0p4	127497 MB	Extended partition
		/dev/cciss/c1d0p5	15655 MB	other_tables
		/dev/cciss/c1d0p6	2746 MB	other_indexes
		/dev/cciss/c1d0p7	109097 MB	temp_tables,
	RAID1+0 209904 MB	/dev/cciss/c1d1p1	55080 MB	/backup_0
		/dev/cciss/c1d1p2	189525 MB	/dev/md4 OS RAID0 stripset for /flatfiles and /flatfiles/ufdata
	2 - 36.4GB			
	RAID1+0 34699 MB	/dev/cciss/c1d2p1	16384 MB	log for node 0

Note: The disks, listed in the table above, that were used to store the flat data files were priced. The disks remained physically connected to the controllers after the initial loading of the 300 GB database. The disks remained physically attached to the system during the execution of the power, throughput, qualification and ACID tests.

5.3 Mapping of Database Partitions/Replications

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

The database was not replicated. The database was physically partitioned into 4 logical nodes.

5.4 Implementation of RAID

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

RAID 0 was used for the database data tables and temporary table spaces. RAID1 was used for the database recovery logs. The Nation and Region tables were on internal drives configured with RAID1.

5.5 DBGEN Modifications

The version number, release number, modification number, and patch level of DBGEN must be disclosed. Any modifications to the DBGEN (see Clause 4.2.1) source code must be disclosed. In the event that a program other than DBGEN was used to populate the database, it must be disclosed in its entirety.

The standard distribution of DBGEN 1.3.0 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 by dividing the total data storage of the priced configuration (expressed in GB) by the size chosen for the test database as defined in 4.1.3.1. The ratio must be reported to the nearest 1/100th, rounded up.

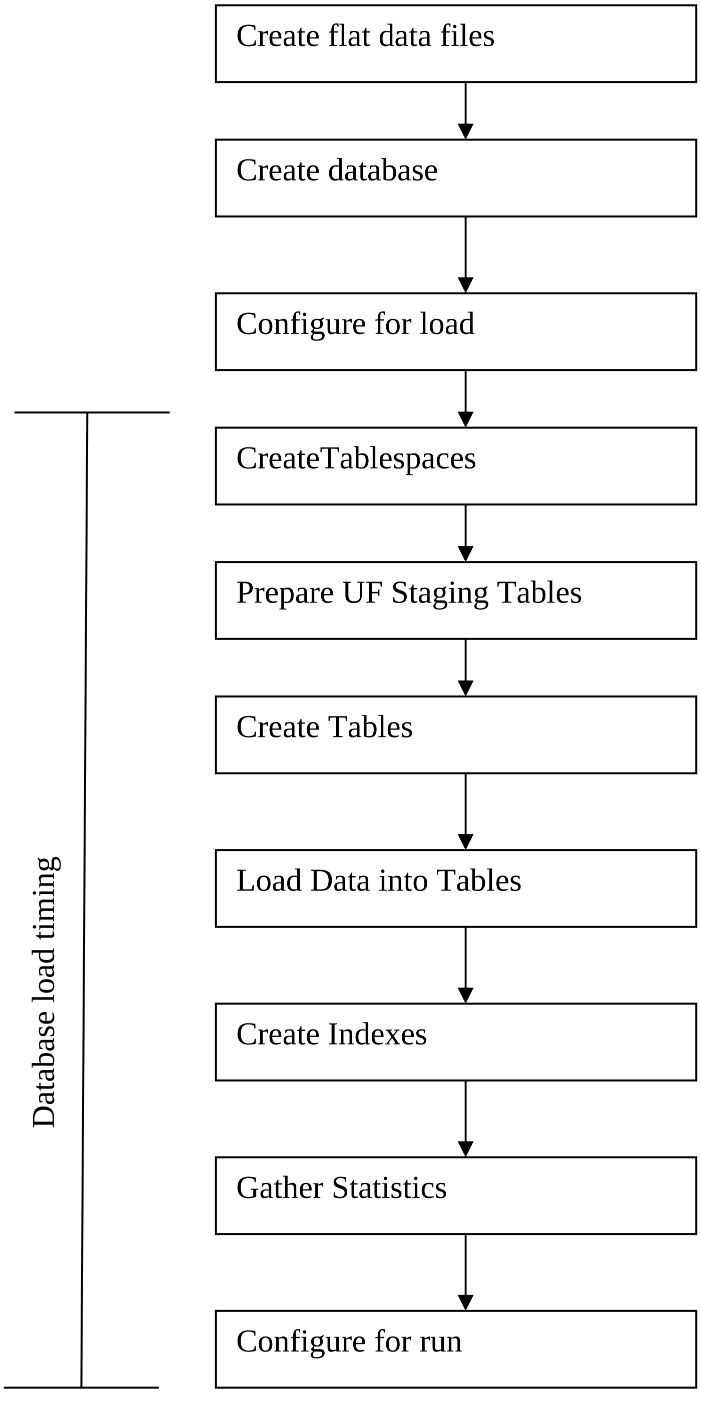
Disk Type	Number of Disks	Space per Disk	Total Disk Space	Scale Factor	Data Storage Ratio
36.4 GB 15K U320	2	36.4 GB	72.8 GB		
36.4 GB 10K SAS	160	36.4 GB	5824.0 GB		
			5896.8 GB	300GB	19.7

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 on node 1 and then loaded from dbgen output. The other tables were loaded on all of the nodes.
The tables were loaded as depicted in Figure 5.8.

Figure 5.8: Block Diagram of Database Load Process



6.0 Clause 5: Performance Metrics and Execution Rules Related Items

6.1 Steps in the Power Test

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

The following steps were used to implement the power test:

1 The system was rebooted
2. RF1 Refresh Transaction
3. Stream 00 Execution
4. RF2 Refresh Transaction.

6.2 Timing Intervals for Each Query and Refresh Function

The timing intervals (see Clause 5.3.6) for each query of the measured set and for both refresh functions must be reported for the power test.

The timing intervals for each query and both refresh functions are given in the Numerical Quantities Summary earlier in the executive summary.

6.3 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.

6.4 Start and End Date/Times for Each Query Stream

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

The Numerical Quantities Summary contains the start and stop times for the query execution streams run on the system reported.

6.5 Total Elapsed Time for the Measurement Interval

The total elapsed time of the measurement interval(see Clause 5.3.5) must be reported for the throughput test.

The Numerical Quantities Summary contains the timing intervals for the throughput test run on the system reported.

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

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

The refresh function start date/time and finish data/time are given in the Numerical Quantities Summary earlier in the executive summary.

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

The timing intervals (see Clause 5.3.6) for each query of each stream and for each update function must be reported for the throughput test.

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

6.8 Performance Metrics

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

The Numerical Quantities Summary contains the performance metrics, related numerical quantities, and the price/performance metric for the system reported.

6.9 The Performance Metric and Numerical Quantities from Both Runs

A description of the method used to determine the reproducibility of the measurement results must be reported. This must include the performance metrics (QppH and QthH) from the reproducibility runs.

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

Run	QppH @ 300GB	QthH @ 300GB	QphH @ 300GB
Run 1	15504.0	9157.2	11915.3
Run 2	15500.6	9361.7	12046.2

6.11 System Activity Between Tests

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

The system was not restarted between runs.

7.0 Clause 6: SUT and Driver Implementation Related Items

7.1 Driver

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

Appendix D "Implementation specific layer and 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 (ISL)

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. For RF1, the data for each node is further split into n equal por-tions 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 the temporary tables (one each for lineitem and orders), containing a column to hold the chunk number. 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

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

Profile-directed optimization was not used.

8.0 Clause 7: Pricing Related Items

8.1 Hardware and Software Used

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

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, at the end of this document.

8.2 Total Three Year Price

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

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, at the end of this document. For a large purchase and cash discount, this purchase qualifies for a 16% discount from HP Corporation.

8.3 Availability Date

The committed delivery date for general availability of products used in the price calculations must be reported. When the priced system includes products with different availability dates, the 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.

The HP ProLiant DL585, system memory, additional processors, disk controllers and hard drives are available at the time of publication. All other hardware is generally available at the time of publication.

The system software, Red Hat Enterprise Linux 4 AS Update 2, used in this test will be generally available on or before October 5, 2005. The database software, IBM DB2 UDB 8.2 for Linux is generally available at the time of publication. Fix Pack 9 for DB2 is generally available at the time of publication.

8.4 Country-Specific Pricing

Additional Clause 7 related items August 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

.

9.0 Clause 9: Related Items

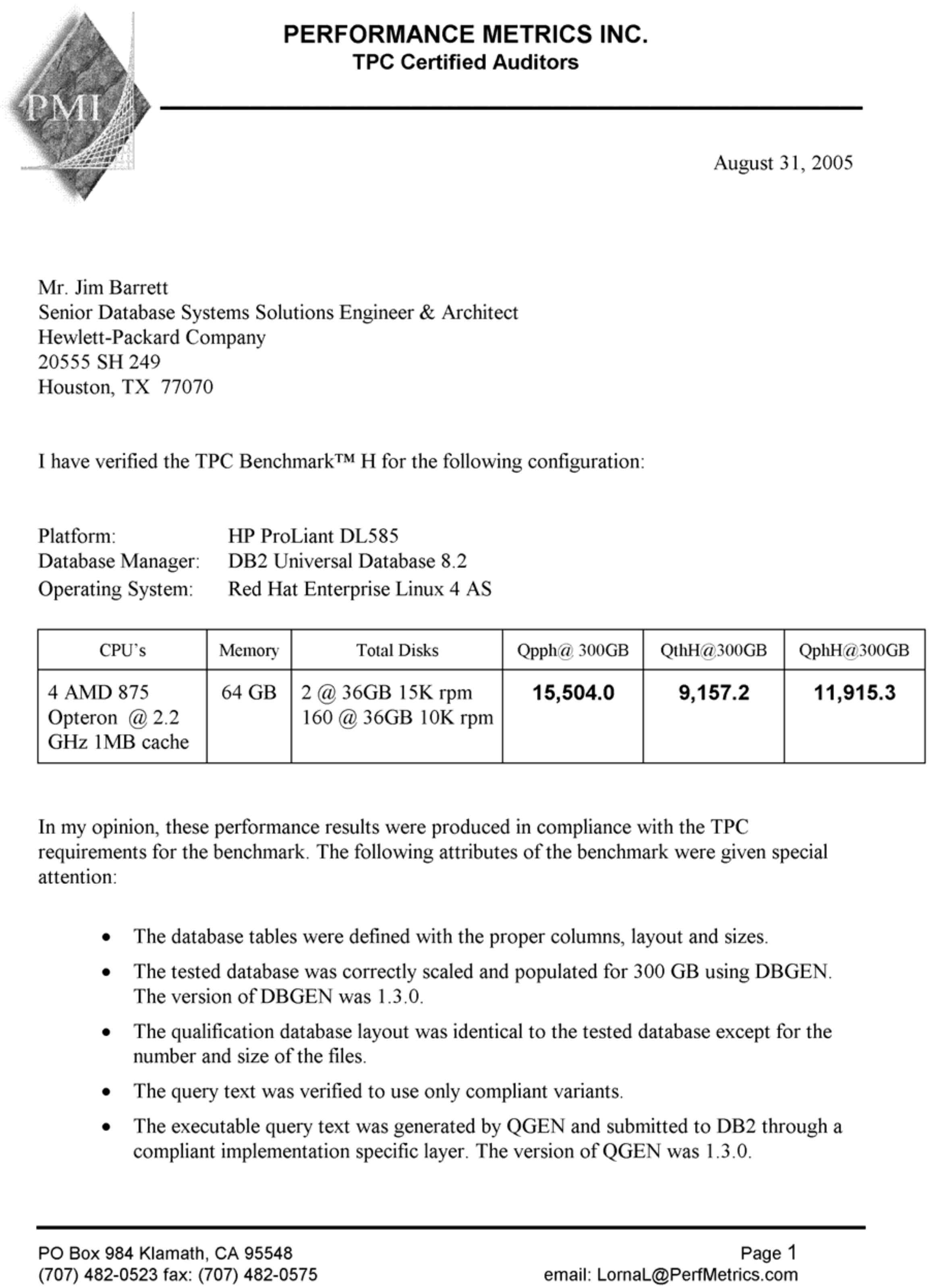
9.1 Auditors' Report

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 Lorna Livingtree of Performance Metrics (www.perfmetrics.com). Further information regarding the audit process August be obtained from:

Performance Metrics, Inc.
PO Box 984
Klamath, CA 95548
Telephone: (707) 482-0523
Fax: (707) 482-0575

For a copy of this disclosure, go to www.tpc.org.



PERFORMANCE METRICS INC.
TPC Certified Auditors

August 31, 2005

Mr. Jim Barrett
Senior Database Systems Solutions Engineer & Architect
Hewlett-Packard Company
20555 SH 249
Houston, TX 77070

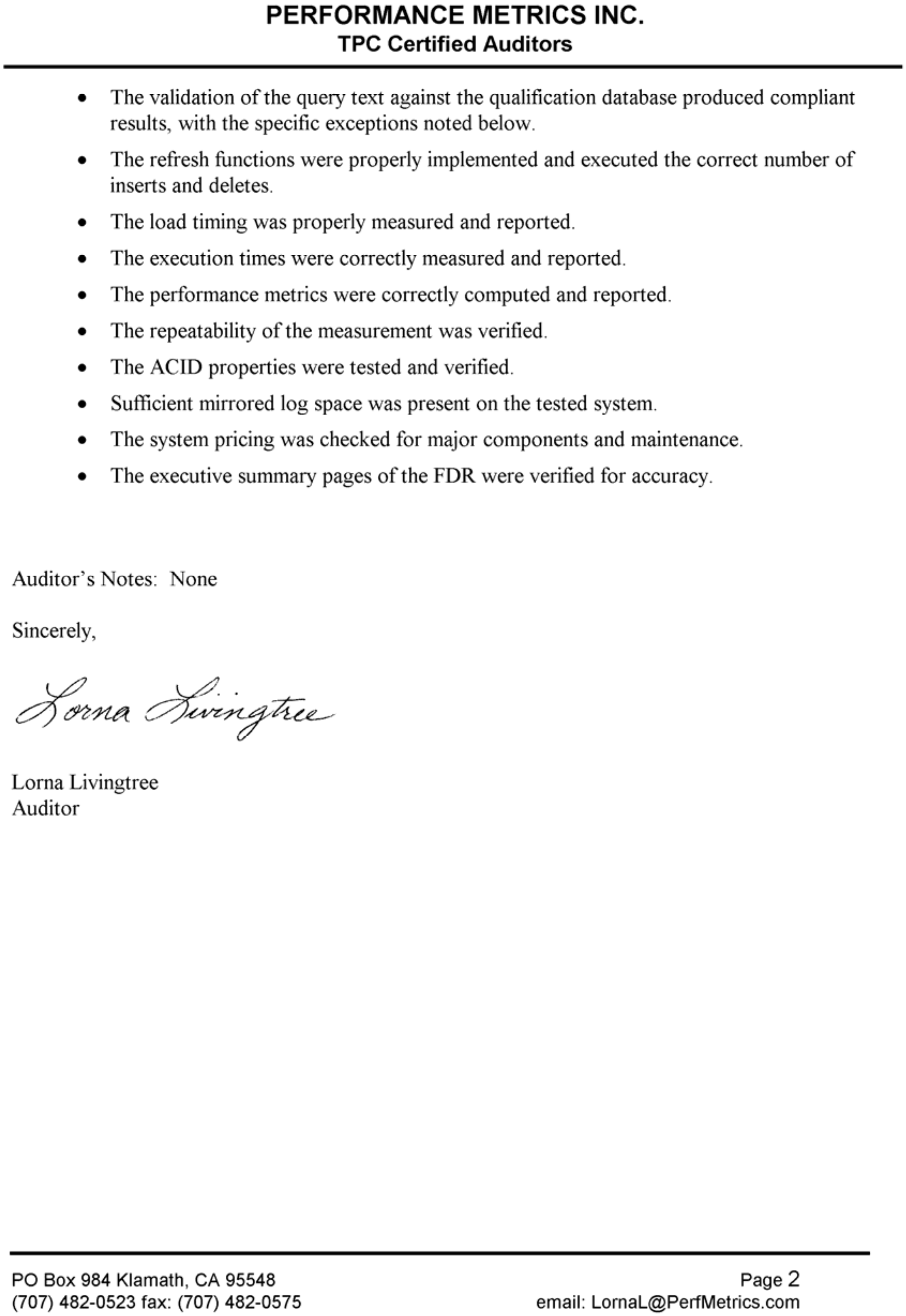
I have verified the TPC Benchmark™ H for the following configuration:

Platform:	HP ProLiant DL585
Database Manager:	DB2 Universal Database 8.2
Operating System:	Red Hat Enterprise Linux 4 AS

CPU's	Memory	Total Disks	Qpph@ 300GB	QthH@300GB	QphH@300GB
4 AMD 875 Opteron @ 2.2 GHz 1MB cache	64 GB	2 @ 36GB 15K rpm 160 @ 36GB 10K rpm	**15,504.0**	**9,157.2**	**11,915.3**

In my opinion, these performance results were produced in compliance with the TPC requirements for the benchmark. The following attributes of the benchmark were given special attention:

- The database tables were defined with the proper columns, layout and sizes.
- The tested database was correctly scaled and populated for 300 GB using DBGEN. The version of DBGEN was 1.3.0.
- The qualification database layout was identical to the tested database except for the number and size of the files.
- The query text was verified to use only compliant variants.
- The executable query text was generated by QGEN and submitted to DB2 through a compliant implementation specific layer. The version of QGEN was 1.3.0.



- The validation of the query text against the qualification database produced compliant results, with the specific exceptions noted below.
- The refresh functions were properly implemented and executed the correct number of inserts and deletes.
- The load timing was properly measured and reported.
- The execution times were correctly measured and reported.
- The performance metrics were correctly computed and reported.
- The repeatability of the measurement was verified.
- The ACID properties were tested and verified.
- Sufficient mirrored log space was present on the tested system.
- The system pricing was checked for major components and maintenance.
- The executive summary pages of the FDR were verified for accuracy.

Auditor's Notes: None

Sincerely,

Lorna Livingtree

Lorna Livingtree
Auditor

Appendix A: Tunable Parameters

- *Note: These are the settings used during the power test. The settings altered for the load are documented in Appendix B.*

A.1 DB2 UDB 8.2 Database Configuration (nodes 0-3)

Node 0

```
      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) = ANY
 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                          = NO
 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) = 40000
 Size of database shared memory (4KB)  (DATABASE_MEMORY) =
AUTOMATIC
 Catalog cache size (4KB)            (CATALOGCACHE_SZ) =
(MAXAPPLS*4)
 Log buffer size (4KB)                  (LOGBUFSZ) = 2048
 Utilities heap size (4KB)              (UTIL_HEAP_SZ) = 20000
 Buffer pool size (pages)                (BUFFPAGE) = 10240
 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) = 60000

 Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 40000
 Percent of mem for appl. group heap   (GROUPHEAP_RATIO) = 70
 Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 2058

 Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) =
(SHEAPTHRES)
 Sort list heap (4KB)                   (SORTHEAP) = 20480
 SQL statement heap (4KB)                 (STMTHEAP) = 40000
 Default application heap (4KB)           (APPLHEAPSZ) = 16000
 Package cache size (4KB)               (PCKCACHESZ) = (MAXAPPLS*8)
 Statistics heap size (4KB)             (STAT_HEAP_SZ) = 20000

 Interval for checking deadlock (ms)        (DLCHKTIME) = 5000
 Percent. of lock lists per application      (MAXLOCKS) = 40
 Lock timeout (sec)                    (LOCKTIMEOUT) = -1

 Changed pages threshold              (CHNGPGS_THRESH) = 60
 Number of asynchronous page cleaners   (NUM_IOCLEANERS) = 4
 Number of I/O servers                 (NUM_IOSERVERS) = 16
 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) = 4
 Number of secondary log files             (LOGSECOND) = 2
 Changed path to log files                (NEWLOGPATH) =
 Path to log files                              = /dev/cciss/c1d2p1
 Overflow log path                  (OVERFLOWLOGPATH) =
 Mirror log path                   (MIRRORLOGPATH) =
 First active log file                           = S0000030.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) = 100
 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
```

Node 1

```
       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) = ANY
 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                          = NO
 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) = 40000
 Size of database shared memory (4KB)  (DATABASE_MEMORY) =
AUTOMATIC
 Catalog cache size (4KB)            (CATALOGCACHE_SZ) =
(MAXAPPLS*4)
 Log buffer size (4KB)                    (LOGBUFSZ) = 2048
 Utilities heap size (4KB)              (UTIL_HEAP_SZ) = 20000
 Buffer pool size (pages)                 (BUFFPAGE) = 10240
 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) = 60000

 Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 40000
 Percent of mem for appl. group heap   (GROUPHEAP_RATIO) = 70
 Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 2058

 Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) =
(SHEAPTHRES)
 Sort list heap (4KB)                   (SORTHEAP) = 20480
 SQL statement heap (4KB)                 (STMTHEAP) = 40000
 Default application heap (4KB)           (APPLHEAPSZ) = 16000
 Package cache size (4KB)                (PCKCACHESZ) = (MAXAPPLS*8)
 Statistics heap size (4KB)             (STAT_HEAP_SZ) = 20000

 Interval for checking deadlock (ms)        (DLCHKTIME) = 5000
 Percent. of lock lists per application       (MAXLOCKS) = 40
 Lock timeout (sec)                     (LOCKTIMEOUT) = -1

 Changed pages threshold               (CHNGPGS_THRESH) = 60
 Number of asynchronous page cleaners   (NUM_IOCLEANERS) = 4
 Number of I/O servers                 (NUM_IOSERVERS) = 16
 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) = 4
 Number of secondary log files             (LOGSECOND) = 2
 Changed path to log files               (NEWLOGPATH) =
 Path to log files                              = /dev/cciss/c2d2p1
 Overflow log path                   (OVERFLOWLOGPATH) =
 Mirror log path                    (MIRRORLOGPATH) =
 First active log file                            = S0000030.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) = 100
 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
```

Node 2

```
       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) = ANY
 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                                  = NO
 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) = 40000
 Size of database shared memory (4KB)  (DATABASE_MEMORY) =
AUTOMATIC
 Catalog cache size (4KB)              (CATALOGCACHE_SZ) =
(MAXAPPLS*4)
 Log buffer size (4KB)                        (LOGBUFSZ) = 2048
 Utilities heap size (4KB)                (UTIL_HEAP_SZ) = 20000
 Buffer pool size (pages)                     (BUFFPAGE) = 10240
 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) = 60000

 Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 40000
 Percent of mem for appl. group heap   (GROUPHEAP_RATIO) = 70
 Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 2058

 Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) =
(SHEAPTHRES)
 Sort list heap (4KB)                         (SORTHEAP) = 20480
 SQL statement heap (4KB)                     (STMTHEAP) = 40000
 Default application heap (4KB)             (APPLHEAPSZ) = 16000
 Package cache size (4KB)                  (PCKCACHESZ) = (MAXAPPLS*8)
 Statistics heap size (4KB)               (STAT_HEAP_SZ) = 20000

 Interval for checking deadlock (ms)         (DLCHKTIME) = 5000
 Percent. of lock lists per application       (MAXLOCKS) = 40
 Lock timeout (sec)                        (LOCKTIMEOUT) = -1

 Changed pages threshold                (CHNGPGS_THRESH) = 60
 Number of asynchronous page cleaners   (NUM_IOCLEANERS) = 4
 Number of I/O servers                   (NUM_IOSERVERS) = 16
 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) = 4
 Number of secondary log files               (LOGSECOND) = 2
 Changed path to log files                  (NEWLOGPATH) =
 Path to log files                                       = /dev/cciss/c3d2p1
 Overflow log path                     (OVERFLOWLOGPATH) =
 Mirror log path                         (MIRRORLOGPATH) =
 First active log file                                   = S0000030.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) = 100
 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
```

Node 3

```
       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) = ANY
 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                          = NO
 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) = 40000
 Size of database shared memory (4KB)  (DATABASE_MEMORY) =
AUTOMATIC
 Catalog cache size (4KB)             (CATALOGCACHE_SZ) =
(MAXAPPLS*4)
 Log buffer size (4KB)                    (LOGBUFSZ) = 2048
 Utilities heap size (4KB)             (UTIL_HEAP_SZ) = 20000
 Buffer pool size (pages)                 (BUFFPAGE) = 10240
 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) = 60000

 Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 40000
 Percent of mem for appl. group heap   (GROUPHEAP_RATIO) = 70
 Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 2058

 Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) =
(SHEAPTHRES)
 Sort list heap (4KB)                    (SORTHEAP) = 20480
 SQL statement heap (4KB)                  (STMTHEAP) = 40000
 Default application heap (4KB)           (APPLHEAPSZ) = 16000
 Package cache size (4KB)                (PCKCACHESZ) = (MAXAPPLS*8)
 Statistics heap size (4KB)             (STAT_HEAP_SZ) = 20000

 Interval for checking deadlock (ms)        (DLCHKTIME) = 5000
 Percent. of lock lists per application      (MAXLOCKS) = 40
 Lock timeout (sec)                     (LOCKTIMEOUT) = -1

 Changed pages threshold              (CHNGPGS_THRESH) = 60
 Number of asynchronous page cleaners   (NUM_IOCLEANERS) = 4
 Number of I/O servers                (NUM_IOSERVERS) = 16
 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) = 4
 Number of secondary log files              (LOGSECOND) = 2
 Changed path to log files               (NEWLOGPATH) =
 Path to log files                            = /dev/cciss/c4d2p1
 Overflow log path                    (OVERFLOWLOGPATH) =
 Mirror log path                        (MIRRORLOGPATH) =
 First active log file                                 = S0000030.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) = 100
 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
```

A.2 DB2 UDB 8.2 Database Manager Configuration

```
          Database Manager Configuration

     Node type = Enterprise Server Edition with local and remote clients

 Database manager configuration release level            = 0x0a00

 CPU speed (millisec/instruction)             (CPUSPEED) = 1.889377e-07
 Communications bandwidth (MB/sec)      (COMM_BANDWIDTH) =
2.800000e+00

 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) = 0
 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/tpch

 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) = 320000

 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) = DB2_tpch
 Discovery mode                               (DISCOVER) = SEARCH
 Discover server instance                (DISCOVER_INST) = ENABLE

 Maximum query degree of parallelism   (MAX_QUERYDEGREE) = ANY
 Enable intra-partition parallelism     (INTRA_PARALLEL) = YES

 No. of int. communication buffers(4KB)(FCM_NUM_BUFFERS) = 20480
 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
```

A.3 DB2 Environment Variables

```
DB2LINUXAIO=TRUE
DB2_SCATTERED_IO=OFF
DB2_LGPAGE_BP=ON
DB2_EXTENDED_OPTIMIZATION=Y
DB2_ANTIJOIN=Y
DB2_STRIPED_CONTAINERS=ON
DB2BPVARS=/home/tpch/custom/bpvar.cfg
DB2_FORCE_FCM_BP=YES
DB2OPTIONS=-t -v +c
DB2_PARALLEL_IO=*
```

A.4 DB2 Version

```
Database and Database manager configuration taken at : Mon Aug 22
23:00:30 CDT 2005
DB21085I  Instance "tpch" uses "64" bits and DB2 code release "SQL08022"
with
level identifier "03030106".
Informational tokens are "DB2 v8.1.1.88", "s050330", "MI00108", and
FixPak "9".
Product is installed at "/opt/IBM/db2/V8.1".
```

A.5 RHEL AS 4 Version

```
Linux bluemoon 2.6.9-15.EL.smp #1 SMP Fri Aug 5 19:00:35 EDT 2005
x86_64 x86_64 x86_64 GNU/Linux
```

A.6 RHEL AS 4 Configuration

/boot/grub/grub.conf

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/cciss/c0d0p3
#          initrd /initrd-version.img
#boot=/dev/cciss/c0d0
default=1
fallback=0
splashimage=(hd0,0)/grub/splash.xpm.gz
#hiddenmenu
title Red Hat Enterprise Linux AS (2.6.9-15.ELsmp)
        root (hd0,0)
        kernel /vmlinuz-2.6.9-15.ELsmp ro root=LABEL=/ numa=on
selinux=0 elevator=deadline
        initrd /initrd-2.6.9-15.ELsmp.img
title HP-2.6.9-15.ELsmp-2
        root (hd0,0)
        kernel /vmlinuz-2.6.9-15.ELsmp ro root=LABEL=/ numa=on
selinux=0 elevator=deadline
        initrd /HP-initrd-2.6.9-15.ELsmp.img-2
```

/etc/rc.local

```
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local

rdate -s 10.1.0.1

chown -R tpch:db2iadm1 /dev/cciss
chmod -R 777 /dev/cciss

# Set smp_affinity for P600 cciss0 thru cciss8
echo "1" > /proc/irq/193/smp_affinity
echo "1" > /proc/irq/217/smp_affinity
echo "2" > /proc/irq/225/smp_affinity
echo "4" > /proc/irq/233/smp_affinity
echo "8" > /proc/irq/50/smp_affinity
echo "1" > /proc/irq/58/smp_affinity
echo "2" > /proc/irq/66/smp_affinity
echo "4" > /proc/irq/74/smp_affinity
echo "8" > /proc/irq/82/smp_affinity
```

/etc/sysctl.conf

```
# Kernel sysctl configuration file for Red Hat Linux
#
# For binary values, 0 is disabled, 1 is enabled.  See sysctl(8) and
# sysctl.conf(5) for more details.

# Controls IP packet forwarding
net.ipv4.ip_forward = 0

# Controls source route verification
net.ipv4.conf.default.rp_filter = 1

# Do not accept source routing
```

```
net.ipv4.conf.default.accept_source_route = 0

# Controls the System Request debugging functionality of the kernel
kernel.sysrq = 0

# Controls whether core dumps will append the PID to the core filename.
# Useful for debugging multi-threaded applications.
kernel.core_uses_pid = 1

# DB2 settings for 64GB memory
vm.swappiness = 0
## the following line was commentd out for the load portion of the test
## it was uncommentted before beginning the perfromance runs
vm.nr_hugepages = 29952
kernel.sem = 250 1024000 400 4096
kernel.shmall = 66060288
kernel.shmmax = 64424509440
kernel.msgmnb = 32768
kernel.msgmni = 16384
```

/etc/sysconfig/i18n

```
LANG=”en_US.ISO8859-1”
SUPPORTED=”en_US.ISO8859-1:en_US:en”
SYSFONT=”latarcyrheb-sun16”
```

A.7 HP Smart Array P600 SAS Controller Cache Settings

The HP P600 SAS Controller cache settings for all the database data drives was configured for the ratio of 0% read / 100% write .
The HP P600 SAS Controller cache settings for the database log drives was disabled.

Appendix B: Database Build Scripts

B.1 alter_bufferpools

```
--------------------------------------------------------
-- Alter Bufferpools
--------------------------------------------------------
ALTER BUFFERPOOL IBMDEFAULTBP SIZE 153600;
COMMIT WORK;
ALTER BUFFERPOOL BP32K SIZE 281600;
COMMIT WORK;
ALTER BUFFERPOOL BP32K NUMBLOCKPAGES 42240;
COMMIT WORK;
ALTER BUFFERPOOL BP32KTEMP SIZE 38400;
COMMIT WORK;
CONNECT RESET;
DB2STOP FORCE;
DB2START;
```

B.2 backupdb.ksh

```
#!/bin/ksh
# backup the database.
# The system has been configured such that each node is backed up on the
subsequent node.
# Node 0 backup up to Node 1, Node 1 backup up to Node 2 etc.
# Backup is to a file system so make output unbuffered.
# do catalog node first
tstamp=$(date +%y%m%d-%H%M%S)
echo "Backup Database Started " $tstamp
db2_all  "<<+0< db2 backup database tpcd to /backup_1, /backup_2,
/backup_3, /backup_5, /backup_6, /backup_7 with 16 buffers parallelism 8
without prompting      "
{
		db2_all  "<<+1< db2 backup database tpcd to /backup_2,
/backup_3, /backup_4, /backup_6, /backup_7, /backup_0 with 16 buffers
parallelism 8 without prompting " &
		db2_all  "<<+2< db2 backup database tpcd to /backup_3,
/backup_4, /backup_5, /backup_7, /backup_0, /backup_1 with 16 buffers
parallelism 8 without prompting " &
		db2_all  "<<+3< db2 backup database tpcd to /backup_4,
/backup_5, /backup_6, /backup_0, /backup_1, /backup_2 with 16 buffers
parallelism 8 without prompting " &
}
wait
tstamp=$(date +%y%m%d-%H%M%S)
echo "Backup Database Finished " $tstamp
```

B.3 bp.vars

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

B.4 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
require "getvars";
require "macro.pl";
require "tpcdmacro.pl";
require "version";
$timestamp=`perl gettimestamp "short"`;

# Make output unbuffered.
open(STDOUT, "| tee buildtpcd.out.${timestamp}");
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){
   $ENV{"TPCD_INDEXDDL"} = "dss.index";
}
if (length($ENV{"TPCD_RUNSTATS"}) <= 0){
   $ENV{"TPCD_RUNSTATS"} = "dss.runstats";
}
if (length($ENV{"TPCD_CONFIGFILE"}) <= 0){
   $ENV{"TPCD_CONFIGFILE"} = "dss.dbconfig";
}



#---------------------------------------------------------------------------#
#  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: 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: create nodegroups
failed.\n";
```

```
        !&createBufferPools || print "ERROR: create bufferpools
failed.\n";
        &outtime("*** Start of audited Load Time - starting to create
tables");
        !&createTablespaces || print "WARNING: create tablespaces
error.\n";
        !&createExplainTbls ||  print "ERROR: create EXPLAIN tables
failed.\n";
        !&createTables || print "ERROR: create tables failed.\n";

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

        # if earlyindex requested, create indexes
        if ( $earlyindex eq "yes" ){
                !&createIndexes("early") || die "ERROR: create early
indexes failed.\n";
        }
        # start the dbgen and load.....call the specific mode for loading
(uni,smp,mln)
        !&loadData || die "ERROR: 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: create
indexes failed.\n";
        }
        if ( $extraindex ne "no" ){
                !&createIndexes("extra") || die "ERROR: 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 grant connect on database to public $sep            \
            db2 grant dbadm on database to $dbname $sep             \
            db2 commit",$once);
        &dodb_noconn("db2 reset database manager configuration",$once);
}

#---------------------------------------------------------------------------#
# 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 $sep \
               db2 alter table tpcd.lineitem append on $sep \
               db2 alter table tpcd.orders append on",
              $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 $sep \
          db2 alter table tpcd.lineitem pctfree 0 $sep \
          db2 alter table tpcd.orders pctfree 0",
              $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 ($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 "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: genloaduni
failed rc = $rc\n";
        }
        else{
            $rc =
&dodb2file_noconn("$ddlpath${delim}$loadscript",$once);
            ($rc == 0) || print "ERROR: load script:
$loadscript failed. rc = $rc\n";
        }
    }
    elsif (( $mode eq "mln" ) || ( $mode eq "mpp" )){
        &outtime("*** Starting the load");
        # call the appropriate dbgen/split/(sort)/load for
mln/mpp
        if ( $loadscript eq "NULL"){
            $rc = system("perl genloadmpp $qual");
            ($rc == 0) || print "ERROR: genloadmpp
failed. rc = $rc\n";
        }
        else{
            system("$ddlpath${delim}$loadscript");
            #$rc =
&dodb2file_noconn("$ddlpath${delim}$loadscript $sf");
            #($rc == 0) || print "ERROR: 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 configuration for $dbname
using dft_degree $smpdegree",$all_ln);
              &dodb_noconn("db2 update database manager configuration
using max_querydegree $smpdegree",$once);
              &dodb2file_noconn("${ddlpath}${delim}$configfile",$all_ln);
              &dodb2file_noconn("${ddlpath}${delim}$dbmconfig",$once);

              if ( $agentpri ne "NULL" ){
                 &dodb_noconn("db2 update dbm cfg using AGENTPRI
$agentpri",$once);
              }
              # set the db2 environment variables for running the benchmark
              if ( $setScript ne "NULL" ){
                 if ( $platform eq "aix" || $platform eq "sun" || $platform eq
"ptx"){
                    $ret=system("${ddlpath}${delim}$setScript");
                 }
                 elsif ( $platform eq "nt" ){
                    if (($mode eq "uni" ) || ($mode eq "smp" )){
                       $ret = system("perl ${ddlpath}${delim}$setScript");
                    }
                    else{
                       $ret = system(" rah \" cd ${ddlpath} & $setScript\" ");
                    }
                 }
                 #($ret == 0 ) || die "failure setting runtime db2set parms from
$setScript \n";
              }
              &outtime("*** Alter Bufferpools started ***");
              $ret=&dodb2file($dbname,"$ddlpath${delim}alter_bufferpools.d
dl",$once);
              &outtime("*** Alter Bufferpools 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.
              &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);
                            #}
             $rc = 0;
             $rc=system("$ddlpath${delim}backupdb_qual.ksh");
                            ($rc == 0) || &outtime("*** Finished the backup");
              }
              else{
             &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);
             #$rc=system("db2_all \"db2 backup database $dbname to
$backupdir without prompting\"");
             #$rc=system("perl $ddlpath${delim}backupdb.pl");
             $rc = 0;
             $rc=system("$ddlpath${delim}backupdb.ksh");
             ($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}dss.dbconfig${sfReal}GB\n";
          $configfile="dss.dbconfig${sfReal}GB";
        }
        if ( $dbmconfig ne "NULL" ){
          print "   Database Manager Configuration parameters taken from
$ddlpath${delim}$dbmconfig\n";
        }
        else{
          print "   Database Manager Configuration paramters taken from
$ddlpath${delim}dss.dbmconfig${sfReal}GB\n";
          $configfile="dss.dbmconfig${sfReal}GB";
        }
        #print "   Copy image for load command created in $copydir\n";
        if ( $log eq "yes" ){
          print "   Backup files placed in $backupdir\n";
        }
        else{
          print "   No backup will be taken.\n";
        }
        print "   Log retain set to $log\n";
        if ( $logDir eq "NULL" ){
            print "   Log files remain in database path\n";
        }
        else{
            print "   Log file path set to $logDir\n";
        }
        if ( $logprimary eq "NULL" ){
            print "   Log Primary left at default\n";
        }
        else{
            print "   Log Primary set to $logprimary\n";
        }
        if ( $logsecond eq "NULL" ){
            print "   Log Second left at default\n";
        }
        else{
            print "   Log second set to $logsecond\n";
        }
        if ( $logfilsiz eq "NULL" ){
            print "   Logfilsiz left at default\n";
        }
        else{
            print "   Logfilsiz set to $logfilsiz\n";
        }
        if (($loadconfigfile eq "") || ($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;
```

B.5 create_bufferpools

```
---------------------------------------------------------
-- Create Bufferpools
---------------------------------------------------------
ALTER BUFFERPOOL IBMDEFAULTBP SIZE 102400;
COMMIT WORK;
CREATE BUFFERPOOL BP32K ALL NODES SIZE 102400
NUMBLOCKPAGES 10240 BLOCKSIZE 16 PAGESIZE 32K;
COMMIT WORK;
CREATE BUFFERPOOL BP32KTEMP ALL NODES SIZE 10240
PAGESIZE 32K;
COMMIT WORK;
```

B.6 create_indexes

```
----------------------------------------------------------
-- Create Indexes
----------------------------------------------------------
values(current timestamp);
ALTER TABLE TPCD.REGION ADD PRIMARY KEY
(R_REGIONKEY);
COMMIT WORK;

values(current timestamp);
ALTER TABLE TPCD.NATION ADD PRIMARY KEY
(N_NATIONKEY);
COMMIT WORK;

values(current timestamp);
ALTER TABLE TPCD.PART ADD PRIMARY KEY (P_PARTKEY);
COMMIT WORK;

values(current timestamp);
ALTER TABLE TPCD.SUPPLIER ADD PRIMARY KEY (S_SUPPKEY);
COMMIT WORK;

values(current timestamp);
ALTER TABLE TPCD.PARTSUPP ADD PRIMARY KEY
(PS_PARTKEY,PS_SUPPKEY);
COMMIT WORK;

values(current timestamp);
ALTER TABLE TPCD.CUSTOMER ADD PRIMARY KEY
(C_CUSTKEY);
COMMIT WORK;

values(current timestamp);
ALTER TABLE TPCD.LINEITEM ADD PRIMARY KEY
(L_ORDERKEY,L_LINENUMBER);
COMMIT WORK;

values(current timestamp);
ALTER TABLE TPCD.ORDERS ADD PRIMARY KEY
(O_ORDERKEY);
COMMIT WORK;

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

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

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

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

values(current timestamp);

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

B.7 create_nodegroups

```
------------------------------------------------------
-- Create Nodegroups
------------------------------------------------------
CREATE NODEGROUP catalog_node ON NODE (0);
CREATE NODEGROUP all_nodes;
```

B.8 create_tables

```
---------------------------------------------------------
-- 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_TABLES
             INDEX IN OTHER_INDEXES
             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_TABLES
             INDEX IN OTHER_INDEXES
     PARTITIONING KEY(S_SUPPKEY) USING HASHING;

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_TABLES
             INDEX IN OTHER_INDEXES
     PARTITIONING KEY(PS_PARTKEY) USING HASHING;

CREATE TABLE TPCD.CUSTOMER ( C_CUSTKEY     INTEGER NOT
NULL,
                    C_NAME       VARCHAR(25) NOT NULL,
                    C_ADDRESS     VARCHAR(40) NOT NULL,
                    C_NATIONKEY   INTEGER NOT NULL,
                    C_PHONE      CHAR(15) NOT NULL,
                    C_ACCTBAL     FLOAT    NOT NULL,
                    C_MKTSEGMENT  CHAR(10) NOT NULL,
                    C_COMMENT     VARCHAR(117) NOT NULL WITH
DEFAULT)
     IN OTHER_TABLES
             INDEX IN OTHER_INDEXES
     PARTITIONING KEY(C_CUSTKEY) USING HASHING;

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)
     IN OTHER_TABLES
             INDEX IN OTHER_INDEXES
     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)
     IN LINEITEM_TABLE
             INDEX IN LINEITEM_INDEXES
     PARTITIONING KEY(L_ORDERKEY) USING HASHING
             ORGANIZE BY (L_SHIPDATE);

COMMIT WORK;
```

B.9 create_tablespaces

```
------------------------------------------------------------
-- Create Tablespaces
------------------------------------------------------------

CREATE regular TABLESPACE small_tables
  IN NODEGROUP catalog_node
  PAGESIZE 4K
  MANAGED BY system
  USING ('/data/small_tables')
  NO FILE SYSTEM CACHING
  BUFFERPOOL IBMDEFAULTBP
  OVERHEAD 7.0
  TRANSFERRATE 0.52;
COMMIT WORK;

CREATE temporary TABLESPACE temp2_tables
  PAGESIZE 4K
  MANAGED BY database
  USING (
    DEVICE '/dev/cciss/c1d0p1' 38784M,
    DEVICE '/dev/cciss/c5d0p1' 38784M
  ) ON NODE (0)
  USING (
    DEVICE '/dev/cciss/c2d0p1' 38784M,
    DEVICE '/dev/cciss/c6d0p1' 38784M
  ) ON NODE (1)
  USING (
    DEVICE '/dev/cciss/c3d0p1' 38784M,
    DEVICE '/dev/cciss/c7d0p1' 38784M
  ) ON NODE (2)
  USING (
    DEVICE '/dev/cciss/c4d0p1' 38784M,
    DEVICE '/dev/cciss/c8d0p1' 38784M
  ) ON NODE (3)
  BUFFERPOOL IBMDEFAULTBP
  EXTENTSIZE 128
  OVERHEAD 0.0579
  TRANSFERRATE 0.52;

CREATE regular TABLESPACE lineitem_table
  IN NODEGROUP all_nodes
  PAGESIZE 32K
  MANAGED BY database
  USING (
    DEVICE '/dev/cciss/c1d0p2' 33935M,
    DEVICE '/dev/cciss/c5d0p2' 33935M
  ) ON NODE (0)
  USING (
    DEVICE '/dev/cciss/c2d0p2' 33935M,
    DEVICE '/dev/cciss/c6d0p2' 33935M
  ) ON NODE (1)
  USING (
    DEVICE '/dev/cciss/c3d0p2' 33935M,
    DEVICE '/dev/cciss/c7d0p2' 33935M
  ) ON NODE (2)
  USING (
    DEVICE '/dev/cciss/c4d0p2' 33935M,
    DEVICE '/dev/cciss/c8d0p2' 33935M
  ) ON NODE (3)
  BUFFERPOOL BP32K
  EXTENTSIZE 16
  OVERHEAD 7.0
  TRANSFERRATE 0.52;
COMMIT WORK;

CREATE regular TABLESPACE lineitem_indexes
  IN NODEGROUP all_nodes
  PAGESIZE 32K
  MANAGED BY database
  USING (
    DEVICE '/dev/cciss/c1d0p3' 4547M,
    DEVICE '/dev/cciss/c5d0p3' 4547M
  ) ON NODE (0)
  USING (
    DEVICE '/dev/cciss/c2d0p3' 4547M,
    DEVICE '/dev/cciss/c6d0p3' 4547M
  ) ON NODE (1)
  USING (
    DEVICE '/dev/cciss/c3d0p3' 4547M,
    DEVICE '/dev/cciss/c7d0p3' 4547M
  ) ON NODE (2)
  USING (
    DEVICE '/dev/cciss/c4d0p3' 4547M,
    DEVICE '/dev/cciss/c8d0p3' 4547M
  ) ON NODE (3)
  BUFFERPOOL BP32K
  EXTENTSIZE 16
  OVERHEAD 57.9
  TRANSFERRATE 0.52;
COMMIT WORK;

CREATE regular TABLESPACE other_tables
  IN NODEGROUP all_nodes
  PAGESIZE 32K
  MANAGED BY database
  USING (
    DEVICE '/dev/cciss/c1d0p5' 15639M,
    DEVICE '/dev/cciss/c5d0p5' 15639M
  ) ON NODE (0)
  USING (
    DEVICE '/dev/cciss/c2d0p5' 15639M,
    DEVICE '/dev/cciss/c6d0p5' 15639M
  ) ON NODE (1)
  USING (
    DEVICE '/dev/cciss/c3d0p5' 15639M,
    DEVICE '/dev/cciss/c7d0p5' 15639M
  ) ON NODE (2)
  USING (
    DEVICE '/dev/cciss/c4d0p5' 15639M,
    DEVICE '/dev/cciss/c8d0p5' 15639M
  ) ON NODE (3)
  BUFFERPOOL BP32K
  EXTENTSIZE 16
  OVERHEAD 7.0
  TRANSFERRATE 0.52;
COMMIT WORK;

CREATE regular TABLESPACE other_indexes
  IN NODEGROUP all_nodes
  PAGESIZE 32K
  MANAGED BY database
  USING (
    DEVICE '/dev/cciss/c1d0p6' 2730M,
    DEVICE '/dev/cciss/c5d0p6' 2730M
  ) ON NODE (0)
  USING (
    DEVICE '/dev/cciss/c2d0p6' 2730M,
    DEVICE '/dev/cciss/c6d0p6' 2730M
  ) ON NODE (1)
  USING (
    DEVICE '/dev/cciss/c3d0p6' 2730M,
    DEVICE '/dev/cciss/c7d0p6' 2730M
  ) ON NODE (2)
  USING (
    DEVICE '/dev/cciss/c4d0p6' 2730M,
    DEVICE '/dev/cciss/c8d0p6' 2730M
  ) ON NODE (3)
  BUFFERPOOL BP32K
  EXTENTSIZE 16
  OVERHEAD 7.0
  TRANSFERRATE 0.52;
COMMIT WORK;

CREATE temporary TABLESPACE temp_tables
  PAGESIZE 32K
  MANAGED BY database
  USING (
    DEVICE '/dev/cciss/c1d0p7' 109081M,
    DEVICE '/dev/cciss/c5d0p7' 109081M
  ) ON NODE (0)
  USING (
    DEVICE '/dev/cciss/c2d0p7' 109081M,
    DEVICE '/dev/cciss/c6d0p7' 109081M
  ) ON NODE (1)
  USING (
    DEVICE '/dev/cciss/c3d0p7' 109081M,
    DEVICE '/dev/cciss/c7d0p7' 109081M
  ) ON NODE (2)
  USING (
    DEVICE '/dev/cciss/c4d0p7' 109081M,
    DEVICE '/dev/cciss/c8d0p7' 109081M
  ) ON NODE (3)
  BUFFERPOOL BP32KTEMP
  EXTENTSIZE 16
  OVERHEAD 0.0579
  TRANSFERRATE 0.52;
COMMIT WORK;

drop tablespace userspace1;
commit work;
drop tablespace tempspace1;
commit work;
```

B.10 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");;
```

B.11 create_UFtables

```
---------------------------------------------------------------
-- Create Update Function Tables
---------------------------------------------------------------
CONNECT TO tpcd;
DROP TABLE TPCDTEMP.ORDERS_NEW;
DROP TABLE TPCDTEMP.ORDERS_DEL;
DROP TABLE TPCDTEMP.LINEITEM_NEW;
COMMIT WORK;

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_TABLES
        INDEX IN OTHER_INDEXES
    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_TABLES
        INDEX IN OTHER_INDEXES
    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;
```

B.12 db2nodes.cfg

```
0 bluemoon 0 bluemoon 0
1 bluemoon 1 bluemoon 1
2 bluemoon 2 bluemoon 2
3 bluemoon 3 bluemoon 3
```

B.13 drop_tables

```
---------------------------------------------------------
-- drop
---------------------------------------------------------
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 ;
COMMIT WORK;get database manager configuration;
```

B.14 drop_tablespaces

```
DROP TABLESPACE small_tables;
DROP TABLESPACE lineitem_table;
DROP TABLESPACE lineitem_indexes;
DROP TABLESPACE other_tables;
DROP TABLESPACE other_indexes;
DROP TABLESPACE temp_tables;
DROP TABLESPACE temp2_tables;
COMMIT WORK;
```

B.15 load_db2set.ksh

```
#!/bin/ksh
db2set DB2OPTIONS="-t -v +c"
db2set DB2_EXTENDED_OPTIMIZATION=Y
db2set DB2_ANTIJOIN=Y
db2set DB2BPVARS=/home/tpch/custom/bpvar.cfg
db2set DB2_PARALLEL_IO="*"
db2set DB2_STRIPED_CONTAINERS=ON
db2set DB2LINUXAIO=TRUE
db2set DB2_LGPAGE_BP=OFF
db2set DB2_FORCE_FCM_BP=YES
db2set DB2_SCATTERED_IO=OFF
```

B.16 load_dbcfg.sql

```
UPDATE DB CFG FOR TPCD USING
        APPGROUP_MEM_SZ 1024
        BUFFPAGE 1250
        CHNGPGS_THRESH 15
        DATABASE_MEMORY AUTOMATIC
        DFT_QUERYOPT 7
        DFT_DEGREE ANY
        LOCKLIST 4096
        LOGFILSIZ 2048
        LOGPRIMARY 10
        LOGSECOND 5
        MAXFILOP 1024
        MAXLOCKS 60
        NUM_IOCLEANERS 4
        NUM_IOSERVERS 16
        NUM_FREQVALUES 0
        NUM_QUANTILES 300
        SHEAPTHRES_SHR 0
        SOFTMAX 600
        SORTHEAP 6250
        UTIL_HEAP_SZ 12500
        ;
GET DB CFG FOR TPCD;
```

B.17 load_dbmcfg.sql

```
UPDATE DBM CFG USING
CPUSPEED -1
SHEAPTHRES 62500
MAX_QUERYDEGREE ANY
INTRA_PARALLEL YES
SVCENAME db2c_db2inst1
DIAGLEVEL 3
NUMDB 1
DFT_MON_TIMESTAMP OFF
NUM_POOLAGENTS 8
NUM_INITAGENTS 8
ASLHEAPSZ 15
RQRIOBLK 32767
MAXAGENTS 256
;
GET DBM CFG;
```

B.18 load_tables.ksh

```
#!/bin/ksh
messages=${TPCD_TMP_DIR}
rawdata=${TPCD_INPUT}
custom=${TPCD_DDLPATH}

echo "Load Summary Time: " > ${messages}/loadstatus.out

# Nation and Region are loaded into the current node

db2 connect to tpcd;

echo "Loading Nation at "`date` >> ${messages}/loadstatus.out
db2 "values(current timestamp,'TS*** Load Nation Started');"
db2 "load from ${rawdata}/nation.tbl OF DEL MODIFIED BY
COLDEL|FASTPARSE ANYORDER MESSAGES
${messages}/nation.msg REPLACE INTO TPCD.nation STATISTICS NO
NONRECOVERABLE;"
db2 "commit work;"
db2 "values(current timestamp,'TS*** Load Nation Finished');"

echo "Loading Region at "`date` >> ${messages}/loadstatus.out
db2 "values(current timestamp,'TS*** Load Region Started');"
db2 "load from ${rawdata}/region.tbl OF DEL MODIFIED BY
COLDEL|FASTPARSE ANYORDER MESSAGES
${messages}/region.msg REPLACE INTO TPCD.region STATISTICS NO
NONRECOVERABLE;"
db2 "commit work;"
db2 "values(current timestamp,'TS*** Load Region Finished');"

db2 commit;
db2 terminate;

echo "Loading Partsupp at "`date` >> ${messages}/loadstatus.out
 db2 -tvf ${custom}/loadpartsupp
#
echo "Loading Orders at "`date` >> ${messages}/loadstatus.out
 db2 -tvf ${custom}/loadorders
```

```
#
echo "Loading Lineitem at "`date` >> ${messages}/loadstatus.out
 db2 -tvf ${custom}/loadlineitem
#
echo "Loading Customer at "`date` >> ${messages}/loadstatus.out
 db2 -tvf ${custom}/loadcustomer
#
echo "Loading Part at "`date` >> ${messages}/loadstatus.out
 db2 -tvf ${custom}/loadpart
#
echo "Loading Supplier at "`date` >> ${messages}/loadstatus.out
 db2 -tvf ${custom}/loadsupplier
#

db2 commit;
db2 terminate;

echo "Finished Loading  at "`date` >> ${messages}/loadstatus.out
echo "-------------------------------------------" >> ${messages}/loadstatus.out

echo "Starting Sanity Chequing at  "`date` >> ${messages}/loadstatus.out
db2 connect to tpcd;
db2 "select count_big(*) as lineitem from tpcd.lineitem" >>
${messages}/loadstatus.out
db2 "select count_big(*) as orders from tpcd.orders" >>
${messages}/loadstatus.out
db2 "select count_big(*) as partsupp from tpcd.partsupp" >>
${messages}/loadstatus.out
db2 "select count_big(*) as customer from tpcd.customer" >>
${messages}/loadstatus.out
db2 "select count_big(*) as part from tpcd.part" >>
${messages}/loadstatus.out
db2 "select count_big(*) as supplier from tpcd.supplier" >>
${messages}/loadstatus.out
db2 "select count(*) as nation from tpcd.nation" >>
${messages}/loadstatus.out
db2 "select count(*) as region from tpcd.region" >>
${messages}/loadstatus.out
db2 terminate;
echo "Finish Sanity Chequing at  "`date` >> ${messages}/loadstatus.out
```

B.19 loadcustomer

```
connect to tpcd;
values(current timestamp,'TS*** Load Customer Started');
load from customer.node
	OF DEL MODIFIED BY COLDEL|FASTPARSE
	ANYORDER MESSAGES /tmp/tpch/customer.msg
	REPLACE INTO TPCD.customer NONRECOVERABLE DATA
BUFFER 256 CPU_PARALLELISM 16
	PARTITIONED DB CONFIG MODE LOAD_ONLY
OUTPUT_DBPARTNUMS (0,1,2,3)
	PART_FILE_LOCATION /flatfiles;
commit work;
values(current timestamp,'TS*** Load Customer Finished');
connect reset;
terminate;
```

B.20 loadlineitem

```
connect to tpcd;
values(current timestamp,'TS*** Load LineItem Started');
load from lineitem.node
	OF DEL MODIFIED BY COLDEL|FASTPARSE
	ANYORDER MESSAGES /tmp/tpch/lineitem.msg
	REPLACE INTO TPCD.lineitem NONRECOVERABLE DATA
BUFFER 256 CPU_PARALLELISM 16
	PARTITIONED DB CONFIG MODE LOAD_ONLY
OUTPUT_DBPARTNUMS (0,1,2,3)
	PART_FILE_LOCATION /flatfiles;
commit work;
values(current timestamp,'TS*** Load LineItem Finished');
connect reset;
terminate;
```

B.21 loadorders

```
connect to tpcd;
values(current timestamp,'TS*** Load Orders Started');
load from orders.node
	OF DEL MODIFIED BY COLDEL|FASTPARSE
	ANYORDER MESSAGES /tmp/tpch/orders.msg
	REPLACE INTO TPCD.orders NONRECOVERABLE DATA
BUFFER 256 CPU_PARALLELISM 16
	PARTITIONED DB CONFIG MODE LOAD_ONLY
OUTPUT_DBPARTNUMS (0,1,2,3)
	PART_FILE_LOCATION /flatfiles;
commit work;
values(current timestamp,'TS*** Load Orders Finished');
connect reset;
terminate;
```

B.22 loadpart

```
connect to tpcd;
values(current timestamp,'TS*** Load Part Started');
load from part.node
	OF DEL MODIFIED BY COLDEL|FASTPARSE
	ANYORDER MESSAGES /tmp/tpch/part.msg
	REPLACE INTO TPCD.part NONRECOVERABLE DATA BUFFER
256 CPU_PARALLELISM 16
	PARTITIONED DB CONFIG MODE LOAD_ONLY
OUTPUT_DBPARTNUMS (0,1,2,3)
	PART_FILE_LOCATION /flatfiles;
commit work;
values(current timestamp,'TS*** Load Part Finished');
connect reset;
terminate;
```

B.23 loadpartsupp

```
connect to tpcd;
values(current timestamp,'TS*** Load PartSupp Started');
load from partsupp.node
	OF DEL MODIFIED BY COLDEL|FASTPARSE
	ANYORDER MESSAGES /tmp/tpch/partsupp.msg
	REPLACE INTO TPCD.partsupp NONRECOVERABLE DATA
BUFFER 256 CPU_PARALLELISM 16
	PARTITIONED DB CONFIG MODE LOAD_ONLY
OUTPUT_DBPARTNUMS (0,1,2,3)
	PART_FILE_LOCATION /flatfiles;
commit work;
values(current timestamp,'TS*** Load PartSupp Finished');
connect reset;
terminate;
```

B.24 loadsupplier

```
connect to tpcd;
values(current timestamp,'TS*** Load Supplier Started');
load from /flatfiles/supplier.tbl.sorted
	OF DEL MODIFIED BY COLDEL|FASTPARSE
	ANYORDER MESSAGES /tmp/tpch/supplier.msg
	REPLACE INTO TPCD.supplier NONRECOVERABLE DATA
BUFFER 256 CPU_PARALLELISM 16
	PARTITIONED DB CONFIG MODE PARTITION_AND_LOAD
PARTITIONING_DBPARTNUMS (0,1,2,3);
```

```
commit work;
connect reset;
terminate;
```

B.25 remove_UFtables

```
-----------------------------------------------------------
-- Delete UF Tables
-----------------------------------------------------------
connect to tpcd;
delete from TPCDTEMP.ORDERS_NEW;
commit work;
delete from TPCDTEMP.LINEITEM_NEW;
commit work;
delete from TPCDTEMP.ORDERS_DEL;
commit work;
connect reset;
```

B.26 run_db2set.ksh

```
#!/bin/ksh
db2set DB2OPTIONS="-t -v +c"
db2set DB2_EXTENDED_OPTIMIZATION=Y
db2set DB2_ANTIJOIN=Y
db2set DB2BPVARS=/home/tpch/custom/bpvar.cfg
db2set DB2_FORCE_FCM_BP=YES
db2set DB2_PARALLEL_IO="*"
db2set DB2_SCATTERED_IO=OFF
db2set DB2_LGPAGE_BP=ON
db2set DB2_STRIPED_CONTAINERS=ON
db2set DB2LINUXAIO=true
```

B.27 run_dbcfg

```
UPDATE DB CFG FOR TPCD USING
DBHEAP 40000
SORTHEAP 20480
DATABASE_MEMORY AUTOMATIC
UTIL_HEAP_SZ 20000
SHEAPTHRES_SHR 0
UTIL_HEAP_SZ 20000
DFT_QUERYOPT 7
DFT_DEGREE ANY
NUM_FREQVALUES 0
NUM_QUANTILES 300
LOCKLIST 60000
MAXLOCKS 40
CHNGPGS_THRESH 60
NUM_IOCLEANERS 4
NUM_IOSERVERS 16
MAXFILOP 1024
LOGFILSIZ 16384
LOGPRIMARY 4
LOGSECOND 2
SOFTMAX 100
LOGBUFSZ 2048
MINCOMMIT 1
APPLHEAPSZ 16000
STMTHEAP 40000
STAT_HEAP_SZ 20000
DLCHKTIME 5000
CATALOGCACHE_SZ -1
APP_CTL_HEAP_SZ 2058
PCKCACHESZ -1
MAXAPPLS 40
DFT_PREFETCH_SZ AUTOMATIC
APPGROUP_MEM_SZ 40000
;
UPDATE DB CFG FOR TPCD USING
BUFFPAGE 10240
;
```

B.28 run_dbmcfg.sql

```
UPDATE DBM CFG USING
HEALTH_MON OFF
SHEAPTHRES 320000
MAX_QUERYDEGREE ANY
INTRA_PARALLEL YES
FCM_NUM_BUFFERS 20480
FCM_NUM_RQB AUTOMATIC
NUM_POOLAGENTS 64
NUM_INITAGENTS 4
JAVA_HEAP_SZ 1024
CONN_ELAPSE 10
DFT_MON_TIMESTAMP OFF
CPUSPEED -1
MON_HEAP_SZ 90
MAXAGENTS 256
DIAGLEVEL 0
NOTIFYLEVEL 0
COMM_BANDWIDTH 2.8
INDEXREC RESTART
ASLHEAPSZ 15
RQRIOBLK 32767
NUMDB 1
SVCENAME DB2_tpch
;
```

B.29 runstats

```
values(current timestamp,'TS*** runstats 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;
values(current timestamp,'TS*** runstats done region');
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 supplier');
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 part');
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 partsupp');
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 customer');
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 orders');
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 done lineitem');
values(current timestamp,'TS*** runstats END like');
```

B.30 setlogs.ksh

```
#!/bin/ksh
# Set the new log path for the database
db2_all "<<+0< db2 update db cfg for tpcd using newlogpath
/dev/cciss/c1d2p1 "
db2_all "<<+1< db2 update db cfg for tpcd using newlogpath
/dev/cciss/c2d2p1 "
db2_all "<<+2< db2 update db cfg for tpcd using newlogpath
/dev/cciss/c3d2p1 "
db2_all "<<+3< db2 update db cfg for tpcd using newlogpath
/dev/cciss/c4d2p1 "
```

B.31 test_tables.ksh

```
#echo "Starting Sanity Chequing at  "`date`
db2 connect to tpcd;
db2 "select count_big(*) as lineitem from tpcd.lineitem"
db2 "select count_big(*) as orders from tpcd.orders"
db2 "select count_big(*) as partsupp from tpcd.partsupp"
db2 "select count_big(*) as customer from tpcd.customer"
db2 "select count_big(*) as part from tpcd.part"
db2 "select count_big(*) as supplier from tpcd.supplier"
db2 "select count(*) as nation from tpcd.nation"
db2 "select count(*) as region from tpcd.region"
db2 connect reset
db2 terminate;
```

B.32 tpcd.setup

```
# NOTE: ALL variable defitions must have a comment at the end - haven't
got
#       the getvars script recognizing the uncommented line yet
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/tpch/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/tpch              # path for database (defaults to
home)
TPCD_DDLPATH=/home/tpch/custom       # path for all ddl files and
customized
                                    # scripts (load script), config files,etc
TPCD_BUFFERPOOL_DEF=create_bufferpools.ddl   # name of file with
bufferpool definitions
                                         # and sizes
TPCD_NODEGROUP_DEF=create_nodegroups.ddl     # 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   # ddl file for tablespaces
TPCD_DDL=create_tables.ddl          # ddl file for tables
TPCD_QUAL_TBSP_DDL=create_tablespaces.ddl
              # create_tablespaces_qual.ddl   # ddl file for tablespaces for qual
```

```
TPCD_QUAL_DDL=create_tables.ddl                          #
create_tables.ddl       # ddl file for qualification database
                        # tablespaces and tables should be identical
                        # to regular ddl except container names
TPCD_INDEXDDL=create_indexes.ddl      # ddl file for indexes
TPCD_EXTRAINDEX=no                    # no = no extra indexes
                        # filename = If you want to create some
                        # indices before
                        # the load, and some indices after, then
                        # use this additional file to specify the
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=runstats.ddl            # ddl file for runstats.  If you have
                        # created indices before the load (ie
                        # TPCD_EARLYINDEX=yes), have specified to
                        # gather stats on the load command (either
                        # through your own load script or by using
                        # TPCD_LOADSTATS=yes, AND you have
                        # specified a file for TPCD_EXTRAINDEX
                        # then this runstats file should include
                        # the runstats commands specifically for
                        # the extra indices.
TPCD_RUNSTATSHORT=NULL                # 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/tpch/tpcd/appendix.v2/dbgen    # path name to data
generation code

                        # Parameters used to specify source of
                        # data for load scripts
TPCD_INPUT=/flatfiles                 # NULL - use dbgen generated data  OR
                        # path name -  to the pre-generated
                        #           flat files
                        #
TPCD_QUAL_INPUT=/flatfiles/1gb_qual    # NULL - use dbgen generated
data  OR
                        # path name -  to the pre-generated
                        #           flat files

TPCD_TAILOR_DIR=/home/tpch/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=load_db2set.ksh          # Script that
contains the db2set commands
                        # for the load process Use NULL if not
                        # specified
TPCD_LOAD_CONFIGFILE=load_dbcfg.ddl    #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.ddl    # 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.ddl  # 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.ddl       # 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
                        # used in load script only
TPCD_SORTBUF=4096                     # sortbuf size for LOAD
                        # used in load script only
TPCD_LOAD_PARALLELISM=0               # degree of parallelism to use
on load
                        # 0 = use the "intelligent default" that
                        # the load will chose at run time
                        # used in load script only
TPCD_COPY_DIR=/dev/null               # directory where copy image is
created
                        # on load command CURRENTLY UNUSED
                        # used in load script only
TPCD_FASTPARSE=yes                    # use fastparse on load
                        # used in load script only

                        # Backup and logfile specific parameters follow:
TPCD_BACKUP_DIR=/backup_0             # 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/tpch/logs          # directory where log files stored..
                        # NULL leaves them in the dbpath
TPCD_LOG_DIR_SETUP_SCRIPT=setlogs.ksh  # 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.ksh      # Script that contains the
db2set commands
                        # for the benchmark run.  Use NULL if not
                        # specified
TPCD_CONFIGFILE=run_dbcfg.ddl         # name of configuration file in
ddl path
                        # that will be used for the benchmark run
TPCD_DBM_CONFIG=run_dbmcfg.ddl        # name of config file for
database manager
                        # cfg parms
```

```
TPCD_QUALCONFIGFILE=run_dbcfg.ddl      # name of database cfg file
in ddl path
                        # for qualification database
TPCD_DBM_QUALCONFIG=run_dbmcfg.ddl     # 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 is specified
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=/home/tpch/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=/flatfiles/ufdata            # /home/tpch/ufdata
#/backup_3/ufdata
                        # where to generate/read flat files
                        # for update functions
TPCD_STAGING_TABLE_DDL=create_UFtables.ddl    # script that
contains the ddl for creating
                            # the staging tables if they are used for
                            # the update functions
TPCD_PRELOAD_STAGING_TABLE_SCRIPT=              #
preloadUF.ksh
                                # file that contains the sql for preloading
                                # and gathering stats on sample UF data
                                # Note that the data used is sample data
                                # and is not data from any of the applied
                                # update pairs
TPCD_DELETE_STAGING_TABLE_SQL=remove_UFtables.ddl     # 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=8        # 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_CONCURRENT_DELETES=8        # number of delete chunks
that are run
TPCD_SPLIT_DELETES=256           # number of portions to split the
delete
                        # function into.
                        # this variable is only valid in UNI/SMP
                        # mode.
TPCD_GEN_UPDATEPAIRS=40          # 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_tables.ksh      # 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_tables_qual.ksh          # 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=/home/tpch/sqllib/db2dump  # directory wehre the
db2diag.log can
                              # be found for the durability tests
TPCD_APPEND_ON=no                   # set to no if the cluster indexes are
used
```

B.33 verifytpcdbatch.clp

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

Appendix C: Qualification Query Output

C.1 Qualification Queries

Qualification Query 1

```
---------------------------------------------
Start timestamp 08/24/05 10:10:36.984961

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

-- 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 08/24/05 10:10:42.140118
Query Time =           5.2 secs
---------------------------------------------
```

Qualification Query 2

```
---------------------------------------------
Start timestamp 08/24/05 10:09:48.272581

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

-- 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

... rows deleted ...

         7937.930  Supplier#000009012        ROMANIA
83995  Manufacturer#2            iUiTziH,Ek3i4lwSgunXMgrcTzwdb
29-250-925-9690  blithely bold ideas haggle quickly final, regular request
         7914.450  Supplier#000001013        RUSSIA
125988  Manufacturer#2           riRcntps4KEDtYScjpMIWeYF6mNnR
32-194-698-3365  final, ironic theodolites alongside of the ironic
         7912.910  Supplier#000004211        GERMANY
159180  Manufacturer#5
2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG        17-266-947-7315  final
requests integrate slyly above the silent, even
         7912.910  Supplier#000004211        GERMANY
184210  Manufacturer#4
2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG        17-266-947-7315  final
requests integrate slyly above the silent, even
         7894.560  Supplier#000007981        GERMANY
85472  Manufacturer#4            NSJ96vMROAbeXP                        17-
963-404-3760  regular, even theodolites integrate carefully. bold, special
theodolites are slyly fluffily iron
         7887.080  Supplier#000009792        GERMANY
164759  Manufacturer#3           Y28ITVeYriT3kIGdV2K8fSZ
V2UqT5H1Otz       17-988-938-4296  pending, ironic packages sleep
among the carefully ironic accounts. quickly final accounts
         7871.500  Supplier#000007206        RUSSIA
104695  Manufacturer#1           3w fNCnrVmvJjE95sgWZzvW
32-432-452-7731  furiously dogged pinto beans cajole. bold, express
notornis until the slyly pending
         7852.450  Supplier#000005864        RUSSIA
8363  Manufacturer#4            WCNfBPZeSXh3h,c                      32-
454-883-3821  blithely regular deposits
         7850.660  Supplier#000001518        UNITED KINGDOM
86501  Manufacturer#1            ONda3YJiHKJOC                        33-
730-383-3892  furiously final accounts wake carefully idle requests. even
dolphins wake acc
         7843.520  Supplier#000006683        FRANCE
11680  Manufacturer#4            2Z0JGkiv01Y00oCFwUGfviIbhzCdy
16-464-517-8943  carefully bold accounts doub


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


Stop timestamp 08/24/05 10:09:48.563030
Query Time =            0.3 secs
---------------------------------------------
```

Qualification Query 3

```
---------------------------------------------
Start timestamp 08/24/05 10:10:27.867917

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


-- 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 08/24/05 10:10:30.495184
```

```
Query Time =         2.6 secs
---------------------------------------------
```

Qualification Query 4

```
---------------------------------------------
Start timestamp 08/24/05 10:10:31.443041

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

-- 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 08/24/05 10:10:36.743719
Query Time =         5.3 secs
---------------------------------------------
```

Qualification Query 5

```
---------------------------------------------
Start timestamp 08/24/05 10:10:52.985801

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

-- 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 08/24/05 10:10:58.969659
Query Time =         6.0 secs
---------------------------------------------
```

Qualification Query 6

```
---------------------------------------------
Start timestamp 08/24/05 10:10:00.216203

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

-- 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 08/24/05 10:10:00.258668
Query Time =         0.0 secs
---------------------------------------------
```

Qualification Query 7

```
---------------------------------------------
Start timestamp 08/24/05 10:10:58.969659

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

-- 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 08/24/05 10:11:00.016971
Query Time =            1.0 secs
---------------------------------------------
```

Qualification Query 8

```
---------------------------------------------
Start timestamp 08/24/05 10:10:11.801219

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

-- 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 08/24/05 10:10:17.569744
Query Time =            5.8 secs
---------------------------------------------
```

Qualification Query 9

```
---------------------------------------------
Start timestamp 08/24/05 10:09:48.563030
```

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

-- 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

... rows deleted ...

UNITED STATES                  1994        49296747.183
UNITED STATES                  1993        48029946.801
UNITED STATES                  1992        48671944.498
VIETNAM                        1998        30442736.059
VIETNAM                        1997        50309179.794
VIETNAM                        1996        50488161.410
VIETNAM                        1995        49658284.612
VIETNAM                        1994        50596057.261
VIETNAM                        1993        50953919.152
VIETNAM                        1992        49613838.315


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


Stop timestamp 08/24/05 10:09:59.614012
```

```
Query Time =           11.1 secs
---------------------------------------------
```

Qualification Query 10

```
---------------------------------------------
Start timestamp 08/24/05 10:10:42.140118

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

-- 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#0000057040                734235.246              632.870
JAPAN                    Eioyzjf4pp                        22-895-641-3466
requests sleep blithely about the furiously i
    143347  Customer#0000143347                721002.695
2557.470  EGYPT                    1aReFYv,Kw4                            14-742-
935-3718  fluffily bold excuses haggle finally after the u
     60838  Customer#0000060838                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#0000101998                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#0000125341               633508.086
4983.510  GERMANY                  S29ODD6bceU8QSuuEJznkNaK
17-582-695-5962  quickly express requests wake quickly blithely
     25501  Customer#0000025501               620269.785
7725.040  ETHIOPIA
W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ       15-874-808-6793
quickly special requests sleep evenly among the special deposits. special
deposi
    115831  Customer#0000115831               596423.867
5098.100  FRANCE                   rFeBbEEyk dl
ne7zV5fDrmiq1oK09wV7pxqCgIc  16-715-386-3788  carefully bold
excuses sleep alongside of the thinly idle
     84223  Customer#0000084223               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#0000054289               585603.392
5583.020  IRAN                     vXCxoCsU0Bad5JQI ,oobkZ               20-
834-292-4707  express requests sublate blithely regular requests. regular,
even ideas solve.
     39922  Customer#0000039922               584878.113
7321.110  GERMANY
Zgy4s50l2GKN4pLDPBU8m342gIw6R          17-147-757-8036  even
pinto beans haggle. slyly bold accounts inte
      6226  Customer#0000006226               576783.761           2230.090
UNITED KINGDOM          8gPu8,NPGkfyQQ0hcIYUGPIBWc,ybP5g,
33-657-701-3391  quickly final requests against the regular instructions
wake blithely final instructions. pa
       922  Customer#0000000922               576767.533           3869.250
GERMANY                 Az9RFaut7NkPnc5zSD2PwHgVwr4jRzq
17-945-916-9648  boldly final requests cajole blith
    147946  Customer#0000147946               576455.132
2030.130  ALGERIA                  iANyZHjqhyy7Ajah0pTrYyhJ
10-886-956-3143  furiously even accounts are blithely above the furiousl
    115640  Customer#0000115640               569341.193
6436.100  ARGENTINA                Vtgfia9qI 7EpHgecU1X
11-411-543-4901  final instructions are slyly according to the
     73606  Customer#0000073606               568656.858
1785.670  JAPAN                    xuR0Tro5yChDfOCrjkd2ol               22-
437-653-6966  furiously bold orbits about the furiously busy requests wake
across the furiously quiet theodolites. d
    110246  Customer#0000110246               566842.981
7763.350  VIETNAM                  7KzflgX MDOq7sOkI
31-943-426-9837  dolphins sleep blithely among the slyly final
    142549  Customer#0000142549               563537.237
5085.990  INDONESIA                ChqEoK43OysjdHbtKCp6dKqjNyvvi9
19-955-562-2398  regular, unusual dependencies boost slyly; ironic
attainments nag fluffily into the unusual packages?
    146149  Customer#0000146149               557254.986
1791.550  ROMANIA                  s87fvzFQpU                         29-744-
164-6487  silent, unusual requests detect quickly slyly regul
     52528  Customer#0000052528               556397.351            551.790
ARGENTINA               NFztyTOR10UOJ                        11-208-192-
3205  unusual requests detect. slyly dogged theodolites use slyly. deposit
     23431  Customer#0000023431               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 08/24/05 10:10:47.758797
Query Time =             5.6 secs
---------------------------------------------
```

Qualification Query 11

```
---------------------------------------------
Start timestamp 08/24/05 10:10:36.743719

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

-- 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

... row deleted ...

     113808         7893353.880
      27901         7892952.000
     128820         7892882.720
      25891         7890511.200
     122819         7888881.020
     154731         7888301.330
     101674         7879324.600
      51968         7879102.210
      72073         7877736.110
       5182         7874521.730


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


Stop timestamp 08/24/05 10:10:36.841316
Query Time =            0.1 secs
---------------------------------------------
```

Qualification Query 12

```
---------------------------------------------
Start timestamp 08/24/05 10:11:00.016971

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

-- 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 08/24/05 10:11:05.458527
Query Time =            5.4 secs
---------------------------------------------
```

Qualification Query 13

```
---------------------------------------------
Start timestamp 08/24/05 10:10:25.950564

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

-- 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

... rows deleted ...

        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 08/24/05 10:10:27.867917
Query Time =            1.9 secs
---------------------------------------------
```

Qualification Query 14

```
---------------------------------------------
Start timestamp 08/24/05 10:09:34.532768

---------------------------------------------
--#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 08/24/05 10:09:48.272581
Query Time =           13.7 secs
---------------------------------------------
```

Qualification Query 15

```
---------------------------------------------
Start timestamp 08/24/05 10:10:36.841316

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

-- 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 08/24/05 10:10:36.984961
Query Time =           0.1 secs
---------------------------------------------
```

Qualification Query 16

```
---------------------------------------------
Start timestamp 08/24/05 10:10:31.042168

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

-- 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

... rows deleted ...

Brand#25    LARGE PLATED STEEL                19            3
Brand#32    STANDARD ANODIZED COPPER          23            3
Brand#33    SMALL ANODIZED BRASS               9            3
Brand#35    MEDIUM ANODIZED TIN               19            3
Brand#51    SMALL PLATED BRASS                23            3
Brand#52    MEDIUM BRUSHED BRASS              45            3
```

```
Brand#53    MEDIUM BRUSHED TIN                45          3
Brand#54    ECONOMY POLISHED BRASS             9          3
Brand#55    PROMO PLATED BRASS                19          3
Brand#55    STANDARD PLATED TIN               49          3


Number of rows retrieved is:  18314
---------------------------------------------


Stop timestamp 08/24/05 10:10:31.443041
Query Time =            0.4 secs
---------------------------------------------
```

Qualification Query 17

```
---------------------------------------------
Start timestamp 08/24/05 10:10:00.258668

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

-- 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 08/24/05 10:10:04.747937
Query Time =            4.5 secs
---------------------------------------------
```

Qualification Query 18

```
---------------------------------------------
Start timestamp 08/24/05 10:10:04.747937

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

-- 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

... rows deleted ...

Customer#000112987          112987      4439686  1996-09-17
418161.490              305.000
Customer#000012599           12599      4259524  1998-02-12
415200.610              304.000
Customer#000105410          105410      4478371  1996-03-05
412754.510              302.000
Customer#000149842          149842      5156581  1994-05-30
411329.350              302.000
Customer#000010129           10129      5849444  1994-03-21
409129.850              309.000
Customer#000069904           69904      1742403  1996-10-19
408513.000              305.000
Customer#000017746           17746         6882  1997-04-09
408446.930              303.000
```

```
Customer#0000013072          13072      1481925  1998-03-15
399195.470             301.000
Customer#0000082441          82441       857959  1994-02-07
382579.740             305.000
Customer#0000088703          88703      2995076  1994-01-30
363812.120             302.000


Number of rows retrieved is:     57
---------------------------------------------


Stop timestamp 08/24/05 10:10:11.801219
Query Time =            7.1 secs
---------------------------------------------
```

Qualification Query 19

```
---------------------------------------------
Start timestamp 08/24/05 10:10:47.758797

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

-- 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 08/24/05 10:10:52.985801
Query Time =            5.2 secs
---------------------------------------------
```

Qualification Query 20

```
---------------------------------------------
Start timestamp 08/24/05 10:09:59.614012

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

-- 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

... rows deleted ...
```

```
Supplier#000009753        wLhVEcRmd7PkJF4FBnGK7Z
Supplier#000009796        z,y4Idmr15DOvPUqYG
Supplier#000009799        4wNjXGa4OKWl
Supplier#000009811        E3iuyq7UnZxU7oPZIe2Gu6
Supplier#000009812        APFRMy3lCbgFga53n5t9DxzFPQPgnjrGt32
Supplier#000009862        rJzweWeN58
Supplier#000009868        ROjGgx5gvtkmnUUoeyy7v
Supplier#000009869
ucLqxzrpBTRMewGSM29t0rNTM30g1Tu3Xgg3mKag
Supplier#000009899        7XdpAHrzr1t,UQFZE
Supplier#000009974        7wJ,J5DKcxSU4Kp1cQLpbcAvB5AsvKT


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


Stop timestamp 08/24/05 10:10:00.216203
Query Time =            0.6 secs
---------------------------------------------
```

Qualification Query 21

```
---------------------------------------------
Start timestamp 08/24/05 10:10:17.569744

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

-- 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

... rows deleted ...

Supplier#000000673              12
Supplier#000000762              12
Supplier#000000811              12
Supplier#000000821              12
Supplier#000001337              12
Supplier#000001916              12
Supplier#000001925              12
Supplier#000002039              12
Supplier#000002357              12
Supplier#000002483              12


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


Stop timestamp 08/24/05 10:10:25.950564
Query Time =            8.4 secs
---------------------------------------------
```

Qualification Query 22

```
---------------------------------------------
Start timestamp 08/24/05 10:10:30.495184

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

-- 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 08/24/05 10:10:31.042168
Query Time =            0.5 secs
---------------------------------------------
```

C.2 First 10 Rows of Test Database Tables

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

R_REGIONKEY R_NAME                    R_COMMENT
----------- ------------------------- -----------------------------------------------------
------------------------------------------------------------------------------------------
--------
          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
          0 AFRICA                    special Tiresias about the furiously even
dolphins are furi
          1 AMERICA                   even, ironic theodolites according to the
bold platelets wa

  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
----------- ------------------------------------------------------- -----------------------
-- ---------- ------------------------- ----------- ----------- ------------------------ ---
--------------------
      22075 linen royal puff aquamarine mint                       Manufacturer#1
Brand#11   LARGE ANODIZED COPPER             14 LG BOX
+9.97070000000000E+002 caref
      22132 beige chiffon azure red sandy                          Manufacturer#1
Brand#11   PROMO BURNISHED TIN               34 SM PACK
+1.05413000000000E+003 slyly expres
      22149 sky sienna goldenrod lemon brown
Manufacturer#1            Brand#11   PROMO BRUSHED BRASS
20 MED PACK      +1.07114000000000E+003 slyly pendin
      22381 rose pink lace white deep                              Manufacturer#1
Brand#11   MEDIUM ANODIZED TIN               50 MED BAG
+1.30338000000000E+003 final dependencies w
      22651 red dim coral ghost linen                              Manufacturer#1
Brand#11   PROMO ANODIZED BRASS              19 MED JAR
+1.57365000000000E+003 regular
      22677 thistle brown rose peach sienna                        Manufacturer#1
Brand#11   LARGE POLISHED STEEL              24 SM CAN
+1.59967000000000E+003 ironic foxes de
      22728 burnished cream violet maroon seashell
Manufacturer#1            Brand#11   PROMO PLATED TIN                  47
SM PKG        +1.65072000000000E+003 carefully spec
      22871 salmon dark black moccasin ghost
Manufacturer#1            Brand#11   SMALL BURNISHED STEEL
13 WRAP BAG      +1.79387000000000E+003 regular deposit
      22924 light magenta cyan wheat cornsilk                      Manufacturer#1
Brand#11   MEDIUM BRUSHED COPPER             34 SM DRUM
+1.84692000000000E+003 even
      23044 orange coral forest dim chartreuse                     Manufacturer#1
Brand#11   PROMO BURNISHED COPPER            31 MED JAR
+9.67040000000000E+002 brave, regu

  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
----------- ------------------------- ---------------------------------------- ----------- -
--------------- ------------------------ --------------------------------------------------
--------------------------------------------------
      11471 Supplier#000011471        PvzycPzOn6uR, lYA
0 10-784-325-8124   +8.22740000000000E+003 slyly final ideas cajole
blithely carefully regular waters. final requests serve quietl
```

```
     11620 Supplier#000011620       48 mTL21EIBATgJu
0 10-122-491-7701   +4.77407000000000E+003 pending foxes sleep. furi
     11701 Supplier#000011701
FqBkY76yPHDIKAnbdOkUJN5j5ji6m                    0 10-530-944-8919
+7.01997000000000E+003 silent pinto beans haggle. quickly special
packages alongside of the blit
     11739 Supplier#000011739       rc2DRfObobTTS2X1NTym
flIzVXDAl29fAf              0 10-560-912-7542
+7.98751000000000E+003 final, express foxes use furiously furiously
unusual platelets.
     11764 Supplier#000011764
lxb44yjpIANfngPoWL8Qm5TDggO1wuJ                  0 10-464-387-7184
+7.74000000000000E+000 enticingly even deposits unwind ruthlessly
unusual foxes; forges wake among the requests
     11805 Supplier#000011805       O4ubOvfW69ZxrKUYjQ
0 10-278-920-6709   -3.72710000000000E+002 theodolites breach
furiously bold ideas. fluffily final deposit
     11940 Supplier#000011940       t7DJvueXDdZ0VvVdbiXg5vT87p
0 10-830-398-6128   +5.28746000000000E+003 regular requests haggle
about the furiously special dependencies; express, special accou
     12040 Supplier#000012040       vxUz5Io4nm
0 10-406-956-9346   +3.20750000000000E+003 slyly ironic theodolites
sleep. brave pinto beans alongside of the
     12110 Supplier#000012110       jHaqshAOIrD5X4qiV
0 10-176-152-6732   +7.71829000000000E+003 instructions are against
the s
     12133 Supplier#000012133       keMNotiysX5OpLErTO6B22
0 10-139-240-6961   +7.70059000000000E+003 furiously express ideas
grow blithely even, fluffy deposits. fluffily pending dolphins sleep sl

  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
----------- ------------------------- ---------------------------------------- ----------- -
-------------- ------------------------ ------------ ---------------------------------------
-----------------------------------------------------------------------------
         76 Customer#0000000076       m3sbCvjMOHyaOofH,e UkGPtqc4
0 10-349-718-3044   +5.74533000000000E+003 FURNITURE    blithely
final theodolites across the furiously stealthy attainments haggle sl
         80 Customer#0000000080       K,vtXp8qYB
0 10-267-172-7101   +7.38353000000000E+003 FURNITURE    bold
asymptotes about the express, express asymptotes wake flu
        643 Customer#0000000643       9T 2avhfyF PQ
0 10-978-597-2747   +5.18470000000000E+003 FURNITURE    quickly
even instructions sleep slyly around the furiously special instructions.
quickly silent deposits integrate c
        659 Customer#0000000659       ThR9miOedPuwVEZyz 3MMjHPwB
0 10-834-287-1466   +5.29768000000000E+003 HOUSEHOLD    final
requests integrate carefully above the carefully ironic foxes. furiously bold
re
        895 Customer#0000000895       MDaJr8ekGTS79bS7CH8f1WgWPU
0 10-933-819-2037   +9.04430000000000E+002 AUTOMOBILE   even,
unusual requests wake furiously fluffi
        915 Customer#0000000915       mtGezp1BRzcfPVl,1,G8Wl
0 10-452-398-2445   +3.77653000000000E+003 AUTOMOBILE   blithely
bold gifts cajole furiously fur
        951 Customer#0000000951       PnC4Xlds,v
0 10-813-916-8297   +7.49947000000000E+003 FURNITURE    furiously
ironic instructions unwind regular, express foxes. even, ironic dependencies
against the regular,
        968 Customer#0000000968       eu 5FA1WHs9jq0pcdlVVA
0 10-470-740-2657   +8.92197000000000E+003 BUILDING     quickly
regular requests detect. slyly pending packages cajole quickly after the
fluffily silent instruct
       1400 Customer#0000001400       BuouRkR7J f
0 10-217-180-5310   +2.43273000000000E+003 BUILDING     furiously
express platelets use carefully quickly regular packages. spec
       1430 Customer#0000001430       mv 9MEDwd8yPeQj7N
0 10-209-317-6929   -9.20400000000000E+002 BUILDING     carefully
even instructions wake blithely express, bold asymptotes. final deposits
sleep accor

  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
----------- ----------- ------------- ------------------------ ----------- --------------- -
-------------- ------------- ------------------------------------------------------------
------------------
  153131461    42646457 F             +7.66065100000000E+004 01/01/1992
5-LOW           Clerk#000265093              0 foxes sleep. slyly express
dependencies wake along the quickly
  153149030    31506916 F             +3.24007560000000E+005 01/01/1992
4-NOT SPECIFIED Clerk#000180200              0 slyly unusual foxes nag
above the
  153179811    39037042 F             +2.18581550000000E+005 01/01/1992
4-NOT SPECIFIED Clerk#000231900              0 quickly regular accounts
nag carefully decoys. carefully ironic excuses
  153243716    41836229 F             +2.62967510000000E+005 01/01/1992
3-MEDIUM        Clerk#000246508              0 deposits cajole across the
accounts. carefully bold excuses wake at the fluffi
```

```
  153248930    15303767 F              +4.75660200000000E+004 01/01/1992
3-MEDIUM       Clerk#000277411              0 special ideas try to are
carefully final accounts. ironic dolphins aff
  153292161    19659569 F              +2.53844750000000E+005 01/01/1992
5-LOW          Clerk#000276961              0 slyly bold epitaphs are.
furiously unusua
  153313509    34257410 F              +1.01972200000000E+004 01/01/1992
5-LOW          Clerk#000136469              0 regular requests kindle blithely-
- pending escapades near
  153352225    22509920 F              +1.36812540000000E+005 01/01/1992
5-LOW          Clerk#000268124              0 final, even platelets alongs
  153357479    35964910 F              +9.86039500000000E+004 01/01/1992
4-NOT SPECIFIED Clerk#000025022             0 final pearls accord
  153367968    43104979 F              +1.50425170000000E+005 01/01/1992
1-URGENT       Clerk#000032224              0 furiously final accounts
across the thinly silent the

  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
----------- ----------- ----------- ------------ ------------------------ ------------------
------ ------------------------ ------------------------ ------------ ------------ ---------
- ------------ ------------- ------------------------- ---------- --------------------------
------------------
     414725    16565088     1565089            1   +2.30000000000000E+001
+2.65019800000000E+004   +0.00000000000000E+000
+0.00000000000000E+000 R            F            01/03/1992 02/02/1992
01/05/1992    NONE                       AIR        even excuses use closely
express pa
     600929    10856993     1856994            4   +4.40000000000000E+001
+8.57758000000000E+004   +9.00000000000000E-002
+2.00000000000000E-002 A            F            01/03/1992 03/21/1992
01/29/1992    NONE                       REG AIR    blithely special requests
about
     646854    50512284     1012317            4   +1.30000000000000E+001
+1.68188800000000E+004   +1.00000000000000E-001
+1.00000000000000E-002 A            F            01/03/1992 03/25/1992
01/19/1992    DELIVER IN PERSON          REG AIR    furiously ironic
patterns cajole ca
    1302822    30261977     2512008            2   +2.00000000000000E+000
+3.87492000000000E+003   +1.00000000000000E-002
+6.00000000000000E-002 R            F            01/03/1992 03/28/1992
01/10/1992    TAKE BACK RETURN           TRUCK      blithely even
accounts are slyly accounts.
    1804929    25503501     2253510            6   +5.00000000000000E+001
+7.51615000000000E+004   +3.00000000000000E-002
+4.00000000000000E-002 R            F            01/03/1992 03/10/1992
01/08/1992    DELIVER IN PERSON          RAIL       blithely even
asymptotes h
    1834631    30310793      310794            2   +2.10000000000000E+001
+3.78478800000000E+004   +3.00000000000000E-002
+0.00000000000000E+000 R            F            01/03/1992 02/10/1992
01/26/1992    DELIVER IN PERSON          MAIL       unusual accounts a
    2543170    53852746      602764            3   +2.00000000000000E+001
+3.39210000000000E+004   +7.00000000000000E-002
+6.00000000000000E-002 R            F            01/03/1992 02/17/1992
01/18/1992    DELIVER IN PERSON          FOB        furiously even exc
    2789249    34059801     1059802            6   +3.50000000000000E+001
+6.15685000000000E+004   +1.00000000000000E-001
+2.00000000000000E-002 A            F            01/03/1992 02/01/1992
01/28/1992    COLLECT COD                REG AIR    slyly regul
    3406183    32120910     2120911            2   +2.50000000000000E+001
+4.82327500000000E+004   +0.00000000000000E+000
+1.00000000000000E-002 A            F            01/03/1992 03/22/1992
01/25/1992    COLLECT COD                MAIL       pains are among the
final, bol
    3589984    38322819     1572856            1   +1.00000000000000E+000
+1.83990000000000E+003   +2.00000000000000E-002
+7.00000000000000E-002 A            F            01/03/1992 02/07/1992
01/20/1992    NONE                       SHIP       blithely silent ideas sleep
blithely

  10 record(s) selected.
```

C.3 Query Substitution Parameters

```
Power stream    Seed = 822210303
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 822210303 as a seed to the RNG
Q1  DELTA       99
Q2  SIZE        46
    TYPE        TIN
    REGION      EUROPE
Q3  SEGMENT     MACHINERY
    DATE        1995-03-22
Q4  DATE        1996-06-01
Q5  REGION      MIDDLE EAST
    DATE        1993-01-01
Q6  DATE        1993-01-01
    DISCOUNT    0.06
    QUANTITY    25
Q7  NATION1     JORDAN
    NATION2     MOZAMBIQUE
Q8  NATION      MOZAMBIQUE
    REGION      AFRICA
    TYPE        SMALL ANODIZED BRASS
Q9  COLOR       coral
Q10 DATE        1993-10-01
Q11 NATION      MOZAMBIQUE
    FRACTION    0.0000003333
Q12 SHIPMODE1   REG AIR
    SHIPMODE2   TRUCK
    DATE        1993-01-01
Q13 WORD1       pending
    WORD2       packages
Q14 DATE        1993-10-01
Q15 DATE        1995-04-01
Q16 BRAND       Brand#31
    TYPE        MEDIUM BURNISHED
    SIZE1       19
    SIZE2       28
    SIZE3       20
    SIZE4       31
    SIZE5       24
    SIZE6       41
    SIZE7       49
    SIZE8       17
Q17 BRAND       Brand#12
    CONTAINER   LG CAN
Q18 QUANTITY    313
Q19 BRAND1      Brand#23
    BRAND2      Brand#55
    BRAND3      Brand#23
    QUANTITY1   9
    QUANTITY2   17
    QUANTITY3   21
Q20 COLOUR      midnight
    DATE        1997-01-01
    NATION      UNITED STATES
Q21 NATION      UNITED KINGDOM
Q22 I1          34
    I2          26
    I3          30
```

```
   I4         14
   I5         12
   I6         31
   I7         11

Throughput Stream = 1    Seed = 822210304
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 822210304 as a seed to the RNG
Q1  DELTA       107
Q2  SIZE        34
   TYPE        COPPER
   REGION      AMERICA
Q3  SEGMENT     BUILDING
   DATE        1995-03-08
Q4  DATE        1994-03-01
Q5  REGION      AFRICA
   DATE        1993-01-01
Q6  DATE        1993-01-01
   DISCOUNT    0.03
   QUANTITY    24
Q7  NATION1     ETHIOPIA
   NATION2     INDIA
Q8  NATION      INDIA
   REGION      ASIA
   TYPE        STANDARD POLISHED BRASS
Q9  COLOR       brown
Q10 DATE        1994-07-01
Q11 NATION      EGYPT
   FRACTION    0.0000003333
Q12 SHIPMODE1   FOB
   SHIPMODE2   TRUCK
   DATE        1994-01-01
Q13 WORD1       unusual
   WORD2       packages
Q14 DATE        1994-02-01
Q15 DATE        1997-11-01
Q16 BRAND       Brand#11
   TYPE        ECONOMY PLATED
   SIZE1       22
   SIZE2       20
   SIZE3       31
   SIZE4       50
   SIZE5       2
   SIZE6       25
   SIZE7       4
   SIZE8       40
Q17 BRAND       Brand#14
   CONTAINER   MED BOX
Q18 QUANTITY    315
Q19 BRAND1      Brand#35
   BRAND2      Brand#43
   BRAND3      Brand#12
   QUANTITY1   5
   QUANTITY2   18
   QUANTITY3   28
Q20 COLOUR      wheat
   DATE        1996-01-01
   NATION      KENYA
Q21 NATION      MOROCCO
Q22 I1          12
   I2         14
   I3         23
   I4         20
   I5         29
   I6         28
   I7         21

Throughput Stream = 2    Seed = 822210305
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 822210305 as a seed to the RNG
Q1  DELTA       115
Q2  SIZE        22
   TYPE        STEEL
   REGION      EUROPE
Q3  SEGMENT     MACHINERY
   DATE        1995-03-24
Q4  DATE        1996-10-01
Q5  REGION      AMERICA
   DATE        1994-01-01
Q6  DATE        1994-01-01
   DISCOUNT    0.09
   QUANTITY    24
Q7  NATION1     RUSSIA
   NATION2     ALGERIA
Q8  NATION      ALGERIA
   REGION      AFRICA
   TYPE        STANDARD BURNISHED BRASS
Q9  COLOR       beige
Q10 DATE        1993-04-01
Q11 NATION      PERU
   FRACTION    0.0000003333
Q12 SHIPMODE1   MAIL
   SHIPMODE2   TRUCK
   DATE        1994-01-01
Q13 WORD1       unusual
   WORD2       requests
Q14 DATE        1994-05-01
Q15 DATE        1995-08-01
Q16 BRAND       Brand#41
   TYPE        STANDARD BRUSHED
   SIZE1       13
   SIZE2       4
   SIZE3       15
   SIZE4       23
   SIZE5       32
   SIZE6       7
   SIZE7       47
   SIZE8       39
Q17 BRAND       Brand#11
   CONTAINER   MED JAR
Q18 QUANTITY    312
Q19 BRAND1      Brand#32
   BRAND2      Brand#21
   BRAND3      Brand#11
   QUANTITY1   10
   QUANTITY2   19
   QUANTITY3   24
Q20 COLOUR      hot
   DATE        1994-01-01
   NATION      EGYPT
Q21 NATION      GERMANY
Q22 I1          26
   I2         19
   I3         13
   I4         33
   I5         17
   I6         11
   I7         12

Throughput Stream = 3    Seed = 822210306
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 822210306 as a seed to the RNG
Q1  DELTA       62
Q2  SIZE        9
   TYPE        BRASS
   REGION      AMERICA
Q3  SEGMENT     BUILDING
   DATE        1995-03-10
Q4  DATE        1994-07-01
Q5  REGION      ASIA
```

```
   DATE        1994-01-01
Q6  DATE        1994-01-01
   DISCOUNT     0.06
   QUANTITY     25
Q7  NATION1      KENYA
   NATION2      PERU
Q8  NATION       PERU
   REGION       AMERICA
   TYPE        PROMO BRUSHED STEEL
Q9  COLOR       white
Q10 DATE        1994-01-01
Q11 NATION       ETHIOPIA
   FRACTION     0.0000003333
Q12 SHIPMODE1    TRUCK
   SHIPMODE2    MAIL
   DATE        1994-01-01
Q13 WORD1       unusual
   WORD2       requests
Q14 DATE        1994-08-01
Q15 DATE        1993-04-01
Q16 BRAND       Brand#31
   TYPE        LARGE ANODIZED
   SIZE1       20
   SIZE2       41
   SIZE3       39
   SIZE4       27
   SIZE5       9
   SIZE6       19
   SIZE7       23
   SIZE8       6
Q17 BRAND       Brand#13
   CONTAINER    MED CAN
Q18 QUANTITY     314
Q19 BRAND1      Brand#35
   BRAND2      Brand#54
   BRAND3      Brand#15
   QUANTITY1    5
   QUANTITY2    20
   QUANTITY3    21
Q20 COLOUR      salmon
   DATE        1997-01-01
   NATION      CHINA
Q21 NATION       UNITED STATES
Q22 I1          26
   I2          33
   I3          25
   I4          21
   I5          12
   I6          20
   I7          30

Throughput Stream = 4    Seed = 822210307
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 822210307 as a seed to the RNG
Q1  DELTA       70
Q2  SIZE        47
   TYPE        NICKEL
   REGION      MIDDLE EAST
Q3  SEGMENT     HOUSEHOLD
   DATE        1995-03-26
Q4  DATE        1997-02-01
Q5  REGION      MIDDLE EAST
   DATE        1994-01-01
Q6  DATE        1994-01-01
   DISCOUNT     0.04
   QUANTITY     24
Q7  NATION1      FRANCE
   NATION2      INDONESIA
Q8  NATION       INDONESIA
   REGION      ASIA
```

```
   TYPE        PROMO PLATED STEEL
Q9  COLOR       tan
Q10 DATE        1994-10-01
Q11 NATION       CHINA
   FRACTION     0.0000003333
Q12 SHIPMODE1    RAIL
   SHIPMODE2    MAIL
   DATE        1994-01-01
Q13 WORD1       unusual
   WORD2       requests
Q14 DATE        1994-11-01
Q15 DATE        1995-11-01
Q16 BRAND       Brand#11
   TYPE        PROMO PLATED
   SIZE1       15
   SIZE2       12
   SIZE3       34
   SIZE4       8
   SIZE5       49
   SIZE6       3
   SIZE7       27
   SIZE8       17
Q17 BRAND       Brand#15
   CONTAINER    JUMBO BOX
Q18 QUANTITY     315
Q19 BRAND1      Brand#42
   BRAND2      Brand#42
   BRAND3      Brand#55
   QUANTITY1    10
   QUANTITY2    10
   QUANTITY3    28
Q20 COLOUR      cyan
   DATE        1996-01-01
   NATION      INDIA
Q21 NATION       MOZAMBIQUE
Q22 I1          26
   I2          14
   I3          30
   I4          17
   I5          34
   I6          19
   I7          32

Throughput Stream = 5    Seed = 822210308
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 822210308 as a seed to the RNG
Q1  DELTA       78
Q2  SIZE        35
   TYPE        COPPER
   REGION      AMERICA
Q3  SEGMENT     BUILDING
   DATE        1995-03-12
Q4  DATE        1994-11-01
Q5  REGION      AFRICA
   DATE        1994-01-01
Q6  DATE        1994-01-01
   DISCOUNT     0.09
   QUANTITY     24
Q7  NATION1      UNITED KINGDOM
   NATION2      ARGENTINA
Q8  NATION       ARGENTINA
   REGION      AMERICA
   TYPE        PROMO ANODIZED STEEL
Q9  COLOR       sky
Q10 DATE        1993-08-01
Q11 NATION       FRANCE
   FRACTION     0.0000003333
Q12 SHIPMODE1    AIR
   SHIPMODE2    MAIL
   DATE        1995-01-01
```

```
Q13 WORD1       unusual
    WORD2       requests
Q14 DATE        1995-02-01
Q15 DATE        1993-08-01
Q16 BRAND       Brand#41
    TYPE        SMALL POLISHED
    SIZE1       7
    SIZE2       16
    SIZE3       18
    SIZE4       47
    SIZE5       46
    SIZE6       28
    SIZE7       26
    SIZE8       8
Q17 BRAND       Brand#12
    CONTAINER   JUMBO JAR
Q18 QUANTITY    313
Q19 BRAND1      Brand#44
    BRAND2      Brand#25
    BRAND3      Brand#54
    QUANTITY1   6
    QUANTITY2   11
    QUANTITY3   24
Q20 COLOUR      orchid
    DATE        1994-01-01
    NATION      UNITED KINGDOM
Q21 NATION      INDIA
Q22 I1          11
    I2          32
    I3          27
    I4          14
    I5          23
    I6          10
    I7          18

Throughput Stream = 6    Seed = 822210309
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 822210309 as a seed to the RNG
Q1  DELTA       86
Q2  SIZE        23
    TYPE        STEEL
    REGION      MIDDLE EAST
Q3  SEGMENT     HOUSEHOLD
    DATE        1995-03-28
Q4  DATE        1997-06-01
Q5  REGION      AMERICA
    DATE        1995-01-01
Q6  DATE        1995-01-01
    DISCOUNT    0.07
    QUANTITY    25
Q7  NATION1     MOROCCO
    NATION2     CHINA
Q8  NATION      CHINA
    REGION      ASIA
    TYPE        ECONOMY POLISHED STEEL
Q9  COLOR       royal
Q10 DATE        1994-05-01
Q11 NATION      ROMANIA
    FRACTION    0.0000003333
Q12 SHIPMODE1   SHIP
    SHIPMODE2   MAIL
    DATE        1995-01-01
Q13 WORD1       unusual
    WORD2       requests
Q14 DATE        1995-06-01
Q15 DATE        1996-03-01
Q16 BRAND       Brand#31
    TYPE        ECONOMY ANODIZED
    SIZE1       14
    SIZE2       4
    SIZE3       30
    SIZE4       26
    SIZE5       42
    SIZE6       31
    SIZE7       45
    SIZE8       13
Q17 BRAND       Brand#14
    CONTAINER   JUMBO CAN
Q18 QUANTITY    314
Q19 BRAND1      Brand#41
    BRAND2      Brand#13
    BRAND3      Brand#53
    QUANTITY1   1
    QUANTITY2   12
    QUANTITY3   20
Q20 COLOUR      bisque
    DATE        1993-01-01
    NATION      JAPAN
Q21 NATION      ARGENTINA
Q22 I1          20
    I2          15
    I3          10
    I4          30
    I5          16
    I6          13
    I7          33
```

Appendix D: Implementation Specific Layer and Driver Source Code

D.1 load_line_uf

```
#!/bin/ksh
RFpair=$1;
db2 connect to tpcd
db2 "load from lineitem.tbl.u$RFpair of del modified by coldel| fastparse messages /dev/null replace into TPCDTEMP.LINEITEM_new nonrecoverable partitioned db config mode load_only part_file_location /backup_3/ufdata;"
db2 commit;
db2 connect reset
db2 terminate
```

D.2 load_orders_uf

```
#!/bin/ksh
RFpair=$1;
db2 connect to tpcd
db2 "load from orders.tbl.u$RFpair of del modified by coldel| fastparse messages /dev/null replace into TPCDTEMP.ORDERS_new nonrecoverable partitioned db config mode load_only part_file_location /backup_3/ufdata; "
db2 commit;
db2 connect reset
db2 terminate
```

D.3 load_update

```
#!/usr/bin/perl
# usage load_update updatepair function inlistmax
# where updatepair ($1) is the number of update pair to load
#       function ($2) is 1 = running INSERTs
#                        2 = running DELETEs
#               inlistmax ($3) is the maximum number of keys to delete when running UF2

($myName = $0) =~ s@.*/@@; $usage="
Usage: load_update updatepair function inlistmax
    updatepair = number of update pair to load
    function   = UF to implement : 1 for INSERT, 2 for DELETE
    inlistmax  = max num of keys to delete when running UF2\n";

die $usage if (@ARGV < 2);

$updatepair = $ARGV[0];
$function = $ARGV[1];

if (@ARGV > 2) {
  $inlistmax = $ARGV[2];
}

# 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";

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

$dbname=$ENV{"TPCD_DBNAME"};
$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$uftemp=$ENV{"TPCD_UFTEMP"};
$delim=$ENV{"TPCD_PATH_DELIM"};
$split_updates=$ENV{"TPCD_SPLIT_UPDATES"};
$concurrentload=$ENV{"TPCD_CONCURRENT_INSERTS_LOAD"};
$split_deletes=$ENV{"TPCD_SPLIT_DELETES"};
$uftemppath=$ENV{"TPCD_UFTEMPPATH"};
$platform=$ENV{"TPCD_PLATFORM"};
$flatfilepath=$ENV{"TPCD_FLATFILES"};
$tempdir=$ENV{"TPCD_TMP_DIR"};
$physnode=$ENV{"TPCD_PHYS_NODE"};
$lnperpn=$ENV{"TPCD_LN_PER_PN"};
$ordTblName=$ENV{"TPCD_ORDTBL_NAME"};

# check a log file of the load command to determine if the load is successful
sub checkFileSuccess {
  $read = -1;
  $loaded = -1;
  $file = $_[0];
  open(FILE, $file);
  while (<FILE>)
  {
    # get the number of rows read
    if (/^Number of rows read \s+ = (\d+)/)
    {
      $read=$1;
    }
    # get the number of rows loaded
    elsif (/^Number of rows loaded \s+ = (\d+)/)
    {
      $loaded=$1;
    }
  }
  close(FILE);
  # if either 1 of the 2 lines is absent or the number of rows read and loaded
  # are not the same, the load is not successful
  if (($read == -1) || ($loaded == -1) || ($read != $loaded ))
  {
    0;
  }
  else
  {
    1;
  }
}

sub checkInsertResults {
  $valid = 1;
  $maxnode = $physnode * $lnperpn - 1;

  $logfilename = "$tempdir${delim}UF1.log";
  $s = "echo > $logfilename";
  system($s);

  for ($loopa = 0; $loopa < $split_updates; $loopa ++) {
    for ($loopb = 0, $pad = "00000"; $loopb <= $maxnode; $loopb ++, $pad
++) {
      $filename =
"$tempdir${delim}loaduf1.lineitem.u${updatepair}.$pad.$loopa";
      if (!&checkFileSuccess($filename)) {
        $valid = 0;
      }
      $s = "cat $filename >> $logfilename";
      system($s);

      $filename =
"$tempdir${delim}loaduf1.order.u${updatepair}.$pad.$loopa";
      if (!&checkFileSuccess($filename)) {
        $valid = 0;
      }
      $s = "cat $filename >> $logfilename";
      system($s);
    }
  }
  $valid;
}

sub checkDeleteResults {
  open(FILE, "$tempdir${delim}UF2.log");
  $count = 0;
  while (<FILE>) {
    if (/completed ok/) {
      $count ++;
    }
  }
  close(FILE);
  if ($count == ( $physnode * $lnperpn )) {
    1;
  }
  else {
    0;
  }
}


# call db2_all to load from UF1 flatfiles into temporary tables for each
# parallel stream

sub runInsertChild {
  $start=$_[0];
  # first set RAH variable
  system("export
RAHOSTFILE=\$HOME${delim}sqllib${delim}db2nodes.cfg");

  # header of db2_all command
  $s = "RAHBUFNAME='rahout.$$' db2_all '||\"']}typeset -i ln=##;typeset -
Z5 LN3=\$ln;cd $tempdir;db2 connect to $dbname;";
  $ldinc= $split_updates / $concurrentload;
  for ($i = 0, $seq = $_[0]; $i < $ldinc; $i++, $seq++ )
  {
      # load from lineitem table
      $s .= "str=\"db2 \\\"load from
$flatfilepath${delim}lineitem.tbl.u${updatepair}.\${LN3}.$seq of del
modified by coldel| fastparse messages
${tempdir}${delim}line.msg.u${updatepair}.\${LN3}.$seq remote file
uf1_line$seq replace into TPCDTEMP.LINEITEM_$seq nonrecoverable
data buffer 500 cpu_parallelism 1\\\"
>loaduf${function}.lineitem.u${updatepair}.\${LN3}.$seq 2>&1\";print --
\"\$str\";eval \"\$str\";";

      # load from orders table
      $s .= "str=\"db2 \\\"load from
$flatfilepath${delim}${ordTblName}.tbl.u${updatepair}.\${LN3}.$seq of
del modified by coldel| fastparse messages
${tempdir}${delim}ord.msg.u${updatepair}.\${LN3}.$seq remote file
uf1_order$seq replace into TPCDTEMP.ORDERS_$seq nonrecoverable data
buffer 500 cpu_parallelism 1\\\"
>loaduf${function}.order.u${updatepair}.\${LN3}.$seq 2>&1\";print --
\"\$str\";eval \"\$str\";";

      # footer of db2_all command
      $s .= "db2 commit;";
  }
  $s .= "db2 connect reset;db2 terminate'";

  system("$s");
}

# forks off (# of parallel stream) children to do runInsertChild
sub runInsert {
  print "parent waiting for children ...\n";
  $ldincr= $split_updates / $concurrentload;
  for ($j = 0; $j < $split_updates; $j+= $ldincr) {
    if (($pid = fork) == 0) {
      &runInsertChild($j);
      exit;
    }
  }
  for ($j = 0; $j < $split_updates; $j+= $ldincr ) {
    wait;
  }
  print "all children have returned.\n";
#  &checkInsertResults;
}


# call db2_all to delete keys from each logical node
sub runDelete {

  # setup RAH variable
  system("export
RAHOSTFILE=\$HOME${delim}sqllib${delim}db2nodes.cfg");

  # construct db2_all command
  $s = "RAHBUFNAME='rahout.$$' RAHSLEEPTIME=86400
RAHWAITTIME=0 db2_all  '||\"']}export
TPCD_FLATFILES=$flatfilepath;export
TPCD_TMP_DIR=$tempdir;typeset -i ln=##;";
  for ( $numChunk = 0; $numChunk < $split_deletes; $numChunk++ )
  {
      $s .= "$auditDir${delim}auditruns${delim}tpcdbatch -z -d $dbname -i
$updatepair -j 2 -k \$ln -m $inlistmax -x $numChunk & ";
  }
  $s .= "'";

  system("$s > $tempdir${delim}UF2.log");
  &checkDeleteResults;
}

if ( $platform eq 'nt' )
{
    die "not implemented on nt yet!\n";
}
else
{
    if ( $function == 1 ) {
        $return = &runInsert;
```

```
   }
   elsif ( $function == 2 ) {
      $return = &runDelete;
   }
   else {
      die "unknown function\n";
   }
}
if ($return == 0) {
   die("error occurred in load_update\n");
}
1;
```

D.4 Makefile

```
##############################################################
#######
# MAKEFILE for tpcdbatch program
# Enter the Following:
#
#      make tpcdbatch     -- makes tpcdbatch
#
#      make cleanup       -- removes builds from tpcdbatch program
#
#  NOTE:  You must have the TPCD_DBNAME environment variable set or
#         this will not work, I'm trying to figure out a way to see
#         if it is set, and if not, to default to tpcd, but so far
#         no luck.
##############################################################
#######

#LOCAL=tpcd

BASE=$(HOME)/sqllib
COMPILE_FLAGS= -m64 -c -DSQLAIX -DLINUX -I$(BASE)/include -g
#COMPILE_FLAGS= -c -DSQLAIX -I$(BASE)/include -g
# if using an installed db2 image use the 2nd link_flags value
LINK_FLAGS= -m64 -o $@ -L$(BASE)/lib -ldb2
#LINK_FLAGS= -o $@ -L/usr/lpp/db2_05_00/lib -ldb2
COMPILER=cc
LIB_LINKER=ld
LIB_LINK_FLAGS= -o $@ -H512 -T512 -bE:$@.exp -L$(BASE)/lib -ldb2
-lc

cleanup :
              rm -f tpcdbatch tpcdbatch.bnd tpcdbatch.o tpcdbatch.c
tpcdbatch.u tpcdUF.bnd tpcdUF.o tpcdUF.c tpcdUF.u 2>/dev/null

all : tpcdbatch


tpcdbatch.c : tpcdbatch.sqc
              @echo -e 'connect to $(TPCD_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
```

D.5 ploaduf1

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

D.6 ploaduf2

```
#!/bin/ksh
RFpair=$1;
db2 connect to tpcd
db2 "load from delete.$RFpair of del modified by coldel| fastparse  messages
/dev/null replace into TPCDTEMP.ORDERS_DEL nonrecoverable
partitioned db config mode load_only part_file_location /backup_3/ufdata;"
db2 commit;
db2 connect reset
db2 terminate
```

D.7 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");
}

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 ($gatherstats eq "on")
{
   # gather vm io and net stats
   if ($platform eq "aix" || $platform eq "sun" || $platform eq "ptx" ||
$platform eq "linux")
    {
      # kill the stats that were being gathered
       if ($platform eq "ptx")
       {
          $rc= `perl5 zap "-f" "sar"`;
          $rc= `perl5 zap "-f" "sadc"`;
       }
       else
       {
          $rc= `perl5 zap "-f" "vmstat"`;
          $rc= `perl5 zap "-f" "iostat"`;
       }
       if ( $pn > 1 )
       {
          $rc= `perl5 zap "-f" "netstat"`;
       }
       $rc= `perl5 zap "-f" "getstats"`;
    }
}

open(MISO, ">>$misofile") || die "Can't open $misofile: $!\n";
$curTs = `perl gettimestamp "long"`;
print MISO "Timestamp and isolation level of tpcdbatch after power run at :
$curTs\n";
close(MISO);

if ( $product eq "pe" )
{
   system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_id,isolation from sysibm.sysplan where name like
'TPCD%'\";db2 connect reset;db2 terminate  >>
$runDir${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.metri
cs");

if ($runUF eq "UF") {

&cat("$runDir${delim}mpufinter*","$runDir${delim}mpinter$runNum.metr
ics");
}

#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");
}
```

D.8 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.metri
cs");

if ($runUF ne "no") {

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

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");
}
```

D.9 tpcdbatch.h

```
/**************************************************************
****************
*
*   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 August 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 August 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 August 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.
*****************************************************************
****************/

/* 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 August 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 August 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);
  }
#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);
  }
#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 )
        {
          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);



   /* 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 */

   EXEC SQL CONNECT RESET;
#ifndef SQLWINT
      if (verbose)
      {
        fprintf(stderr,
              "cleanup: semkey = %ld, semid = %d, file = %s, stream = %d\n",
              semkey,semid,g_struct->update_num_file,
              g_struct->c_l_opt->intStreamNum);
      }
#endif
   }


   /** Summary table processing **/                        /* @d24993 tjg */
   summary_table(g_struct);

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

   fclose(outstream);           /* Close the output data stream.     */
   fclose(instream);            /* Close the SQL input stream.       */

   return (TRUE);
```

```
}

void create_semaphores(struct global_struct *g_struct)
{

#ifndef SQLWINT
  int             semid;          /* semaphore for controlling UFs*/
  key_t           semkey;         /* key to generate semid */
#else
  HANDLE          hSem;
  HANDLE          hSem2;
  int             SemTimeout = 600000;    /* Des time out period of 1 minute
*/
#endif
        fprintf(stderr,"numstreams = %d\n",g_struct->c_l_opt->intStreamNum);
        fprintf(stderr,"Update stream creating semaphore(s) for update and
query sequencing\n");
#ifdef SQLWINT

        fprintf(stderr,"semfile = %s\n",g_struct->sem_file);
        if (g_struct->c_l_opt->intStreamNum == 0)
        /*RUNPOWER*/
        {
                fprintf(stderr,"semfile2 = %s\n",g_struct->sem_file2);
                hSem = CreateSemaphore(NULL, 0,1,(LPCTSTR)(g_struct-
>sem_file));
                hSem2 = CreateSemaphore(NULL, 0,1,(LPCTSTR)(g_struct-
>sem_file2));
                if ((hSem == NULL) || (hSem2 == NULL))
                {
                  fprintf(stderr,
                        "CreateSemaphores (ready semaphore) failed, 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);
                }
        }
        else
        /* THROUGHPUT */
        {

                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);
                }
                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
```

D.10 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 August 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 August 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 August 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.
***************************************************************
****************/

/* 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 August 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 August 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);
  }
#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);
   }
#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 )
      {
        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);



  /* 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 */

  EXEC SQL CONNECT RESET;
#ifndef SQLWINT
    if (verbose)
    {
```

```
      fprintf(stderr,
            "cleanup: semkey = %ld, semid = %d, file = %s, stream = %d\n",
            semkey,semid,g_struct->update_num_file,
            g_struct->c_l_opt->intStreamNum);
    }
#endif
  }


  /** Summary table processing **/                    /* @d24993 tjg */
  summary_table(g_struct);

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

  fclose(outstream);          /* Close the output data stream.     */
  fclose(instream);           /* Close the SQL input stream.       */

  return (TRUE);
}

void create_semaphores(struct global_struct *g_struct)
{

#ifndef SQLWINT
  int               semid;            /* semaphore for controlling UFs*/
  key_t              semkey;           /* key to generate semid */
#else
  HANDLE            hSem;
  HANDLE            hSem2;
  int               SemTimeout = 600000;    /* Des time out period of 1 minute
*/
#endif
      fprintf(stderr,"numstreams = %d\n",g_struct->c_l_opt->intStreamNum);
      fprintf(stderr,"Update stream creating semaphore(s) for update and
query sequencing\n");
#ifdef SQLWINT

      fprintf(stderr,"semfile = %s\n",g_struct->sem_file);
      if (g_struct->c_l_opt->intStreamNum == 0)
      /*RUNPOWER*/
      {
            fprintf(stderr,"semfile2 = %s\n",g_struct->sem_file2);
            hSem = CreateSemaphore(NULL, 0,1,(LPCTSTR)(g_struct-
>sem_file));
            hSem2 = CreateSemaphore(NULL, 0,1,(LPCTSTR)(g_struct-
>sem_file2));
            if ((hSem == NULL) || (hSem2 == NULL))
            {
              fprintf(stderr,
                    "CreateSemaphores (ready semaphore) failed, 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);
        }
    }
    else
    /* THROUGHPUT */
    {

        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);
        }
        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
```

D.11 tpcduf.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 August 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 August 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 August 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 calcmetricpl.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.
****************************************************************
****************/

/* 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 August 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 August 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);
   }
#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);
  }
#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 )
      {
        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);



  /* 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 */

  EXEC SQL CONNECT RESET;
#ifndef SQLWINT
    if (verbose)
    {
      fprintf(stderr,
            "cleanup: semkey = %ld, semid = %d, file = %s, stream = %d\n",
            semkey,semid,g_struct->update_num_file,
            g_struct->c_l_opt->intStreamNum);
    }
#endif
  }


  /** Summary table processing **/                    /* @d24993 tjg */
  summary_table(g_struct);

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

  fclose(outstream);          /* Close the output data stream.     */
  fclose(instream);           /* Close the SQL input stream.       */

  return (TRUE);
}

void create_semaphores(struct global_struct *g_struct)
{

#ifndef SQLWINT
  int               semid;             /* semaphore for controlling UFs*/
  key_t             semkey;            /* key to generate semid */
#else
  HANDLE            hSem;
  HANDLE            hSem2;
  int               SemTimeout = 600000;    /* Des time out period of 1 minute
*/
#endif
      fprintf(stderr,"numstreams = %d\n",g_struct->c_l_opt->intStreamNum);
      fprintf(stderr,"Update stream creating semaphore(s) for update and
query sequencing\n");
#ifdef SQLWINT

      fprintf(stderr,"semfile = %s\n",g_struct->sem_file);
      if (g_struct->c_l_opt->intStreamNum == 0)
      /*RUNPOWER*/
      {
            fprintf(stderr,"semfile2 = %s\n",g_struct->sem_file2);
            hSem = CreateSemaphore(NULL, 0,1,(LPCTSTR)(g_struct-
>sem_file));
            hSem2 = CreateSemaphore(NULL, 0,1,(LPCTSTR)(g_struct-
>sem_file2));
            if ((hSem == NULL) || (hSem2 == NULL))
            {
              fprintf(stderr,
                  "CreateSemaphores (ready semaphore) failed, 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);
            }
      }
      else
      /* THROUGHPUT */
      {

            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);
            }
            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
}
```

Appendix E: ACID Transaction Source Code

E.1 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 */
```

E.2 acid.h

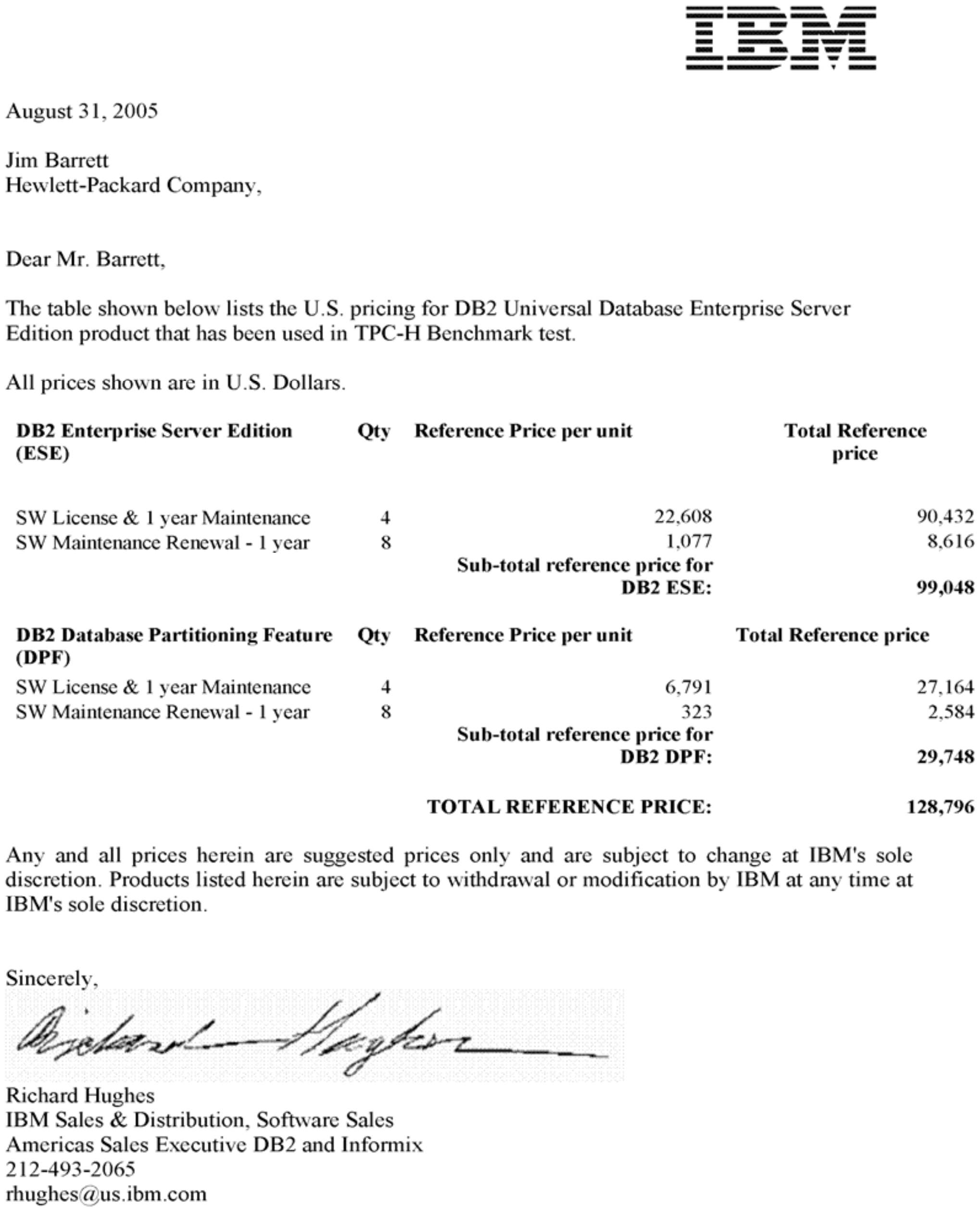
```
/*************************************************************
***********/
/*      File: acid.h                                          */
/*************************************************************
***********/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#ifdef SQLWINT
  #include <windows.h>
  #include <sys\timeb.h>
  #include <sys\stat.h>
  #include <stdlib.h>
  #include <io.h>
#else
  #include <unistd.h>
  #include <sys/time.h>
  #include <sys/timeb.h>
#endif

#include <string.h>
#include <math.h>

#define acidtime(tvsec,tvusec) tvsec*1000+tvusec/1000
#define TSLEN 20

#if 0 /* needed on NT, not on AIX */
typedef struct timeval {
        long    tv_sec;         /* seconds */
        long    tv_usec;        /* and microseconds */
};
#endif

struct update_struct {
   int     qnum;
};

struct acidQ_struct {
   int     tag;
   long    o_key;
   double  l_extendedprice;
};

struct acidT_struct {
   int     termination;
   int     tag;
   int     logging;
   long    o_key;
   long    l_key;
   long    delta;
   long    l_partkey;
   long    l_suppkey;
   double  l_quantity;
   double  l_tax;
   double  l_discount;
   double  l_extendedprice;
   double  o_totalprice;
};

/*
** in acid.sqc
*/

int updateQ (struct update_struct *us);
```

```
char del(void);

#ifdef
```

E.3 Makefile

```
DBNAME =        $(TPCD_QUAL_DBNAME)

INCLUDE =       $(HOME)/sqllib/include
DBNAME =        tpcd
#CFLAGS =       -I$(INCLUDE) -g -Dpascal= -DLINT_ARGS \
#       -Dfar= -D_loadds= -DSQLA_NOLINES -qflag=i:i -qlanglvl=ansi

#LFLAGS  =      -lm -lcurses -ls -ll -ly -liconv -lbsd
CFLAGS   =      -I$(INCLUDE) -Dpascal= -DLINT_ARGS  \
       -DSQLA_NOLINES -DLinux
# .. sun        -DSQLA_NOLINES

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


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

CC      =       g++

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



August 31, 2005

Jim Barrett
Hewlett-Packard Company,

Dear Mr. Barrett,

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

All prices shown are in U.S. Dollars.

DB2 Enterprise Server Edition (ESE)	**Qty**	**Reference Price per unit**	**Total Reference price**
SW License & 1 year Maintenance	4	22,608	90,432
SW Maintenance Renewal - 1 year	8	1,077	8,616
		Sub-total reference price for DB2 ESE:	**99,048**

DB2 Database Partitioning Feature (DPF)	**Qty**	**Reference Price per unit**	**Total Reference price**
SW License & 1 year Maintenance	4	6,791	27,164
SW Maintenance Renewal - 1 year	8	323	2,584
		Sub-total reference price for DB2 DPF:	**29,748**
		TOTAL REFERENCE PRICE:	**128,796**

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