IBM RISC System/6000
Enterprise Server S70 Advanced
using
DB2 Universal Database 5.2.0
executing
5 Concurrent Query Streams

TPC Benchmark™ D
Full Disclosure Report

IBM System Performance and Evaluation Center

Submitted For Review
November 2, 1998

Special Notices

The following terms used in this publication are trademarks of **International Business Machines** Corporation in the United States and/or other countries:

RISC System/6000
AIX
DB2
DB2 UDB
IBM

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

TPC Benchmark	Trademark of the Transaction Processing Performance Council
TPC-D	Trademark of the Transaction Processing Performance Council
QppD	Trademark of the Transaction Processing Performance Council
QthD	Trademark of the Transaction Processing Performance Council
QphD	Trademark of the Transaction Processing Performance Council

First Edition November 2, 1998

The information contained in this document is distributed on an AS IS basis without any warranty either expressed or implied. The use of this information or the implementation of any of these techniques is a customer's responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

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

It is possible that this material may contain references to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such products, programming, or services in your country.

All performance data contained in this publication was obtained in a controlled environment, and therefore the results which may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data in their specific environment.

Request for additional copies of this document should be sent to the following address:

TPC Benchmark Administrator
IBM System and Performance Evaluation Center
Mail Stop 9221
11400 Burnet Road
Austin, TX 78758
FAX Number (512) 838-1852

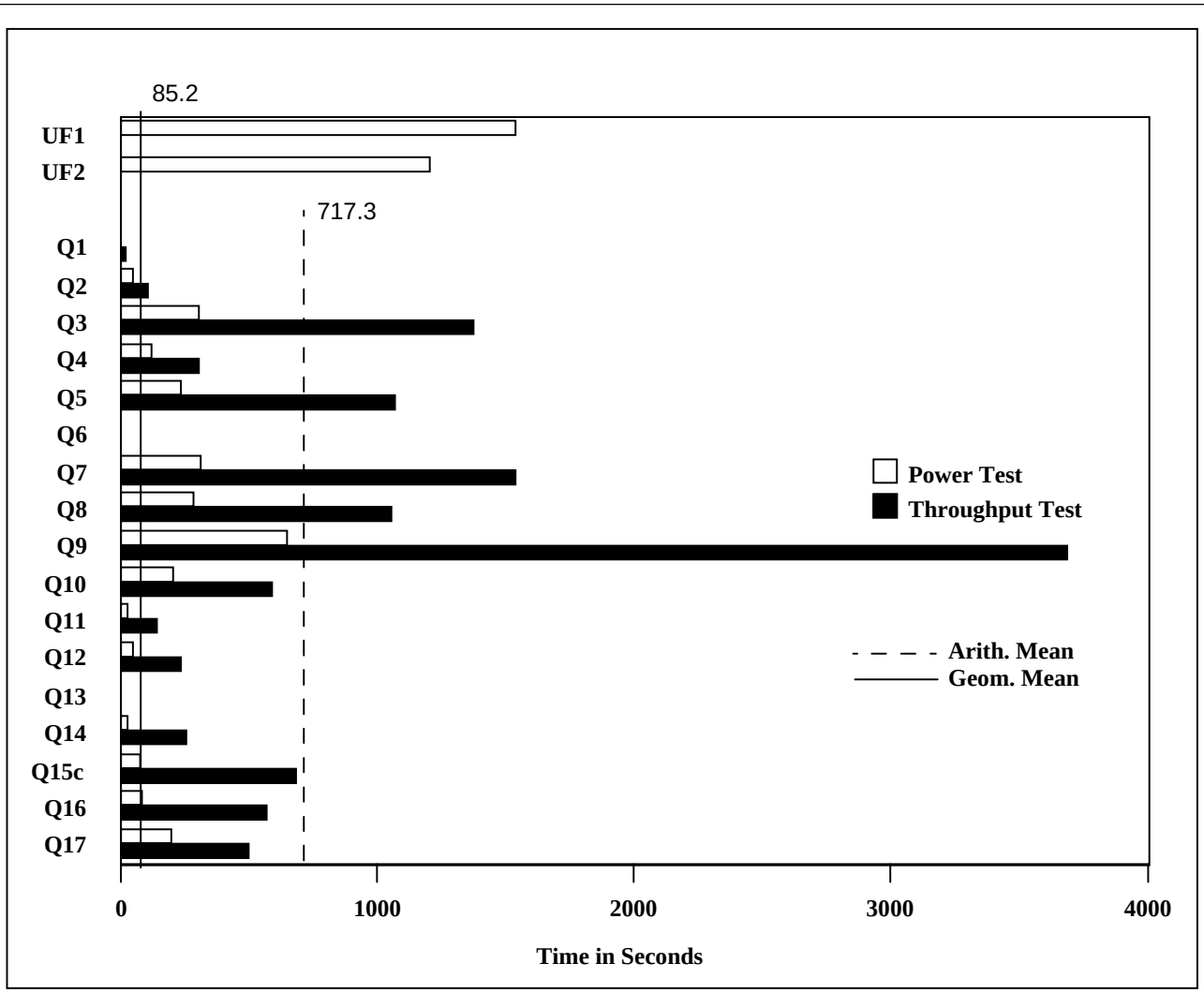
IBM RS/6000 S70 Advanced with DB2 UDB 5.2.0

TPC-D Rev. 1.3.1

Report Date: November 2, 1998

Total System Cost	TPC-D Power	TPC-D Throughput	Price/Performance
$1,266,483	**4,226.5** QppD@100GB	**1,092.6** QthD@100GB	**$589** $ / QphD@100GB

Database Size	Database Manager	Operating System	Other Software	Availability Date
100GB	DB2 UDB 5.2.0	IBM AIX 4.3.2	None	Dec. 31, 1998



Database Load Time: 11:16:36	Total Data Storage/Database Size: 10.22	Raid: Y (RAID 1)

System Components

Processors	12 x 262MHz PowerPC RS64-2 w/ 8MB L2 cache
Memory	16GB
Disk Controllers	3 PCI F/W SCSI 12 PCI SSA (Serial Storage Architecture)
Disk Drives	7 x 9.1GB F/W SCSI Hot Pluggable 192 x 4.5GB SSA Hot Pluggable, 16 x 9.1GB SSA Hot Pluggable
Total GB Storage	1022.75 GB

IBM RS/6000 S70 Advanced with DB2 UDB 5.2.0

TPC-D Rev. 1.3.1

Report Date: November 2, 1998

Description	Number	Source	Price	Qty	Disc	Price	Price
Server Hardware							
RS/6000 Enterprise Server S7A, 1.44 Diskette Dr	7017-S7A	IBM	13,271	1		13,271	48,960
4-way 262MHz RS64-2 Proc. Card, 8MB L2 ea.	5312	IBM	75,000	2		150,000	80,640
4-way 262MHz RS64-2 Proc. Card, 8MB L2 ea.	5313	IBM	75,000	1		75,000	40,320
8192 MB R1 Memory (4 x 2048MB cards)	4179	IBM	98,304	2		196,608	0
Token Ring Adapter	2920	IBM	795	1		795	0
Remote I/O Cable - Drawer to Drawer	3126	IBM	595	1		595	0
Drawer to Drawer Power Control Cable	6006	IBM	60	1		60	0
I/O Rack	7000	IBM	3,500	1		3,500	0
Support Processor Group	6322	IBM	2,700	1		2,700	0
Ultra SCSI PCI - Bus Adapter	6206	IBM	395	3		1,185	0
SSA Multi-Initiator/RAID EL Adapter - PCI	6215	IBM	3,000	12		36,000	0
SSA Fast Write Cache Option	6222	IBM	3,700	12		44,400	0
Prestige 6000 VA, UPS - CPU Drawer	9910-EP8	IBM	5,505	1		5,505	0
Prestige 3000 VA, UPS - SSA Drawers	9910-U33	IBM	4,221	4		16,884	0
Internal 12/24GB 4mm Tape Drive	6159	IBM	3,200	1		3,200	0
IBM ASCII Terminal, Keyboard, Cable	3153-BG3	IBM	622	1		622	660
			Subtotal			550,325	170,580
Server Software							
AIX 4.3.2 for S7A - 50 Designated Users	5765-C34	IBM	4,610	1		4,610	0
IBM C for AIX V4.4	5765-C64	IBM	1,578	1		1,578	0
Performance Toolbox V2.2	5765-654	IBM	850	1		850	0
DB2 UDB EEE for AIX - Media Package	31L0212	IBM	34,999	1		34,999	0
DB2 UDB EEE 50 User NC Pack	31L0224	IBM	0	1		0	0
DB2 UDB EEE up to 16 Processors	31L0227	IBM	244,999	1		244,999	0
			Subtotal			287,036	0
Storage Devices							
9.1 GB SCSI-2 F/W Hot Swap Disk	2913	IBM	2,200	7		15,400	0
SCSI I/O Drawer	6320	IBM	23,766	2		47,532	0
Primary I/O Drawer Group	6321	IBM	600	1		600	0
Secondary I/O Drawer Group	6323	IBM	734	1		734	0
SCSI 6-pack Hot Swap Back Plane	6547	IBM	600	2		1,200	0
System Rack Model R00	7015-R00	IBM	3,110	2		6,220	2,976
SSA Disk Subsystem w/ 4 4.5GB Disks	7133-020	IBM	19,350	13		251,550	124,800
4.5 GB Disk Drive Module	3401	IBM	1,900	144		273,600	0
9.1 GB Disk Drive Module	3901	IBM	3,000	12		36,000	0
9.1 GB Disk Drive Module Select	3904	IBM	2,100	4		8,400	0
SSA Cables	5006	IBM	40	26		1,040	0
			Subtotal			642,276	127,776
			Total			**1,479,637**	**298,356**
Discounts							
Midrange Service Option					17%	(50,721)	
Extended Maintenance Option					17%	(42,098)	
IBM Software Revenue Discount - DB2					21%	(58,800)	
IBM Dollar Volume Discount					30%	(359,892)	
	Five Year Cost of Ownership:					**$1,266,483**	
	QppD		4,226.5				
	QthD		1,092.6				
	QphD		2,148.9				
			$/QphD@100GB			**$589**	

Audited by Francois Raab, Information Paradigm, Inc.

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

IBM RS/6000 S70 Advanced with DB2 UDB 5.2.0

TPC-D Rev. 1.3.1

Report Date: November 2, 1998

Numerical Quantities Summary

Measurement Results:

Database Scaling (SF/Size)	=	100
Total Data Storage/Database Size	=	10.22
Database Load Time	=	11:16:36
Query Streams for Throughput Test	=	5
TPC-D Power Metric (QppD@100GB)	=	4,226.5
TPC-D Throughput Metric (QthD@100GB)	=	1,092.6
Composite Query-per-Hour Rating (QphD@100GB)	=	2,148.9
Total System Price over 5 years	=	$1,266,483
TPC-D Price/Performance Metric ($/QphD@100GB)	=	$589

Measurement Intervals

Measurement Interval in Throughput Test (Ts) = 28,006 seconds

Duration of Stream Execution

Stream ID	Seed Used	Start Date	Start Time	End Date	End Time
Stream 00	304789933	10/26/98	22:48:46	10/27/98	00:19:43
Stream 01	542987759	10/27/98	00:27:50	10/27/98	03:49:16
Stream 02	300838752	10/27/98	00:27:50	10/27/98	03:52:31
Stream 03	1046734243	10/27/98	00:27:50	10/27/98	03:50:17
Stream 04	731427844	10/27/98	00:27:50	10/27/98	03:52:23
Stream 05	1768551525	10/27/98	00:27:50	10/27/98	03:50:49
Updates		10/27/98	00:27:50	10/27/98	08:14:36

Timing Intervals (in Seconds)

Stream ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
Stream 00	3.5	52.7	308.5	128.1	241.3	0.5	320.7	289.7	654.1
Stream 01	34.2	135.5	1315.7	399.2	946.2	1.0	1533.0	1030.8	3429.9
Stream 02	24.8	92.0	1331.2	384.1	1053.4	4.9	1692.5	1063.1	3754.6
Stream 03	18.2	78.3	1467.2	322.9	1100.8	3.8	1555.4	1274.7	3368.1
Stream 04	29.2	83.7	1177.4	146.6	1184.9	1.4	1421.8	1037.4	4389.5
Stream 05	30.8	168.5	1603.6	304.1	1093.4	0.7	1502.4	870.3	3499.9
Minimum	18.2	78.3	1177.4	146.6	946.2	0.7	1421.8	870.3	3368.1
Average	27.4	111.6	1379.0	311.4	1075.7	2.4	1541.0	1055.3	3688.4
Maximum	34.2	168.5	1603.6	399.2	1184.9	4.9	1692.5	1274.7	4389.5

Stream ID	Q10	Q11	Q12	Q13	Q14	Q15c	Q16	Q17	UF1	UF2
Stream 00	212.2	28.8	53.6	2.1	31.5	85.1	91.7	200.8	1211.6	1541.1
Stream 01	594.8	144.3	254.5	3.9	289.0	551.8	813.6	608.7	13381.9	1918.3
Stream 02	606.8	192.9	237.1	7.9	283.9	684.6	435.1	432.3	1248.6	1904.4
Stream 03	527.6	185.2	270.2	4.3	208.7	797.3	483.3	481.6	1330.7	1787.1
Stream 04	691.9	96.9	225.0	3.7	220.0	530.0	501.9	532.2	1323.7	1951.0
Stream 05	562.5	122.5	190.5	4.5	282.7	871.7	628.3	442.6	1331.5	1829.4
Minimum	527.6	96.9	190.5	3.7	208.7	530.0	435.1	432.3	1248.6	1787.1
Average	596.7	148.4	253.5	4.9	256.9	687.1	572.4	499.5	3723.3	1878.0
Maximum	691.9	192.9	270.2	7.9	289.0	871.7	813.6	608.7	13381.9	1951.0

Information Paradigm

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL

Certified Auditor

Test Sponsors: John Fowler
RS/6000 Division
IBM Corporation
11400 Burnet Road
Austin, TX 78758

November 2, 1998

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

Platform: IBM RS/6000 S70 Advanced
DataBase Manager: DB2 UDB Version 5.2.0
Operating System: IBM AIX Version 4.3.2

The results were:

CPU's Speed	Memory	Disks	QppD@100GB	QthD@100GB
IBM RS/6000 S70 Advanced				
12 x PowerPC RS64-2 (262 MHz)	8 MB L2 each 16 GB main	192 x 4.5 GB SSA 16 x 9.1 GB SSA 7 x 9.1 GB SCSI	4,226.5	1,092.6

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

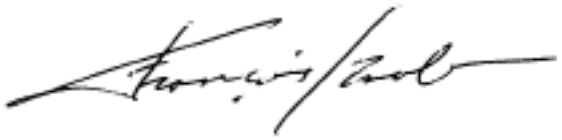
- The TIME table was not used
- The input variables were generated by QGEN
- The database was populated using DBGEN
- The database was maintained by the "evolve" method
- The throughput metric was computed using the results from a 5-stream test
- The ratio between the longest and the shortest query was such that the query time for Q6 was adjusted.
- A compliant implementation specific layer was used
- The query text was produced using compliant variants and minor modifications

- The database records were defined with the proper layout and size
- The database was properly scaled to 100GB and populated accordingly
- The database load time was correctly measured and reported
- The ACID Properties were verified and met
- The reported execution times were correctly measured and reported
- Measurement repeatability was verified
- At least 8 hours of database log was configured
- The system pricing was verified for major components and maintenance
- The major pages from the FDR were verified for accuracy

Additional Audit Notes:

None.

Respectfully Yours,



François Raab
President

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

Abstract

This report documents the full disclosure information required by the TPC Benchmark ™ D Standard Specification Revision 1.3.1 dated February 12, 1998, for measurements on the IBM RISC System/6000 Enterprise Server S70 Advanced. The phrase RS/6000 will be substituted for RISC System/6000 for the remainder of this document.

The software used includes AIX Version 4.3.2 and DB2 Universal Database Enterprise Extended Edition 5.2.0.

Preface

TPC Benchmark™ D Standard Specification was developed by the Transaction Processing Performance Council (TPC). It was released on May 5, 1995, and most recently revised (Revision 1.3.1) on February 12, 1998. This is the full disclosure report for benchmark testing of the IBM RS/6000 Enterprise Server S70 Advanced according to the TPC Benchmark™ D Standard Specification.

TPC Benchmark™ D is a Decision Support benchmark. It is a suite of business oriented queries and concurrent updates. The queries and the data populating the database have been chosen to have broad industry -wide relevance while maintaining a sufficient degree of ease of implementation. This benchmark illustrates Decision Support systems that:

- Examine large volumes of data;
- Execute queries with a high degree of complexity;
- Give answers to critical business questions.

TPC-D evaluates the performance of various Decision Support Systems by the execution of sets of queries against a standard database under controlled conditions. The TPC -D queries:

- Give answers to real-world business questions;
- Are far more complex than most OLTP transactions;
- Include a rich breadth of operators and selectivity constraints;
- Generate intensive activity on the part of the database server component of the system under test;
- Are executed against a database complying to specific population and scaling requirements;
- Are implemented with constraints derived from staying closely synchronized with an on -line production database.

The TPC-D operations are modeled after the following business environment:

- The database is continuously available 24 hours a day, 7 days a week, for queries or updates against all tables for multiple users, except possibly during infrequent (e.g., once a month) maintenance sessions;
- The TPC-D database tracks, possibly with some delay, the state of the OLTP database through ongoing updates which batch together a number of modifications impacting some part of the Decision Support database;
- Due to the worldwide nature of the business data stored in the TPC -D database, the queries and the updates may be executed against the database at any time, especially in relation to each other. In addition, this mix of queries and updates is subject to specific ACIDity requirements, since queries and updates may execute concurrently;
- To achieve the optimal compromise between performance and operational requirements, the database administrator can set, once and for all, the locking levels and the concurrent scheduling rules for queries and updates.

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

The performance metrics reported by TPC-D measure multiple aspects of the capability of the system to process queries. These aspects include the selected database size against which the queries are executed, the TPC-D query processing power at the selected size (QppD@Size), and the TPC-D throughput at the selected size (QthD@Size) when queries are submitted by one or more concurrent users. The TPC-D Price/Performance metric is expressed as $/QphD@Size and is based on a composite query-per-hour rating derived from QppD and QthD. To be compliant with the TPC-D standard, all references to TPC-D results for a given configuration must include all required reporting components (see Clause 5.4.7).

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

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

1.0 General Items

1.1 Benchmark Sponsor

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

This benchmark was sponsored by **International Business Machines Corporation.**

1.2 Parameter Settings

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

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

Appendix A. "Tunable Parameters" contains a list of all DB2 parameters, operating system parameters and compiler options. Session initialization parameters can be set during or immediately after establishing the connection to the database within the tpcdbatch program documented in Appendix D. "Driver Source Code" . This result uses the default session initialization parameters established during preprocessing/binding of the tpcdbatch program. The procedure for preprocessing, binding, compiling and linking the tpcdbatch program is documented in Appendix A.5 , "Compiler Options" .

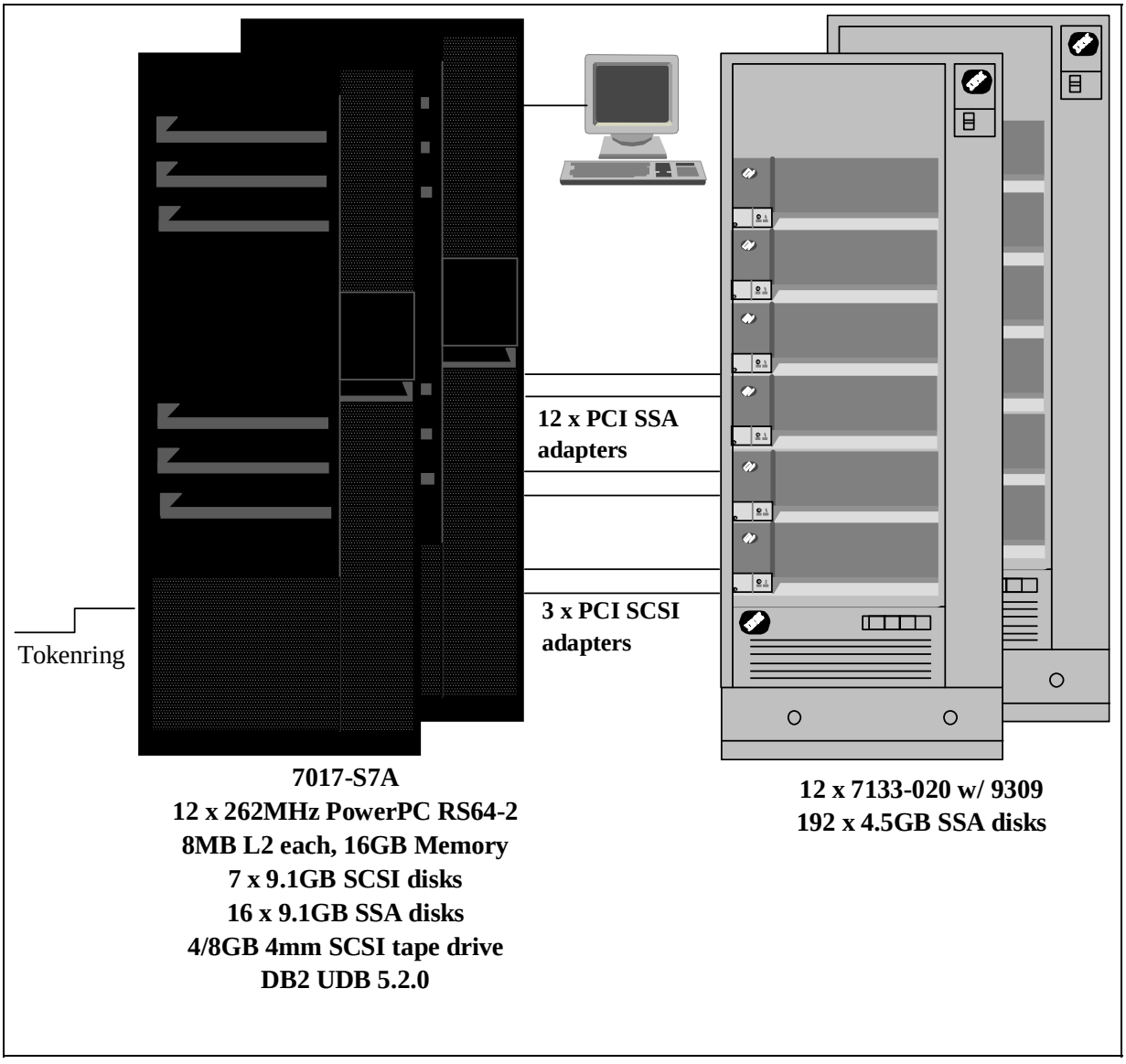
1.3 Configuration Diagrams

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

- *Number and type of processors*
- *Size of allocated memory, and any specific mapping/partitioning of memory unique to the test and type of disk units (and controllers, if applicable)*
- *Number and type of disk units (and controllers, if applicable).*
- *Number of channels or bus connections to disk units, including the protocol type*

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

IBM RS/6000 Enterprise Server S70 Advanced Benchmark and Priced Configuration



For full details of the Priced configuration see the Pricing spreadsheet in the Executive Summary.

2.0 Clause 1: Logical Database Design

Related Items

Appendix B. "Database Build Scripts" contains the programs and input files used to load the test database. The test and qualification databases use the same table definitions, indices and partitioning methods. Thus, the buildtpcd script documented in Appendix B.1 was used for both the qualification and test databases except that different input files were used to define the tablespace devices and sizes.

There are three phases for the loading of the database, the generation of the flat data files, the splitting of the data into separate files for each logical node, and the building of the database from this data. The generation and splitting of the data use the gensplit100GB.ksh script, documented in Appendix B.2. The buildtpcd script is then used to create the database, load the data into the tables, create indices, gather statistics, add constraints, create automatic summary table, and set configuration. It calls the doload.ksh script which then calls the appropriate load script for each of the tables. These load scripts are documented in Appendix B.3 through B.9

A brief description of the other files which are in Appendix B.11 through B.23 follows:

- B.10 - DB2 nodes file (db2nodes.cfg) contains the names of the nodes where the database is to be build.
- B.11 - database config file (dss.dbconfig100GB.acme) containse the database configuration parameters that are set after the database build, prior to executing the tests.
- B.12 - database manager config file (dss.dbmconfig100.acme) contains the database manager configuration parameters that are set after the database has been built and prior to running the measurements.
- B.13 - DB2 commands to create the tablespaces on the defined AIX logical volumes (dss.ddl100GB.tbsp.acme.12w.mirror.addtmp).
- B.14 - DB2 commands to create the tables in the defined tablespaces (dss.ddl100GB.tbl.acme).
- B.15 - DB2 commands to creates the indexes for the tables (dss.index.acme)
- B.16 - DB2 commands to add constraints to the tables (dss.ri3).
- B.17 - DB2 commands to add Automatic Summary Table (dss.AST).
- B.18 - DB2 commands to run statistics for the indexes (dss.runstats)
- B.19 - build/run environment variable configuration (tpcd.setup) contains the environment variables the indicate where to build the database, names of files used by both buildtpcd and tpcdbatch (the implementation specific layer)
- B.20 - database configuration file (dss.loaddbconfig) contains the database configuration parameters that are set during the database build.
- B.21 - database manager config file (dss.loaddbmconfig) contains the database manager configuration parameters that are during the database built.
- B.22 & B.23 - DB2 commands to create the update staging tables.

2.1 Table Definitions

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

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

2.2 Database Organization

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

Appendix B. "Database Build Scripts" contains the DDL for the 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.13.

2.4 Replication

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

No replication beyond the creation of indices and automatic summary tables (see Appendix B. "Database Build Scripts") was used.

3.0 Clause 2: Queries and Update Functions

3.1 Query Language

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

SQL was the query language used.

3.2 Verification for the Random Number Generator

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

The supplied DBGEN version 1.3.1 and QGEN version 1.3.1 were used to generate all database populations.

3.3 Substitution Parameters

3.3.1 Method of Generation

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

The supplied QGEN version 1.3.1 was used to generate the substitution parameters.

3.4 Query Text

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

Appendix C.1 "Qualification Query Output" contains the output for each of the queries. The functional query definitions and variants used in this disclosure use the following minor query modifications.

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

3.5 Disclosure

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

Appendix C4 , "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 was **repeatable read**

3.7 Update Functions

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

The update function is part of the implementation specific layer/driver code included in Appendix D. "Driver Source Code" .

3.8 Database Maintenance Option

The details of the database maintenance option selected (i.e., reset or evolve) must be disclosed (including source code of any non-commercial program used).

This implementation uses the evolve option.

4.0 Clause 3: Database System Properties Related Items

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

All ACID tests were conducted according to specification. The Atomicity, Isolation, Consistency and Durability tests were performed on the RS/6000 F50-332. Appendix E. "Acid Transaction Source Code" contains the source code for the ACID transaction and query.

4.1 Atomicity Requirements

The system under test must guarantee that transactions are atomic; the system will either perform all individual operations on the data, or will assure that no partially-completed operations leave any effects on the data.

4.1.1 Atomicity of Completed Transaction

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

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

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

4.1.2 Atomicity of Aborted Transactions

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

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

1. The 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 step 2 and step 3. It was verified that the extended price and the total price were the same as in step 2.

4.2 Consistency Requirements

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

4.2.1 Consistency Condition

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

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

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

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

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

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

4.2.2 Consistency Tests

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

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

After executing 6 streams of 100 ACID transactions each, the queries defined in 4.2.1 , "Consistency Condition" were run again. The queries showed that the database was still in a consistent state.

4.3 Isolation Requirements

4.3.1 Isolation Test 1

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

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

1. 1st session: Start an ACID transaction with a randomly selected O_KEY,L_KEY and DELTA. The transaction is delayed for 60 seconds just prior to the Commit.

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

4.3.2 Isolation Test 2

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

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

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

4.3.3 Isolation Test 3

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

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

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

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

4.3.4 Isolation Test 4

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

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

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

4.3.5 Isolation Test 5

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

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

4.3.6 Isolation Test 6

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

1. 1st session: A transaction T1, which executes modified TPC-D query 1 with DELTA=0, was started.
2. 2nd session: Before T1 completed, an ACID transaction T2, with randomly selected values of O_KEY, L_KEY and DELTA, was started.
3. 3rd session: Before T1 completed, a transaction T3, which executes modified TPC-D query 1 with a randomly selected value of DELTA (not equal to 0), was started.
4. T1 completed.
5. T2 completed.
6. T3 completed.
7. It was verified that the appropriate rows in the ORDERS, LINEITEM and HISTORY tables were changed.

4.3.7 Isolation Test 7 (Isolation of Automatic Summary Tables)

This test demonstrates isolation for the read-write conflict of a read-write committed transaction and a read-only transaction when both transactions access a base table on which an automatic summary table (AST) has been defined.

1. 1st session: Verify the AST is not defined and execute the ACID query. Verify the ACID query accesses every row of the LINEITEM table.
2. 2nd session: Create the AST (and bind the SQL) and execute the ACID query. Verify the ACID query accesses the AST and gets the same result as when it accessed the LINEITEM table.
3. 3rd session: Start an ACID transaction with a randomly selected O_KEY, L_KEY, and DELTA that will update the LINEITEM table (and automatically refresh the AST). The transaction is delayed for 60 seconds just prior to the Commit.
4. 4th session: Start the same ACID query as in the 1st session.
5. 4th session: The ACID query attempts to read from the AST but is locked out by the the ACID transaction waiting to complete.
6. 3rd session: The ACID transaction is released and the Commit is executed releasing the record. With the LINEITEM record and the associated AST record(s) now released, the ACID query can now complete.
7. 4th session: Verify that the ACID query delays for approximately 60 seconds, that it accesses the AST, and that the result is different than in the 1st and 2nd sessions.
8. 5th session: Drop the AST (and bind the SQL) and execute the same ACID query. Verify the acid query accesses every row of the base table and gets the same answer as when it accessed the AST (in the 4th session).

4.4 Durability Requirements

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

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

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

Failure of Durable Medium Containing Recovery Log Data, Database tables and Loss of System Power/Memory

1. The complete database is backed up to disk.
2. The consistency test described in section 4.2.1 was verified.
3. The current count of the total number of records in the HISTORY table was determined giving hist1.
4. A test to run 100 ACID transactions on each of 6 execution streams was started.
5. One of the disks containing the DB2 transaction log recovery data was powered off after at least 15 ACID transactions had completed from each of the execution streams.
6. Because the disks were mirrored the applications continued running the ACID transactions.

7. One of the disks containing DB2 database tables was powered off after at least 15 additional ACID transactions had completed for each stream.
8. The transactions were not affected because of mirroring and the execution streams continued successfully.
9. The system was shutdown by switching the Emergency Power Off, after at least another 15 transactions had completed for each stream.
10. A SSA controller was replaced and the original fastwrite cache was restored (with committed transactions).
11. The system was powered back on and rebooted.
12. The mirrored partition on the disk s were reestablished and synchronized.
13. Step 2 was performed giving hist2. It was verified that hist2 - hist1 was greater than or equal to the number of records in the success file.
14. Consistency condition described in 4.2.1 was verified.

5.0 Clause 4: Scaling and Database Population Related Items

5.1 Cardinality of Tables

The cardinality (e.g., the number of rows) of each table of the test database, as it existed at the completion of the database load (see Clause 4.2.5), must be disclosed. The following table contains the TPC Benchmark™ D defined tables and the number of rows for each table as they existed upon build completion:

Table	Rows
Lineitem	600,037,902
Orders	150,000,000
Customer	15,000,000
Supplier	1,000,000
Part	20,000,000
Partsupp	80,000,000
Nation	25
Region	5

5.2 Distribution of Tables and Logs

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

The following diagram depicts the database configuration for each physical node of the system tested:

Table 1: Distribution of tables and logs

Filesystem or Tablespace	Size in MB	Physical Volumes	Description of Contents
rootvg /home/tpcd	9,000MB	scsi0: hdisk0, hdisk1, hdisk2, hdisk3, hdisk4, hdisk5, hdisk6, hdisk7	OS, 2GB paging, user directories, applications, etc...
paging	4,900MB		paging space (2,816MB per disk)
/UF100GB	1,900MB		

TPCDTEMP

Temporary Tablespace for TPC-D database

Size: 96GB

Node:	Logical Volumes:	Physical Volumes:
node0	lv.hdisk18.A1, lv.hdisk24.A1, lv.hdisk22.A1, lv.hdisk14.A1, lv.hdisk215.A1	**ssa0**: hdisk14, hdisk18, hdisk22, hdisk24, hdisk215
node1	lv.hdisk35.A1, lv.hdisk32.A1, lv.hdisk30.A1, lv.hdisk38.A1, lv.hdisk214.A1	**ssa1**: hdisk30, hdisk32, hdisk35, hdisk38, hdisk214
node2	lv.hdisk61.A1, lv.hdisk70.A1, lv.hdisk66.A1, lv.hdisk67.A1, lv.hdisk205.A1	**ssa2**: hdisk61, hdisk66, hdisk67, hdisk70, hdisk205
node3	lv.hdisk127.A1, lv.hdisk130.A1, lv.hdisk135.A1, lv.hdisk133.A1, lv.hdisk207.A1	**ssa3**: hdisk127, hdisk130, hdisk133, hdisk135, hdisk207
node4	lv.hdisk79.A1, lv.hdisk87.A1, lv.hdisk77.A1, lv.hdisk80.A1, lv.hdisk204.A1	**ssa4**: hdisk77, hdisk79, hdisk80, hdisk87, hdisk204
node5	lv.hdisk138.A1, lv.hdisk145.A1, lv.hdisk147.A1, lv.hdisk148.A1, lv.hdisk202.A1	**ssa5**: hdisk138, hdisk145, hdisk147, hdisk148, hdisk202
node6	lv.hdisk184.A1, lv.hdisk171.A1, lv.hdisk182.A1, lv.hdisk181.A1, lv.hdisk216.A1	**ssa6**: hdisk171, hdisk181, hdisk182, hdisk184, hdisk216
node7	lv.hdisk44.A1, lv.hdisk57.A1, lv.hdisk45.A1, lv.hdisk54.A1, lv.hdisk210.A1	**ssa7**: hdisk44, hdisk45, hdisk54, hdisk57, hdisk210
node8	lv.hdisk110.A1, lv.hdisk120.A1, lv.hdisk108.A1, lv.hdisk106.A1, lv.hdisk208.A1	**ssa8**: hdisk106, hdisk108, hdisk110, hdisk120, hdisk208
node9	lv.hdisk157.A1, lv.hdisk165.A1, lv.hdisk156.A1, lv.hdisk167.A1, lv.hdisk206.A1	**ssa9**: hdisk156, hdisk157, hdisk165, hdisk167, hdisk206
node10	lv.hdisk201.A1, lv.hdisk191.A1, lv.hdisk193.A1, lv.hdisk199.A1, lv.hdisk212.A1	**ssa10**: hdisk191, hdisk193, hdisk199, hdisk201, hdisk212
node11	lv.hdisk98.A1, lv.hdisk91.A1, lv.hdisk97.A1, lv.hdisk99.A1, lv.hdisk211.A1	**ssa11**: hdisk91, hdisk97, hdisk98, hdisk99, hdisk211

/db2log

AIX Mirrored filesystemfor DB2 Recovery log files.

Size: 23GB

File System	Logical Volume	Phyical Volumes
/node0vg.log	node0vg.log	**ssa0**: hdisk14, hdisk18, hdisk22, hdisk24
/node1vg.log	node1vg.log	**ssa1**: hdisk30, hdisk32, hdisk35, hdisk38
/node2vg.log	node2vg.log	**ssa2**: hdisk61, hdisk66, hdisk67, hdisk70
/node3vg.log	node3vg.log	**ssa3**: hdisk127, hdisk130, hdisk133, hdisk135
/node4vg.log	node4vg.log	**ssa4**: hdisk77, hdisk79, hdisk80, hdisk87
/node5vg.log	node5vg.log	**ssa5**: hdisk138, hdisk145, hdisk147, hdisk148
/node6vg.log	node6vg.log	**ssa6**: hdisk171, hdisk181, hdisk182, hdisk184
/node7vg.log	node7vg.log	**ssa7**: hdisk44, hdisk45, hdisk54, hdisk57
/node8vg.log	node8vg.log	**ssa8**: hdisk106, hdisk108, hdisk110, hdisk120
/node9vg.log	node9vg.log	**ssa9**: hdisk156, hdisk157, hdisk165, hdisk167
/node10vg.log	node10vg.log	**ssa10**: hdisk191, hdisk193, hdisk199, hdisk201
/node11vg.log	node11vg.log	**ssa11**: hdisk91, hdisk97, hdisk98, hdisk99

TPCDDATA

Data Tablespace for TPC-D database

Size: 154GB

Node:	Logical Volumes:	Physical Volumes:
node0	lv.hdisk13.B1 lv.hdisk25.B1 lv.hdisk12.B1 lv.hdisk20.B1 lv.hdisk11.B1 lv.hdisk21.B1 lv.hdisk23.B1 lv.hdisk19.B1 lv.hdisk17.B1 lv.hdisk10.B1 lv.hdisk15.B1 lv.hdisk16.B1	**ssa0**: hdisk10, hdisk11, hdisk12, hdisk13, hdisk15, hdisk16, hdisk17, hdisk19, hdisk20, hdisk21, hdisk23, hdisk25
node1	lv.hdisk31.B1 lv.hdisk37.B1 lv.hdisk26.B1 lv.hdisk40.B1 lv.hdisk41.B1 lv.hdisk39.B1 lv.hdisk27.B1 lv.hdisk33.B1 lv.hdisk29.B1 lv.hdisk36.B1 lv.hdisk34.B1 lv.hdisk28.B1	**ssa1**: hdisk26, hdisk27, hdisk28, hdisk29, hdisk31, hdisk33, hdisk34, hdisk36, hdisk37, hdisk39, hdisk40, hdisk41
node2	lv.hdisk68.B1 lv.hdisk64.B1 lv.hdisk65.B1 lv.hdisk73.B1 lv.hdisk59.B1 lv.hdisk72.B1 lv.hdisk60.B1 lv.hdisk69.B1 lv.hdisk58.B1 lv.hdisk71.B1 lv.hdisk62.B1 lv.hdisk63.B1	**ssa2**: hdisk58, hdisk59, hdisk60, hdisk62, hdisk63, hdisk64, hdisk65, hdisk68, hdisk69, hdisk71, hdisk72, hdisk73
node3	lv.hdisk123.B1 lv.hdisk137.B1 lv.hdisk134.B1 lv.hdisk125.B1 lv.hdisk129.B1 lv.hdisk131.B1 lv.hdisk122.B1 lv.hdisk126.B1 lv.hdisk132.B1 lv.hdisk136.B1 lv.hdisk124.B1 lv.hdisk128.B1	**ssa3**: hdisk122, hdisk123, hdisk124, hdisk125, hdisk126, hdisk128, hdisk129, hdisk131, hdisk132, hdisk134, hdisk136, hdisk137
node4	lv.hdisk75.B1 lv.hdisk81.B1 lv.hdisk89.B1 lv.hdisk82.B1 lv.hdisk88.B1 lv.hdisk76.B1 lv.hdisk83.B1 lv.hdisk78.B1 lv.hdisk74.B1 lv.hdisk85.B1 lv.hdisk84.B1 lv.hdisk86.B1	**ssa4**: hdisk74, hdisk75, hdisk76, hdisk78, hdisk81, hdisk82, hdisk83, hdisk84, hdisk85, hdisk86, hdisk88, hdisk89
node5	lv.hdisk146.B1 lv.hdisk139.B1 lv.hdisk152.B1 lv.hdisk142.B1 lv.hdisk151.B1 lv.hdisk153.B1 lv.hdisk140.B1 lv.hdisk143.B1 lv.hdisk149.B1 lv.hdisk150.B1 lv.hdisk144.B1 lv.hdisk141.B1	**ssa5**: hdisk139, hdisk140, hdisk141, hdisk142, hdisk143, hdisk144, hdisk146, hdisk149, hdisk150, hdisk151, hdisk152, hdisk153
node6	lv.hdisk175.B1 lv.hdisk178.B1 lv.hdisk174.B1 lv.hdisk180.B1 lv.hdisk183.B1 lv.hdisk185.B1 lv.hdisk172.B1 lv.hdisk177.B1 lv.hdisk176.B1 lv.hdisk179.B1 lv.hdisk173.B1 lv.hdisk170.B1	**ssa6:** hdisk170, hdisk172, hdisk173, hdisk174, hdisk175, hdisk176, hdisk177, hdisk178, hdisk179, hdisk180, hdisk183, hdisk185
node7	lv.hdisk42.B1 lv.hdisk51.B1 lv.hdisk53.B1 lv.hdisk52.B1 lv.hdisk43.B1 lv.hdisk49.B1 lv.hdisk47.B1 lv.hdisk56.B1 lv.hdisk55.B1 lv.hdisk50.B1 lv.hdisk48.B1 lv.hdisk46.B1	**ssa7:** hdisk42, hdisk43, hdisk46, hdisk47, hdisk48, hdisk49, hdisk50, hdisk51, hdisk52, hdisk53, hdisk55, hdisk56
node8	lv.hdisk116.B1 lv.hdisk112.B1 lv.hdisk113.B1 lv.hdisk118.B1 lv.hdisk117.B1 lv.hdisk119.B1 lv.hdisk115.B1 lv.hdisk114.B1 lv.hdisk121.B1 lv.hdisk111.B1 lv.hdisk109.B1 lv.hdisk107.B1	**ssa8:** hdisk107, hdisk109, hdisk111, hdisk112, hdisk113, hdisk114, hdisk115, hdisk116, hdisk117, hdisk118, hdisk119, hdisk121
node9	lv.hdisk161.B1 lv.hdisk154.B1 lv.hdisk166.B1 lv.hdisk162.B1 lv.hdisk163.B1 lv.hdisk168.B1 lv.hdisk169.B1 lv.hdisk160.B1 lv.hdisk159.B1 lv.hdisk158.B1 lv.hdisk155.B1 lv.hdisk164.B1	**ssa9:** hdisk154, hdisk155, hdisk158, hdisk159, hdisk160, hdisk161, hdisk162, hdisk163, hdisk164, hdisk166, hdisk168, hdisk169

node10	lv.hdisk190.B1 lv.hdisk189.B1 lv.hdisk186.B1 lv.hdisk188.B1 lv.hdisk187.B1 lv.hdisk197.B1 lv.hdisk195.B1 lv.hdisk194.B1 lv.hdisk192.B1 lv.hdisk196.B1 lv.hdisk200.B1 lv.hdisk198.B1	**ssa10:** hdisk186, hdisk187, hdisk188, hdisk189, hdisk190, hdisk192, hdisk194, hdisk195, hdisk196, hdisk197, hdisk198, hdisk200
node11	lv.hdisk102.B1 lv.hdisk90.B1 lv.hdisk96.B1 lv.hdisk104.B1 lv.hdisk95.B1 lv.hdisk105.B1 lv.hdisk94.B1 lv.hdisk101.B1 lv.hdisk100.B1 lv.hdisk103.B1 lv.hdisk93.B1 lv.hdisk92.B1	**ssa11:** hdisk90, hdisk92, hdisk93, hdisk94, hdisk95, hdisk96, hdisk100, hdisk101, hdisk102, hdisk103, hdisk104, hdisk105

TPCDINDEX

Data Tablespace for TPC-D database

Size: 154GB

Node:	**Logical Volumes:**	**Physical Volumes:**
node0	lv.hdisk13.C1 lv.hdisk25.C1 lv.hdisk12.C1 lv.hdisk20.C1 lv.hdisk11.C1 lv.hdisk21.C1 lv.hdisk23.C1 lv.hdisk19.C1 lv.hdisk17.C1 lv.hdisk10.C1 lv.hdisk15.C1 lv.hdisk16.C1	**ssa0**: hdisk10, hdisk11, hdisk12, hdisk13, hdisk15, hdisk16, hdisk17, hdisk19, hdisk20, hdisk21, hdisk23, hdisk25
node1	lv.hdisk31.C1 lv.hdisk37.C1 lv.hdisk26.C1 lv.hdisk40.C1 lv.hdisk41.C1 lv.hdisk39.C1 lv.hdisk27.C1 lv.hdisk33.C1 lv.hdisk29.C1 lv.hdisk36.C1 lv.hdisk34.C1 lv.hdisk28.C1	**ssa1**: hdisk26, hdisk27, hdisk28, hdisk29, hdisk31, hdisk33, hdisk34, hdisk36, hdisk37, hdisk39, hdisk40, hdisk41
node2	lv.hdisk68.C1 lv.hdisk64.C1 lv.hdisk65.C1 lv.hdisk73.C1 lv.hdisk59.C1 lv.hdisk72.C1 lv.hdisk60.C1 lv.hdisk69.C1 lv.hdisk58.C1 lv.hdisk71.C1 lv.hdisk62.C1 lv.hdisk63.C1	**ssa2**: hdisk58, hdisk59, hdisk60, hdisk62, hdisk63, hdisk64, hdisk65, hdisk68, hdisk69, hdisk71, hdisk72, hdisk73
node3	lv.hdisk123.C1 lv.hdisk137.C1 lv.hdisk134.C1 lv.hdisk125.C1 lv.hdisk129.C1 lv.hdisk131.C1 lv.hdisk122.C1 lv.hdisk126.C1 lv.hdisk132.C1 lv.hdisk136.C1 lv.hdisk124.C1 lv.hdisk128.C1	**ssa3**: hdisk122, hdisk123, hdisk124, hdisk125, hdisk126, hdisk128, hdisk129, hdisk131, hdisk132, hdisk134, hdisk136, hdisk137
node4	lv.hdisk75.C1 lv.hdisk81.C1 lv.hdisk89.C1 lv.hdisk82.C1 lv.hdisk88.C1 lv.hdisk76.C1 lv.hdisk83.C1 lv.hdisk78.C1 lv.hdisk74.C1 lv.hdisk85.C1 lv.hdisk84.C1 lv.hdisk86.C1	**ssa4**: hdisk74, hdisk75, hdisk76, hdisk78, hdisk81, hdisk82, hdisk83, hdisk84, hdisk85, hdisk86, hdisk88, hdisk89
node5	lv.hdisk146.C1 lv.hdisk139.C1 lv.hdisk152.C1 lv.hdisk142.C1 lv.hdisk151.C1 lv.hdisk153.C1 lv.hdisk140.C1 lv.hdisk143.C1 lv.hdisk149.C1 lv.hdisk150.C1 lv.hdisk144.C1 lv.hdisk141.C1	**ssa5**: hdisk139, hdisk140, hdisk141, hdisk142, hdisk143, hdisk144, hdisk146, hdisk149, hdisk150, hdisk151, hdisk152, hdisk153
node6	lv.hdisk175.C1 lv.hdisk178.C1 lv.hdisk174.C1 lv.hdisk180.C1 lv.hdisk183.C1 lv.hdisk185.C1 lv.hdisk172.C1 lv.hdisk177.C1 lv.hdisk176.C1 lv.hdisk179.C1 lv.hdisk173.C1 lv.hdisk170.C1	**ssa6:** hdisk170, hdisk172, hdisk173, hdisk174, hdisk175, hdisk176, hdisk177, hdisk178, hdisk179, hdisk180, hdisk183, hdisk185
node7	lv.hdisk42.C1 lv.hdisk51.C1 lv.hdisk53.C1 lv.hdisk52.C1 lv.hdisk43.C1 lv.hdisk49.C1 lv.hdisk47.C1 lv.hdisk56.C1 lv.hdisk55.C1 lv.hdisk50.C1 lv.hdisk48.C1 lv.hdisk46.C1	**ssa7:** hdisk42, hdisk43, hdisk46, hdisk47, hdisk48, hdisk49, hdisk50, hdisk51, hdisk52, hdisk53, hdisk55, hdisk56
node8	lv.hdisk116.C1 lv.hdisk112.C1 lv.hdisk113.C1 lv.hdisk118.C1 lv.hdisk117.C1 lv.hdisk119.C1 lv.hdisk115.C1 lv.hdisk114.C1 lv.hdisk121.C1 lv.hdisk111.C1 lv.hdisk109.C1 lv.hdisk107.C1	**ssa8:** hdisk107, hdisk109, hdisk111, hdisk112, hdisk113, hdisk114, hdisk115, hdisk116, hdisk117, hdisk118, hdisk119, hdisk121
node9	lv.hdisk161.C1 lv.hdisk154.C1 lv.hdisk166.C1 lv.hdisk162.C1 lv.hdisk163.C1 lv.hdisk168.C1 lv.hdisk169.C1 lv.hdisk160.C1 lv.hdisk159.C1 lv.hdisk158.C1 lv.hdisk155.C1 lv.hdisk164.C1	**ssa9:** hdisk154, hdisk155, hdisk158, hdisk159, hdisk160, hdisk161, hdisk162, hdisk163, hdisk164, hdisk166, hdisk168, hdisk169

node10	lv.hdisk190.C1 lv.hdisk189.C1 lv.hdisk186.C1 lv.hdisk188.C1 lv.hdisk187.C1 lv.hdisk197.C1 lv.hdisk195.C1 lv.hdisk194.C1 lv.hdisk192.C1 lv.hdisk196.C1 lv.hdisk200.C1 lv.hdisk198.C1	**ssa10:** hdisk186, hdisk187, hdisk188, hdisk189, hdisk190, hdisk192, hdisk194, hdisk195, hdisk196, hdisk197, hdisk198, hdisk200
node11	lv.hdisk102.C1 lv.hdisk90.C1 lv.hdisk96.C1 lv.hdisk104.C1 lv.hdisk95.C1 lv.hdisk105.C1 lv.hdisk94.C1 lv.hdisk101.C1 lv.hdisk100.C1 lv.hdisk103.C1 lv.hdisk93.C1 lv.hdisk92.C1	**ssa11:** hdisk90, hdisk92, hdisk93, hdisk94, hdisk95, hdisk96, hdisk100, hdisk101, hdisk102, hdisk103, hdisk104, hdisk105

5.3 Mapping of Partitions/Replications

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

See Table 1.

5.4 Implementation of RAID

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

RAID level 1 (mirroring) was used across all database tables, temporary tables, indexes and recovery logs.

5.5 DBGEN Modifications

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

The standard distribution DBGEN version 1.3.1 was used for the database population.

5.6 Table Contents

The contents of the first 10 rows of each table in the test database must be disclosed.

Appendix C.2 , "Test Database Table Contents" lists the contents of the first 10 rows of each table in the test database.

5.7 Database Loading

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

The Numerical Quantities Summary contains the database load times for the system tested in this full disclosure report.

5.8 Data Storage Ratio

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

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

Disk Type	# of Disks	Space per Disk	Sub-Total Disk Space	Database Size	Data Storage Ratio
F/W SCSI	7	8,672MB	60704MB		
SSA 4.5GB	192	4,288MB	823,296MB		
SSA 9.1GB	16	8,672MB	138,752MB		
Total			1,022,752MB	100GB	10.22

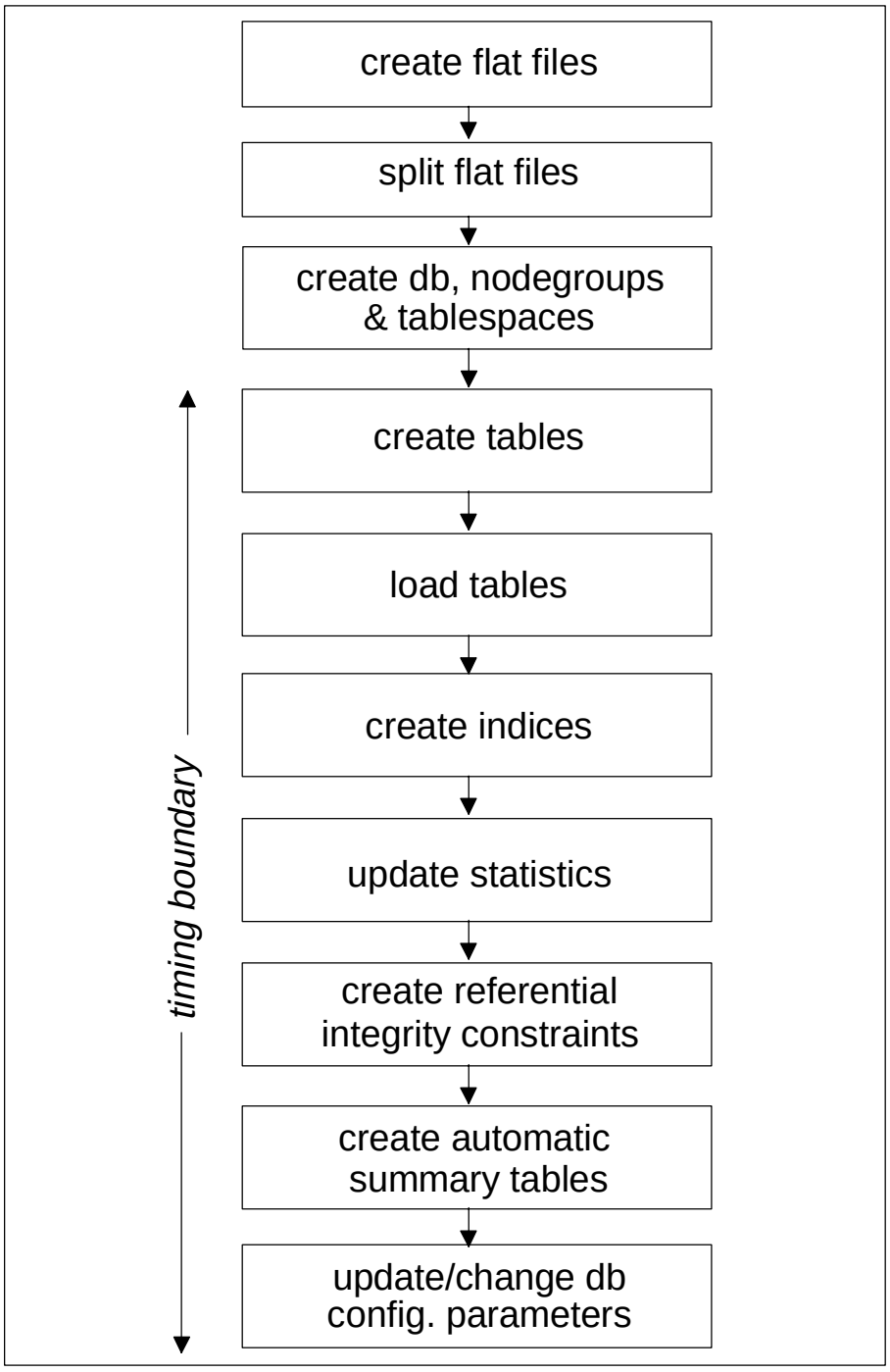
5.9 Details of Database Loading

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

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

The NATION and REGION tables were created on nodegroup 1 and then loaded from dbgen output. The other tables were loaded on all of the nodes, as depicted in the following figure.

Database Load Procedure



6.0 Clause 5: Performance Metrics and Execution Rules

6.1 Power Test

6.1.1 Implementation

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

The following steps are performed prior to running the power test:

1. The database manager is stopped. This clears any DB2 buffers from memory.
2. The system is rebooted.
3. The database manager is restarted.

The power test is then initiated. A warmup run is not used. The power test comprises a single stream that runs the update function UF1, followed by the queries using the sequencing given by stream number 0 in Appendix A of the TPC-D spec, followed by the update function UF2. The tpcdbatch program (see description of the Implementation Specific Layer in the TPC-D specification) is used to run all of the queries.

6.1.2 Timing Intervals

The timing intervals (see Clause 5.3.6) for each query of the measured set (i.e., the query set that follows the warmup set, see Step 4 of Clause 5.3.2.2) and for both update functions must be reported for the power test.

The Numerical Quantities Summary contains the timing intervals for the power test.

6.2 Throughput Test

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

Five concurrent query streams were executed for the throughput test. The Numerical Quantities Summary contains the number of streams and timings.

6.2.1 Stream Times

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.

6.2.2 Measurement Interval

The total elapsed time for 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.

6.2.3 Update Functions

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

The Numerical Quantities Summary contains the timings for the update functions.

6.2.4 Timing Intervals

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 Numerical Quantities Summary contains the timing intervals for the Power test and for each of the five query streams for the Throughput test.

6.3 Performance Metrics

The computer 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 price performance metric for the system reported in this document.

6.4 Reproducibility

A description of the method used to determine the reproducibility of the measurement results must be reported. This must include the performance metrics (QppD, QthD, and QphD) from the reproducibility runs (see Clause 5.4.6).

Two consecutive runs were performed. The following table contains the reproducibility metrics for the system reported in this document.

	QppD@100GB	QthD@100GB	QphD@100GB
Run 1	4,319.8	1,090.2	2,170.1
Run 2	4,226.5	1,092.6	2,148.9

7.0 Clause 6: SUT and Driver Implementation

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. "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 update 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 get the stream number for each of the streams, and the SQL file specific to that stream num ber as the queries to execute. The update function is initiated as a separate call to tpcdbatch with the SQL script for the update functions and the total number of query streams specified.

7.2 Implementation Specific Layer

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

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

A separate instance of tpcdbatch in 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 update stream, as well as an input file containing the 17 queries and/or update 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 update 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 update functions, during the database build all data is first split for each node using the db2split utility. For UF1, the data for each node is further split into n equal portions for both the lineitem and orders tables taking care that the records for the same orderkey remain in the same set. For UF2, the data for each node is further split into m equal portions. During the run, when tpcdbatch encounters a call to execute UF1, it first calls a shell script which loads these n sets of data into n sets of temporary tables (one each for lineitem and orders). Then tpcdbatch forks off n children to do an insert with subselect into the original lineitem and orders tables. When tpcdbatch encounters a call to execute UF2, 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.

8.0 Clause 7: Pricing-Related Items

8.1 Hardware and Programs Used

A detailed list of the 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) must also be reported.

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

8.2 Five Year Cost of System Configuration

The total 5 year price of the entire configuration must be reported, including: hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.

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

The basis for the discounts :

Mid-Range Service Option (MRSO): The Mid-Range Service Option provides a maintenance service discount to those IBM RS/6000 customers who have installed effective system management controls (problem determination, problem reporting and other processes) and who will commit to an IBM maintenance service coverage period. For this TPC-D report, the IBM configuration qualifies for a seventeen percent discount of the monthly maintenance charges.

Extended Maintenance Option (EMO): The Extended Maintenance Option (EMO) provides reduced maintenance charges for newly purchased RS/6000 machine types. Reduction in charges is achieved through pre-payment and elimination of periodic billing. The discount percentage is based on the term of prepayment which is from 12 to 60 months. EMO may be combined with the Mid-Range Service Option. The EMO discount is applied to the net monthly IBM maintenance charges after the MRSO discount is taken. For this report, the selected prepayment term is one year and yields a seventeen percent discount.

IBM Software Revenue Discount: IBM Software Revenue Discount provide discounts on IBM software based upon the total list price value. For DB2 UDB EEE 5.2 the discount is 21%.

IBM Dollar Volume Discount: IBM Dollar Volume Discount provides discounts for RS/6000 hardware and software based upon the total list price value. For the RS/6000 Enterprise Server S70 Advanced used in this report, the Revenue Discount is 30%.

8.3 Availability Dates

The committed delivery date for general availability (availability date) of products used in the price calculations must be reported. When the priced system includes products with different availability dates, 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 (see Clause 7.3.1.3).

The hardware and system software used in this test are generally available at the time of publication. The availability date for the database software is December 31, 1998.

9.0 Clause 9: Audit Items

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

The audit was conducted by François Raab, who can be contacted at the address below.

Information Paradigm

1373 N. Franklin St.

Colorado Springs

Colorado 80903-2527

(719)-473-7555

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

Appendix A: Tunable Parameters

* Denotes changes to default parameter

A.1 DB2 Database TPCD Configuration

```
      Database Configuration for Database TPCD

 Database configuration release level                  = 0x0800
 Database release level                                = 0x0800

 Database territory                                    = C
 Database code page                                    = 819
 Database code set                                     = ISO8859-1
 Database country code                                 = 1

 Directory object name                  (DIR_OBJ_NAME) =
 Discovery support for this database     (DISCOVER_DB) = ENABLE

 Degree of parallelism                    (DFT_DEGREE) = 1
*Default query optimization class         (DFT_QUERYOPT) = 7
 Continue upon arithmetic exceptions   (DFT_SQLMATHWARN) = NO
 Number of frequent values retained     (NUM_FREQVALUES) = 5
 Number of quantiles retained            (NUM_QUANTILES) = 20

 Backup pending                                        = NO

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

 Multi-page file allocation enabled                    = NO

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

 Datalink Access Token Expiry Interval (sec) (DL_EXPINT) = 60
 Datalink Number of Copies               (DL_NUM_COPIES) = 1
 Datalink Number of Backups              (DL_NUM_BACKUP) = 1
 Datalink Time after Drop (days)          (DL_TIME_DROP) = 1

*Database heap (4KB)                           (DBHEAP) = 2400
 Catalog cache size (4KB)              (CATALOGCACHE_SZ) = 64
*Log buffer size (4KB)                         (LOGBUFSZ) = 512
*Utilities heap size (4KB)                (UTIL_HEAP_SZ) = 6000
*Buffer pool size (4KB)                        (BUFFPAGE) = 100000
 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) = 3500

*Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 384

*Sort list heap (4KB)                          (SORTHEAP) = 10000
*SQL statement heap (4KB)                      (STMTHEAP) = 8192
*Default application heap (4KB)            (APPLHEAPSZ) = 384
 Package cache size (4KB)                   (PCKCACHESZ) = (MAXAPPLS*8)
 Statistics heap size (4KB)               (STAT_HEAP_SZ) = 4384

 Interval for checking deadlock (ms)         (DLCHKTIME) = 10000
*Percent. of lock lists per application       (MAXLOCKS) = 6
 Lock timeout (sec)                         (LOCKTIMEOUT) = -1

*Changed pages threshold                (CHNGPGS_THRESH) = 30
*Number of asynchronous page cleaners   (NUM_IOCLEANERS) = 12
*Number of I/O servers                   (NUM_IOSERVERS) = 13
 Index sort flag                            (INDEXSORT) = YES
 Sequential detect flag                     (SEQDETECT) = YES
 Default prefetch size (4KB)            (DFT_PREFETCH_SZ) = 32

 Default number of containers                          = 1
 Default tablespace extentsize (4KB)     (DFT_EXTENT_SZ) = 32

*Max number of active applications            (MAXAPPLS) = 365
 Average number of active applications       (AVG_APPLS) = 1
*Max DB files open per application             (MAXFILOP) = 1024

*Log file size (4KB)                          (LOGFILSIZ) = 3500
*Number of primary log files                 (LOGPRIMARY) = 50
*Number of secondary log files                (LOGSECOND) = 5
 Changed path to log files                   (NEWLOGPATH) =
 Path to log files                                        =
/node0vg.log/NODE0000/
 Next active log file                                     = S0000058.LOG
 First active log file                                    = S0000058.LOG

 Group commit count                           (MINCOMMIT) = 1
*Percent log file reclaimed before soft chckpt (SOFTMAX) = 1000
*Log retain for recovery enabled              (LOGRETAIN) = ON
 User exit for logging enabled                 (USEREXIT) = OFF

 Auto restart enabled                        (AUTORESTART) = ON
 Index re-creation time                         (INDEXREC) = SYSTEM
(RESTART)
 Default number of loadrec sessions     (DFT_LOADREC_SES) = 1
 Recovery history retention (days)      (REC_HIS_RETENTN) = 366

 ADSM management class                   (ADSM_MGMTCLASS) =
 ADSM node name                           (ADSM_NODENAME) =
 ADSM owner                                  (ADSM_OWNER) =
 ADSM password                            (ADSM_PASSWORD) =
```

A.2 DB2 Database Manager Configuration

```
      Database Manager Configuration

     Node type = Partitioned Database Server with local and remote
clients

 Database manager configuration release level            = 0x0800

*CPU speed (millisec/instruction)             (CPUSPEED) = 2.097996e-06
*Communications bandwidth (MB/sec)      (COMM_BANDWIDTH) = 2.000000e+00

*Max number of concurrently active databases     (NUMDB) = 1
 Transaction processor monitor name        (TP_MON_NAME) =

 Default charge-back account           (DFT_ACCOUNT_STR) =

 Java Development Kit 1.1 installation path (JDK11_PATH) =

*Diagnostic error capture level              (DIAGLEVEL) = 3
 Diagnostic data directory path               (DIAGPATH) =

 Default database monitor switches
   Buffer pool                          (DFT_MON_BUFPOOL) = OFF
   Lock                                    (DFT_MON_LOCK) = OFF
   Sort                                    (DFT_MON_SORT) = OFF
   Statement                               (DFT_MON_STMT) = OFF
   Table                                  (DFT_MON_TABLE) = OFF
   Unit of work                            (DFT_MON_UOW) = OFF

 SYSADM group name                          (SYSADM_GROUP) = STAFF
 SYSCTRL group name                        (SYSCTRL_GROUP) =
 SYSMAINT group name                      (SYSMAINT_GROUP) =

 Database manager authentication        (AUTHENTICATION) = SERVER
 Cataloging allowed without authority   (CATALOG_NOAUTH) = NO
 Trust all clients                      (TRUST_ALLCLNTS) = YES
 Trusted client authentication          (TRUST_CLNTAUTH) = CLIENT

 Default database path                        (DFTDBPATH) = /home/tpcd

 Database monitor heap size (4KB)          (MON_HEAP_SZ) = 48
 UDF shared memory set size (4KB)           (UDF_MEM_SZ) = 256
 Java Virtual Machine heap size (4KB)     (JAVA_HEAP_SZ) = 512
 Audit buffer size (4KB)                  (AUDIT_BUF_SZ) = 0

 Backup buffer default size (4KB)            (BACKBUFSZ) = 1024
 Restore buffer default size (4KB)           (RESTBUFSZ) = 1024

*Sort heap threshold (4KB)                  (SHEAPTHRES) = 150000

 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
 DRDA services heap size (4KB)            (DRDA_HEAP_SZ) = 128

 Priority of agents                           (AGENTPRI) = SYSTEM
 Max number of existing agents               (MAXAGENTS) = 400
*Agent pool size                         (NUM_POOLAGENTS) = 10
*Initial number of agents in pool        (NUM_INITAGENTS) = 10
 Max number of coordinating agents      (MAX_COORDAGENTS) = (MAXAGENTS -
NUM_INITAGENTS)
 Max no. of concurrent coordinating agents  (MAXCAGENTS) =
MAX_COORDAGENTS

 Keep DARI process                            (KEEPDARI) = YES
 Max number of DARI processes                  (MAXDARI) =
MAX_COORDAGENTS

 Index re-creation time                       (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) =
 APPC Transaction program name                (TPNAME) =
 IPX/SPX File server name                   (FILESERVER) =
 IPX/SPX DB2 server object name             (OBJECTNAME) =
 IPX/SPX Socket number                      (IPX_SOCKET) = 879E

 Discovery mode                               (DISCOVER) = SEARCH
 Discovery communication protocols        (DISCOVER_COMM) =
 Discover server instance                 (DISCOVER_INST) = ENABLE

 Directory services type                      (DIR_TYPE) = NONE
 Directory path name                     (DIR_PATH_NAME) =
/.:/subsys/database/
 Directory object name                    (DIR_OBJ_NAME) =
 Routing information object name         (ROUTE_OBJ_NAME) =
 Default client comm. protocols          (DFT_CLIENT_COMM) =

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

*No. of int. communication buffers(4KB)(FCM_NUM_BUFFERS) = 16384
*Number of FCM request blocks               (FCM_NUM_RQB) = 2048
 Number of FCM connection entries       (FCM_NUM_CONNECT) = (FCM_NUM_RQB
* 0.75)
 Number of FCM message anchors          (FCM_NUM_ANCHORS) = (FCM_NUM_RQB
* 0.75)

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

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

A.3 DB2 Registry Settings

```
DB2VECTORSPACE=32000
DB2_LIKE_VARCHAR=Y
DB2_CORRELATED_PREDICATES=Y
DB2_VECTOR=Y
DB2_HASH_JOIN=Y
DB2_BINSORT=yes
DB2MEMMAXFREE=8000000
DB2MEMDISCLAIM=yes
DB2_RR_TO_RS=YES
DB2_FORCE_FCM_BP=Y
DB2OPTIONS=-t -v +c
```

A.4 DB2 Environment Variables

```
#
#############################################################################
#
# Licensed Materials - Property of IBM
#
# 5648-A30(C) COPYRIGHT International Business Machines Corp. 1993, 1997
#
# 5648-A32(C) COPYRIGHT International Business Machines Corp. 1993, 1997
#
# 5648-A29(C) COPYRIGHT International Business Machines Corp. 1993, 1997
#
# 5648-A34(C) COPYRIGHT International Business Machines Corp. 1993, 1997
#
# All Rights Reserved
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
#############################################################################
#
# NAME:     db2profile
#
# FUNCTION: This script sets up a default database environment for
#           Bourne shell or Korn shell users.
#
# USAGE:    . db2profile
#           This script can either be invoked directly as above or
#           it can be added to the user's .profile file so that the
#           database environment is established during login.
#           A user may also copy this script into their directory
#           structure and customize it.
#
#
#############################################################################

# Default DB2 product directory
DB2DIR="/usr/lpp/db2_05_00"

#----------------------------------------------------------------------------
# DB2INSTANCE [Default null, values: Any valid instance name]
# Specifies the instance that is active by default.
#----------------------------------------------------------------------------
DB2INSTANCE=tpcd
export DB2INSTANCE

INSTHOME=/home/tpcd

#----------------------------------------------------------------------------
# Add the directories:
#   INSTHOME/sqllib/bin  - database executables
#   INSTHOME/sqllib/adm  - sysadm executables
#   INSTHOME/sqllib/misc - miscellaneous utilities
# to the user's PATH.
#----------------------------------------------------------------------------
PATH=${PATH}:${INSTHOME}/sqllib/bin:${INSTHOME}/sqllib/adm
PATH=${PATH}:${INSTHOME}/sqllib/misc
export PATH
```

A.5 AIX Parameters

```
  keylock   normal  State of system keylock at boot time        False
  maxbuf    200   Maximum number of pages in block I/O BUFFER CACHE True
  maxmbuf   0     Maximum Kbytes of real memory allowed for MBUFS   True
* maxuproc  40000 Maximum number of PROCESSES allowed per user      True
  autorestart false Automatically REBOOT system after a crash       True
  iostat    true    Continuously maintain DISK I/O history          True
  realmem  16777216 Amount of usable physical memory in Kbytes     False
  conslogin enable  System Console Login                           False
  fwversion IBM,19980825 (B) Firmware version and revision levels  False
  maxpout   0       HIGH water mark for pending write I/Os per file True
  minpout   0       LOW water mark for pending write I/Os per file  True
  fullcore  false   Enable full CORE dump                          True
  pre430core false  Use pre-430 style CORE dump                    True
  rtasversion  1    Open Firmware RTAS version                     False
  modelname IBM,7017-S7A  Machine name                             False
  systemid  IBM,011092371 Hardware system identifier               False
  boottype  disk    N/A                                            False
  SW_dist_intr false Enable SW distribution of interrupts          True
* lvm_bufcnt 16
* maxspin    -1
```

A.6 Compiler Options - Makefile.pe

```
#############################################################################
# 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
#############################################################################

#LOCAL=tpcd

BASE=$(HOME)/sqllib
COMPILE_FLAGS= -c -DSQLAIX -I$(BASE)/include -O2 -qmaxmem=-1
# if using an installed db2 image use the 2nd link_flags value
LINK_FLAGS= -o $@ -L$(BASE)/lib -ldb2
COMPILER=xlC
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 2>/dev/null

all : tpcdbatch

tpcdbatch.c : tpcdbatch.sqc
         @echo '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
tpcdbatch : tpcdbatch.c
            $(COMPILER) $(COMPILE_FLAGS) $@.c
            $(COMPILER) $(LINK_FLAGS) $@.o
```

Appendix B: Database Build Scripts

B.1 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';

# 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 ;

# verify that necessary environment variables for building the database
# are present.  Default those that aren't necessary
require "version";
$instance=$ENV{"DB2INSTANCE"};
if (length($ENV{"TPCD_PLATFORM"}) <= 0)
{
  die "TPCD_PLATFORM environment variable not set\n";
}
$platform=$ENV{"TPCD_PLATFORM"};
if ( $platform eq "nt" )
{
    $sep="&";
}
else
{
    $sep=";";
}
if ((length($ENV{"TPCD_BUILD_STAGE"}) <= 0) ||
($ENV{"TPCD_BUILD_STAGE"}) eq "NULL" )
{
    $ENV{"TPCD_BUILD_STAGE"} = "ALL";
}
if (length($ENV{"TPCD_PRODUCT"}) <= 0)
{
    die "Must set TPCD_PRODUCT env't var.\n";
}
if ( length($ENV{"TPCD_DBNAME"}) <= 0 )
{
  die "TPCD_DBNAME environment variable not set\n";
}
if (length($ENV{"TPCD_MODE"}) <= 0)
{
  die "TPCD_MODE environment variable not set - uni/smp/mln \n";
}
if (length($ENV{"TPCD_SF"}) <= 0)
{
  die "TPCD_SF environment variable not set\n";
}
if (length($ENV{"TPCD_DBPATH"}) <= 1)
{
    # if no db pathname specified, build the db in the home directory
    if ( $platform eq "aix" )
    {
        $ENV{"TPCD_DBPATH"} = $ENV{"HOME"};
    }
    elsif ( $platform eq "nt" )
    {
        $ENV{"TPCD_DBPATH"} = $ENV{"HOMEDRIVE"};
    }
    else
    {
        die "platform $platform not supported yet\n";
    }
}
if (length($ENV{"TPCD_DDLPATH"}) <= 0)
{
    # if no db pathname specified, use default
  $ENV{"TPCD_DBPATH"} =
"/afs/tor/groups/dbp/perf/benchmark/tpcd/ddl/vanilla";
}
if (length($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_RUNSTATSHORT"}) <= 0)
{
    $ENV{"TPCD_RUNSTATSHORT"} = "NULL";
}
if (length($ENV{"TPCD_ADD_RI"}) <= 0)
{
    $ENV{"TPCD_ADD_RI"} = "NULL";
}
if (length($ENV{"TPCD_AST"}) <= 0)
{
    $ENV{"TPCD_AST"} = "NULL";
}
if ( ($ENV{"TPCD_INPUT"}) eq "NULL" )
{
    if (length($ENV{"TPCD_DBGEN"}) <= 0)
    {
        die "Must set TPCD_DBGEN if pregenerated flatfiles are not
provided (TPCD_INPUT=NULL)\n";
    }
}
if ( $qual eq "QUAL" )
{
    if ( ($ENV{"TPCD_QUAL_INPUT"}) eq "NULL" )
    {
        if (length($ENV{"TPCD_DBGEN"}) <= 0)
        {
            die "Must set TPCD_DBGEN if pregenerated flatfiles are not
provided (TPCD_QUAL_INPUT=NULL)\n";
        }
}
}
if (length($ENV{"TPCD_TEMP"}) <= 1)
{
    $ENV{"TPCD_TEMP"} = "/u/$instance/sqllib/tmp";
}
if (length($ENV{"TPCD_SORTBUF"}) <= 0)
{
    $ENV{"TPCD_SORTBUF"} = 4096;
}
if (length($ENV{"TPCD_LOAD_PARALLELISM"}) <= 0)
{
    $ENV{"TPCD_LOAD_PARALLELISM"} = 0;
}
if (length($ENV{"TPCD_LOADSTATS"}) <= 0)
{
    $ENV{"TPCD_LOADSTATS"} = "no";
}
if (length($ENV{"TPCD_COPY_DIR"}) <= 0)
{
    $ENV{"TPCD_COPY_DIR"} = "${delim}dev${delim}null";
}
if (length($ENV{"TPCD_FASTPARSE"}) <= 0)
{
    $ENV{"TPCD_FASTPARSE"} = "no";
}
if (length($ENV{"TPCD_BACKUP_DIR"}) <= 0)
{
    $ENV{"TPCD_BACKUP_DIR"} = "${delim}dev${delim}null";
}
if (length($ENV{"TPCD_LOG"}) <= 0)
{
    $ENV{"TPCD_LOG"} = "no";
}
if (length($ENV{"TPCD_LOGPRIMARY"}) <= 0)
{
    $ENV{"TPCD_LOGPRIMARY"} = "NULL";
}
if (length($ENV{"TPCD_LOGSECOND"}) <= 0)
{
    $ENV{"TPCD_LOGSECOND"} = "NULL";
}
if (length($ENV{"TPCD_LOGFILSIZ"}) <= 0)
{
    $ENV{"TPCD_LOGFILSIZ"} = "NULL";
}
if (length($ENV{"TPCD_LOG_DIR"}) <= 0)
{
    $ENV{"TPCD_LOG_DIR"} = "NULL";
}
if (length($ENV{"TPCD_CONFIGFILE"}) <= 0)
{
    $ENV{"TPCD_CONFIGFILE"} = "dss.dbconfig";
}
if (length($ENV{"TPCD_DBM_CONFIG"}) <= 0)
{
    $ENV{"TPCD_DBM_CONFIG"} = "NULL";
}
if (length($ENV{"TPCD_MACHINE"}) <= 0)
{
    $ENV{"TPCD_MACHINE"} = "medium";
}
if (length($ENV{"TPCD_SMPDEGREE"}) <= 0)
{
    $ENV{"TPCD_SMPDEGREE"} = 1;
}
if (length($ENV{"TPCD_AGENTPRI"}) <= 0)
{
    $ENV{"TPCD_AGENTPRI"} = NULL;
}
if (length($ENV{"TPCD_ACTIVATE"}) <= 0)
{
    $ENV{"TPCD_ACTIVATE"} = "no";
}
if (length($ENV{"TPCD_AUDIT"}) <= 0)
{
    die "Must set TPCD_AUDIT env't var.  Real audit timing sequence run
if yes\n";
}
if (length($ENV{"TPCD_AUDIT_DIR"}) <= 0)
{
    die "TPCD_AUDIT_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_LOAD_DB2SET_SCRIPT"}) <= 0)
{
    $ENV{"TPCD_LOAD_DB2SET_SCRIPT"}="NULL"
}
if (length($ENV{"TPCD_DB2SET_SCRIPT"}) <= 0)
```

```
{
    $ENV{"TPCD_DB2SET_SCRIPT"}="NULL"
}
if (length($ENV{"TPCD_BUFFERPOOL_DEF"}) <= 0)
{
    $ENV{"TPCD_BUFFERPOOL_DEF"}="NULL"
}

#set up local variables
$tpcdVersion=$ENV{"TPCD_VERSION"};
$buildStage=$ENV{"TPCD_BUILD_STAGE"};
$product=$ENV{"TPCD_PRODUCT"};
$dbname=$ENV{"TPCD_DBNAME"};
$mode=$ENV{"TPCD_MODE"};
$sf=$ENV{"TPCD_SF"};
$sfReal=$sf;            # need a "saved" one for qualification stuff
$dbpath=$ENV{"TPCD_DBPATH"};
$ddlpath=$ENV{"TPCD_DDLPATH"};
$ddl=$ENV{"TPCD_DDL"};
$buffpooldef=$ENV{"TPCD_BUFFERPOOL_DEF"};
$tbspddl=$ENV{"TPCD_TBSP_DDL"};
$indexddl=$ENV{"TPCD_INDEXDDL"};
$extraindex=$ENV{"TPCD_EXTRAINDEX"};
$runstats=$ENV{"TPCD_RUNSTATS"};
$runstatShort=$ENV{"TPCD_RUNSTATSHORT"};
$dbgen=$ENV{"TPCD_DBGEN"};
$input=$ENV{"TPCD_INPUT"};
$earlyindex=$ENV{"TPCD_EARLYINDEX"};
$ldtemp=$ENV{"TPCD_TEMP"};
$sortbuf=$ENV{"TPCD_SORTBUF"};
$load_parallelism=$ENV{"TPCD_LOAD_PARALLELISM"};
$loadstats=$ENV{"TPCD_LOADSTATS"};
$copydir=$ENV{"TPCD_COPY_DIR"};
$fparse=$ENV{"TPCD_FASTPARSE"};
$addRI=$ENV{"TPCD_ADD_RI"};
$astFile=$ENV{"TPCD_AST"};
if ( $fparse eq "yes" )
{
    $fastparse="FASTPARSE";
}
else
{
    $fastparse=" ";
}
$backupdir=$ENV{"TPCD_BACKUP_DIR"};
$logprimary=$ENV{"TPCD_LOGPRIMARY"};
$logsecond=$ENV{"TPCD_LOGSECOND"};
$logfilsiz=$ENV{"TPCD_LOGFILSIZ"};
$log=$ENV{"TPCD_LOG"};
$logDir=$ENV{"TPCD_LOG_DIR"};
$machine=$ENV{"TPCD_MACHINE"};
$configfile=$ENV{"TPCD_CONFIGFILE"};
$dbmconfig=$ENV{"TPCD_DBM_CONFIG"};
$loadconfigfile=$ENV{"TPCD_LOAD_CONFIGFILE"};
$loadDBMconfig=$ENV{"TPCD_LOAD_DBM_CONFIGFILE"};
$smpdegree=$ENV{"TPCD_SMPDEGREE"};
$agentpri=$ENV{"TPCD_AGENTPRI"};
$activate=$ENV{"TPCD_ACTIVATE"};
$RealAudit=$ENV{"TPCD_AUDIT"};
$loadscript=$ENV{"TPCD_LOAD_SCRIPT"};
$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$loadsetScript=$ENV{"TPCD_LOAD_DB2SET_SCRIPT"};
$setScript=$ENV{"TPCD_DB2SET_SCRIPT"};

if ( $RealAudit eq "yes" )
{
    #need some extra parms if this really is an audit
    $user=$ENV{"USER"};
}

# set up override of some parameters to override for qualification
database
if ( $qual eq "QUAL" )
{
    $loadscript=$ENV{"TPCD_LOAD_SCRIPT_QUAL"};
    if ( length($ENV{"TPCD_QUAL_DBNAME"}) <= 0 )
    {
        die "TPCD_QUAL_DBNAME environment variable not set\n";
    }
    $dbname=$ENV{"TPCD_QUAL_DBNAME"};
    $sf=0.100;
    if ( length($ENV{"TPCD_QUAL_DDL"}) <= 0 )
    {
        die "TPCD_QUAL_DDL environment variable not set\n";
    }
    $ddl=$ENV{"TPCD_QUAL_DDL"};
    if ( length($ENV{"TPCD_QUAL_TBSP_DDL"}) <= 0 )
    {
        die "TPCD_QUAL_TBSP_DDL environment variable not set\n";
    }
    $tbspddl=$ENV{"TPCD_QUAL_TBSP_DDL"};
    $input=$ENV{"TPCD_QUAL_INPUT"};
    if ( length($ENV{"TPCD_QUALCONFIGFILE"}) <= 0 )
    {
        die "TPCD_QUALCONFIGFILE environment variable not set\n";
    }
    $configfile=$ENV{"TPCD_QUALCONFIGFILE"};
    if ( length($ENV{"TPCD_DBM_QUALCONFIG"}) <= 0 )
    {
        die "TPCD_DBM_QUALCONFIG environment variable not set\n";
    }
    $dbmconfig=$ENV{"TPCD_DBM_QUALCONFIG"};
    if ( length($ENV{"TPCD_LOAD_QUALCONFIGFILE"}) <= 0 )
    {
        die "TPCD_LOAD_QUALCONFIGFILE environment variable not set\n";
    }
    $loadconfigfile=$ENV{"TPCD_LOAD_QUALCONFIGFILE"};
    if ( length($ENV{"TPCD_LOAD_DBM_QUALCONFIGFILE"}) <= 0 )
    {
        die "TPCD_LOAD_DBM_QUALCONFIGFILE environment variable not
set\n";
    }
    $loadDBMconfig=$ENV{"TPCD_LOAD_DBM_QUALCONFIGFILE"};
    if (length($ENV{"TPCD_LOG_DIR"}) <= 0)
    {
        $ENV{"TPCD_LOG_DIR"} = "NULL";
    }
    $logDir=$ENV{"TPCD_LOG_QUAL_DIR"};
}

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

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

  if ( $machine eq "small" )
  {
      $buffpage = 5000;
      $sortheap = 3000;
      $sheapthres = 8000;
      $util_heap_sz = 5000;
      $ioservers = 6;
  }
  elsif ( $machine eq "medium" )
  {
      $buffpage = 10000;
      $sortheap =  8000;
      $sheapthres = 20000;
      $util_heap_sz = 10000;
      $ioservers = 10;
  }
  elsif ( $machine eq "big" )
  {
      $buffpage = 30000;
      $sortheap = 20000;
      $sheapthres = 50000;
      $util_heap_sz = 30000;
      $ioservers = 20;
  }
  elsif ( $machine eq "sunsmp" )
  {
      $buffpage = 60000;
      $sortheap = 20000;
      $sheapthres = 80000;
      $util_heap_sz = 30000;
      $ioservers = 80;
  }
  elsif ( $machine eq "eastwood" )
  {
      $buffpage = 80000;
      $sortheap = 50000;
      $sheapthres = 81000;
      $ioclnrs = 4;
      $chngpgs = 30;
      $ioservers = 21;
      $util_heap_sz = 50000;
  }
}
# echo parameter settings to acknowledge what is being built
if ( $buildStage ne "ALL" )
{
    print " ***** STARTING the build process at the $buildStage Stage
*****\n";
}
print "Building a TPC-D Version $tpcdVersion $sf GB database on $dbpath
with: \n";
print "   Mode = $mode \n";
print "   Tablespace ddl in $ddlpath${delim}$tbspddl \n";
if ( $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";
}
print "ASTFILE = $astFile \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";
}
else
{
  if (( $mode eq "mln" ) || ( $mode eq "mpp"))
  {
    die "   Loading script must be specified for mln or mpp
invocations\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";
}

print "Sleeping for 15 seconds to give you a chance to reconsider...\n";
sleep 15;
# set db2options so all usages work
# @db980723wlc
if (( $mode eq "mln" ) || ( $mode eq "mpp"))
{
   $rc=&dodb_noconn("db2set DB2OPTIONS=\\\"-t -v +c\\\"",$all_ln);
}
else
{
   $rc=&dodb_noconn("db2set DB2OPTIONS=\\\"-t -v +c\\\"",$once);
}

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

if ( $platform eq "nt" )
{
   if (( $mode eq "mln" ) || ( $mode eq "mpp"))
   {
      $rc=&dodb_noconn("db2set DB2NTNOCACHE=ON",$all_ln);
   }
   else
   {
      $rc=&dodb_noconn("db2set DB2NTNOCACHE=ON",$once);
   }
}

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

# set the db2 env vars for loading, from the TPCD_LOAD_DB2SET_SCRIPT
script
if ( $loadsetScript ne "NULL" )
{
    $ret=system("${ddlpath}${delim}$loadsetScript");
    if ( $ret != 0 )
    {
        die "failure setting load time db2set parms from $loadsetScript
\n";
    }
}
# stopping and starting db2 before we continue
print "Stopping DB2 ...\n";
$rc=system("db2stop");
if ( $rc < 0 )
{
    die "failure during db2stop rc = $rc \n";
}
print "Starting DB2 ...\n";
$rc=system("db2start");
if ( $rc != 0 )
{
    die "failure during db2start rc = $rc \n";
}

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

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

    # reset the db and dbm configuration before we start
    &dodb_noconn("db2 reset database configuration for
$dbname",$all_ln);
```

```
    &dodb_conn($dbname,"db2 alter bufferpool ibmdefaultbp size -1 $sep
\
             db2 grant connect on database to public $sep            \
             db2 grant dbadm on database to $dbname $sep             \
             db2 commit",$once);
    &dodb_noconn("db2 reset database manager configuration",$once);

    # update the log information first
    # set up the log directory before we do any index creation
#    if ( $logDir ne "NULL" )
#    {
#        &dodb_noconn("db2 update database configuration for $dbname
using newlogpath $logDir",$all_ln);
#    }
    $rc=system("$ddlpath${delim}setnewlogpath.ksh");
    if ( $rc  != 0 )
    {
       die "failed running setnetlogpath.ksh rc=$rc\n";
    }

    $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 $dbname using logbufsz 512";
    &dodb_noconn("$setLogString",$all_ln);


    # setup the load configuration
    &outtime("*** Setting LOAD configuration.");
    if (($loadconfigfile eq "") || ($loadconfigfile eq "NULL"))
    {
       &dodb_noconn("db2 update db cfg for $dbname using buffpage
$buffpage $sep \
             db2 update db cfg for $dbname using sortheap $sortheap $sep
\
             db2 update db cfg for $dbname using num_iocleaners $ioclnrs
$sep \
             db2 update db cfg for $dbname using num_ioservers
$ioservers $sep \
             db2 update db cfg for $dbname using util_heap_sz
$util_heap_sz $sep \
             db2 update db cfg for $dbname using chngpgs_thresh
$chngpgs",
             $all_ln);
       &dodb_noconn("db2 update dbm cfg using sheapthres
$sheapthres",$once);
    }
    else
    {
        &dodb2file_noconn("$ddlpath${delim}$loadconfigfile",$all_ln);
        &dodb2file_noconn("$ddlpath${delim}$loadDBMconfig",$once);
    }
    &dodb_noconn("db2 terminate",$once);
    $rc=system("db2stop");
    if ( $rc != 0 )
    {
        die "failure during db2stop after resetting for load config rc =
$rc \n";
    }
    $rc=system("db2start");
    if ( $rc != 0 )
    {
        die "failure during db2start rc = $rc \n";
    }
}  # end of "CREATE DATABASE" phase
if (( $buildStage eq "ALL" ) || ( $buildStage eq "CRTTBSP" ) ||
    ( $buildStage eq "LOAD" ) ||
    (( $buildStage eq "INDEX" ) && ( $earlyindex eq "yes" )) )
{
    # create the explain tables - these should go into userspace1 since
no
    # other tablespaces exist
    if ( $platform ne "nt" )
    {
        $sqlpath="~${delim}sqllib";
    }
    else
    {
        $sqlpath=$ENV{"DB2PATH"};
    }
    &dodb_conn($dbname,
               "db2 -tvf $sqlpath${delim}misc${delim}EXPLAIN.DDL $sep \
                db2 alter table explain_instance locksize table append
on $sep \
                db2 alter table explain_statement locksize table append
on $sep \
                db2 alter table explain_argument locksize table append
on $sep \
                db2 alter table explain_object locksize table append on
$sep \
                db2 alter table explain_operator locksize table append
on $sep \
                db2 alter table explain_predicate locksize table append
on $sep \
                db2 alter table explain_stream locksize table append
on",
               $once);

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

    }
    # create the tablespaces
    &outtime("*** Ready to start creating the tablespaces");
    &dodb2file($dbname,"$ddlpath${delim}$tbspddl",$once);
    # if we are in audit mode, then we must create the the tablespaces
and
    # tables for the update functions and we must generate the data for
the
    # update functions before we start timing the load.  (All activity
    # on the database after the table creation is started and before the
performance
    # tests are run must be included in load time
    # NOTE: we do not have to do this if we are building the
qualification database
    if (( $RealAudit eq "yes" ) && ( $qual ne "QUAL" ))
    {
        # build the update function staging tables
        if ( $product eq "pe" )
        {
            print "update files for pe v1.2 style are build through
bhatta's scripts\n";
        }
        else
        {
            $rc = system("perl buildUFTbl");
            if ( $rc != 0 )
            {
                die "buildUFTbl failed rc=$rc\n";
            }
            &dodb2file($dbname,"$ddlpath${delim}createUFtbls",$once);
        }
    }
    &outtime("*** Start Load Timing now - starting to create tables");
    &dodb2file($dbname,"$ddlpath${delim}$ddl",$once);

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

    if ( $mode eq "mpp" )
    {
        #need parallel specific
        print "need to figure parallel specific creation of tmp\n";
    }
    mkdir("${delim}tmp${delim}$instance",0777);
}  # end of "CREATE TABLESPACE and TABLES" phase
if (( $buildStage eq "ALL" ) || ( $buildStage eq "CRTTBSP" ) ||
    ( $buildStage eq "LOAD" ) ||
    (( $buildStage eq "INDEX" ) && ( $earlyindex eq "yes" )) )
{
    # do the load stage of the build, or if early index is on do
    # the index creation also
    # setup the load configuration
    if ( $buildStage ne "ALL")
    { # we're restarting a build so reapply the load config
      &outtime("*** Setting LOAD configuration.");
      if (($loadconfigfile eq "") || ($loadconfigfile eq "NULL"))
      {
       &dodb_noconn("db2 update db cfg for $dbname using buffpage
$buffpage $sep \
             db2 update db cfg for $dbname using sortheap $sortheap $sep
\
             db2 update db cfg for $dbname using num_iocleaners $ioclnrs
$sep \
             db2 update db cfg for $dbname using num_ioservers
$ioservers $sep \
             db2 update db cfg for $dbname using util_heap_sz
$util_heap_sz $sep \
             db2 update db cfg for $dbname using chngpgs_thresh
$chngpgs",
             $all_ln);
       &dodb_noconn("db2 update dbm cfg using sheapthres
$sheapthres",$once);
      }
      else
      {
        &dodb2file_noconn("$ddlpath${delim}$loadconfigfile",$all_ln);
        &dodb2file_noconn("$ddlpath${delim}$loadDBMconfig",$once);
      }
      &dodb_noconn("db2 terminate",$once);
      $rc=system("db2stop");
      if ( $rc != 0 )
      {
        die "failure during db2stop after resetting for load config rc =
$rc \n";
      }
      $rc=system("db2start");
      if ( $rc != 0 )
      {
        die "failure during db2start rc = $rc \n";
      }
```

```
    }

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

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

    # start the dbgen and load.....call the specific mode for loading
(uni,smp,mln)
    if (( $mode eq "uni" ) || ( $mode eq "smp" ))
    {
        &outtime("*** Starting the load");
        # call the appropriate dbgen/load for uni/smp
        if (( $loadscript ne "NULL") && ( $loadscript ne "" ))
        {
            &dodb2file_noconn("$ddlpath${delim}$loadscript",$once);
        }
        else
        {
            $rc = system("perl genloaduni $qual");
            if ( $rc != 0 )
            {
                die "genloaduni 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
        $rc = system("$ddlpath${delim}$loadscript $sf");
        if ( $rc != 0 )
        {
            die "doload for $dbname failed rc = $rc\n";
        }
    }
    else
    {
        die "TPCD_MODE not set to one of uni, smp, mln or mpp\n";
    }
}  # end all and load stage (and create index if early index was
specified

if (( $buildStage eq "ALL" ) || ( $buildStage eq "CRTTBSP" ) ||
    ( $buildStage eq "LOAD" ) ||
    ( $buildStage eq "INDEX" ))
{
    if ( $buildStage eq "INDEX")
    { # we're restarting a build so reapply the load config
      &outtime("*** Setting LOAD configuration.");
      if (($loadconfigfile eq "") || ($loadconfigfile eq "NULL"))
      {
       &dodb_noconn("db2 update db cfg for $dbname using buffpage
$buffpage $sep \
            db2 update db cfg for $dbname using sortheap $sortheap $sep
\
            db2 update db cfg for $dbname using num_iocleaners $ioclnrs
$sep \
            db2 update db cfg for $dbname using num_ioservers
$ioservers $sep \
            db2 update db cfg for $dbname using util_heap_sz
$util_heap_sz $sep \
            db2 update db cfg for $dbname using chngpgs_thresh
$chngpgs",
            $all_ln);
       &dodb_noconn("db2 update dbm cfg using sheapthres
$sheapthres",$once);
      }
      else
      {
        &dodb2file_noconn("$ddlpath${delim}$loadconfigfile",$all_ln);
        &dodb2file_noconn("$ddlpath${delim}$loadDBMconfig",$once);
      }
      &dodb_noconn("db2 terminate",$once);
      $rc=system("db2stop");
      if ( $rc != 0 )
      {
        die "failure during db2stop after resetting for load config rc =
$rc \n";
      }
      $rc=system("db2start");
      if ( $rc != 0 )
      {
        die "failure during db2start rc = $rc \n";
      }
    }
    # if indexes haven't been created, do so now
    if ( $earlyindex ne "yes" )
    {
        &outtime("*** Create index started");
        &dodb2file($dbname,"$ddlpath${delim}$indexddl",$once);

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

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

if (( $buildStage eq "ALL" ) ||  ( $buildStage eq "CRTTBSP" ) ||
    ( $buildStage eq "LOAD" ) ||
    ( $buildStage eq "INDEX" ) || ( $buildStage eq "RUNSTATS" ))
{
    if ( $buildStage eq "RUNSTATS")
    { # we're restarting a build so reapply the load config
      &outtime("*** Setting LOAD configuration.");
      if (($loadconfigfile eq "") || ($loadconfigfile eq "NULL"))
      {
       &dodb_noconn("db2 update db cfg for $dbname using buffpage
$buffpage $sep \
            db2 update db cfg for $dbname using sortheap $sortheap $sep
\
            db2 update db cfg for $dbname using num_iocleaners $ioclnrs
$sep \
            db2 update db cfg for $dbname using num_ioservers
$ioservers $sep \
            db2 update db cfg for $dbname using util_heap_sz
$util_heap_sz $sep \
            db2 update db cfg for $dbname using chngpgs_thresh
$chngpgs",
            $all_ln);
       &dodb_noconn("db2 update dbm cfg using sheapthres
$sheapthres",$once);
      }
      else
      {
        &dodb2file_noconn("$ddlpath${delim}$loadconfigfile",$all_ln);
        &dodb2file_noconn("$ddlpath${delim}$loadDBMconfig",$once);
      }
      &dodb_noconn("db2 terminate",$once);
      $rc=system("db2stop");
      if ( $rc != 0 )
      {
        die "failure during db2stop after resetting for load config rc =
$rc \n";
      }
      $rc=system("db2start");
      if ( $rc != 0 )
      {
        die "failure during db2start rc = $rc \n";
      }
    }
    # if statistics not gathered on the load, run runstats  (we have to
run the
    # stats at the same time as the index creation whether it be both
during load,
    # or after load)
    # We need to run the runstats as well if we have specifed an extra
index file
    # for "after load" indexes
    if (( $loadstats eq "no" ) || ( $earlyindex eq "no" ) || (
$extraindex ne "no" ))
    {
        # if loadstats not gathered, then index stats not gathered
either.
        &outtime("*** Runstats started");

        if ( $runstatShort ne "NULL" )
        {
            # we've specified a second runstats file...This runstats
file should do
            # runstats for all table except lineitem. The lineitem
runstats command
            # should be left in the main runstats file.
            if ( $platform eq "aix" )
            {
                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");
    }
    if ( $addRI ne "NULL" )
    {
        &outtime("*** Adding RI contraints started");
        &dodb2file($dbname,"$ddlpath${delim}$addRI",$once);
        &outtime("*** Adding RI contraints completed");
    }
} # end build phaose all/load/index/runstats

#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");
}


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

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

if ( $agentpri ne "NULL" )
{
```

```
    &dodb_noconn("db2 update dbm cfg using AGENTPRI $agentpri",$once);
}

# set the db2 environment variables for running the benchmark
if ( $setScript ne "NULL" )
{
    $ret=system("${ddlpath}${delim}$setScript");
    if ( $ret != 0 )
    {
        die "failure setting runtime db2set parms from $setScript \n";
    }
}
# if logging is enabled, we must take a backup of the database
if ( $log eq "yes" )
{
    &dodb_noconn("db2 update database configuration for $dbname using
LOGRETAIN yes",$all_ln);
    print "\n NOTE:  DO NOT RESET THE DATABASE CONFIGURATION or you will
lose logretain\n";
    &dodb_conn($dbname,"db2 \"select count(*) from
tpcd.region\"",$all_ln);

    if ( $qual eq "QUAL" )
    {
        &outtime("*** Starting the backup");
#need to test parallel specific
        if (( $mode eq "mln" ) || ( $mode eq "mpp"))
        {
            # must back up catalog node first...assume node 00
            $rc=system("db2_all \'}]<<+000< db2 \"backup database
$dbname to $backupdir without prompting\" \' ");
            if ( $rc != 0 )
            {
                die "backup of catalog node failed rc = $rc\n";
            }
            # back up remaining nodes
            $rc=system("db2_all \'}]<<-000< db2 backup database $dbname
to $backupdir without prompting\' ");
            if ( $rc != 0 )
            {
                die "backup of remaining nodes failed rc = $rc\n";
            }
        }
        else
        {
            &dodb_noconn("db2 backup database $dbname to
$backupdir",$once);
        }
        &outtime("*** Finished the backup");
    }
    else
    {
        # This is the test database.  Clause 3.1.4 states that "the test
sponsor is
        # not required to make or have backup copies of the test
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.
        &dodb_noconn("db2bkpen off $dbname",$all_ln);
        #$rc=system("db2bkpen off $dbname");
        #if ( $rc != 0 )
        #{
        #    die "using db2bkpen failed rc = $rc\n";
        #}
    }
}

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

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

if ( $RealAudit eq "yes" )
{
    # if we are in real audit mode then we have to do a number of things
    # set up the audit directory structure and the run directory
structure
    # so that once we have completed the buildtpcd, we are ready to run.
    # first remove any old "update pair number" file so we won't get
prompted
    # doing setupDir
    &rm("$auditDir${delim}$dbname.$user.update.pair.num");
    &rm("$auditDir${delim}tools${delim}tpcd.runsetup");
    system("perl setupDir");
    system("perl setupRun");
    # before we stop the database for the final time
    # if we are in the real audit mode then compile and bind tpcdbatch,
and
    # run dbtables and dbcheck before we print out the final notice that
    # we are ready to run the performance tests
    # if we are building the qualification database then we will bind to
both
    # the dbname database and the qualification database
    $rc = system("perl buildtpcdbatch $qual");
    if ( $rc != 0 )
    {
        die "buildtpcdbatch failed rc=$rc\n";
    }
    if ( $qual eq "QUAL" )
    {
        $verifyType="q";
    }
    else
    {
        $verifyType="t";
    }
    system("perl tablesdb $verifyType");

&dodb2file($dbname,"$auditDir${delim}tools${delim}first10rows.sql",$once
);
}
# stop, restart and activate the database, if necessary
$rc=system("db2stop");
if ( $rc != 0 )
{
    die "failure during db2stop rc = $rc \n";
}

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

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

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

1;
```

B.2 gensplit100GB.ksh

```
#! /bin/ksh

# generate data for each node
# gensplit.ksh

# set up dbgen env't vars
dbgen="/congo/appendix/dbgen";
dssseed="$dbgen/seeds";
dsspath="/dbgen_data";
sf=100;
numchunk=10;
export DSS_DBGEN=$dbgen;
export DSS_CONFIG=$dbgen;
export DSS_SEED=$dssseed;
export DSS_PATH=$dsspath;
export TPCD_SF=$sf;

# customer supplier and part split files will fit in 2GGB file limit
# so don't need multiple chunks
# customer table
# create named pipe for this node's chunk of dbgen
    npipe=$dsspath"/customer.tbl"
    rm $npipe
    mkfifo $npipe
    DSS_OPTIONS="-F -s "$TPCD_SF" -T c"
    #print "options=$DSS_OPTIONS";
    $DSS_DBGEN/dbgen $DSS_OPTIONS &

# run the autosplitter on the file
    db2split -c customer.cfg -i $npipe -o $dsspath/customer

# part table
# create named pipe for this node's chunk of dbgen
    npipe=$dsspath"/part.tbl"
    rm $npipe
    mkfifo $npipe
    DSS_OPTIONS="-F -s "$TPCD_SF" -T P"
    print "options=$DSS_OPTIONS";
    $DSS_DBGEN/dbgen $DSS_OPTIONS &

# run the autosplitter on the file
    db2split -c part.cfg -i $npipe -o $dsspath/part
# supplier table
# create named pipe for this node's chunk of dbgen
   npipe=$dsspath"/supplier.tbl"
   rm $npipe
   mkfifo $npipe
   DSS_OPTIONS="-F -s "$TPCD_SF" -T s"
   print "options=$DSS_OPTIONS";
   $DSS_DBGEN/dbgen $DSS_OPTIONS &

# run the autosplitter on the file
   db2split -c supplier.cfg -i $npipe -o $dsspath/supplier

# partsupp, orders and lineitme split files will be t> 2GB so
# gen them in chunks

# partsupp table
# create named pipe for this node's chunk of dbgen
    npipe=$dsspath"/partsupp.tbl"
    rm $npipe
    mkfifo $npipe
    DSS_OPTIONS="-F -s "$TPCD_SF" -T S"
    #print "options=$DSS_OPTIONS";
    $DSS_DBGEN/dbgen $DSS_OPTIONS &

    # run the autosplitter on the file
    db2split -c partsupp.cfg -i $npipe -o $dsspath/partsupp
    (( i=$i+1 ));
```

```
}
done;

# orders and lineitme table have to be sorted after they are gen/split
# so write them on to /sort_data....they can then be sorted onto the
same
# disks that the other tables are on
dsspath="/sort_data";
export DSS_PATH=$dsspath;

# orders table
(( i=1 ));
while [[ $i -le $numchunk ]];
do {
# create named pipe for this node's chunk of dbgen
    npipe=$dsspath"/order.tbl.$i"
    rm $npipe
    mkfifo $npipe
    DSS_OPTIONS="-F -s "$TPCD_SF" -T O -C 10 -S $i"
    print "options=$DSS_OPTIONS";
    $DSS_DBGEN/dbgen $DSS_OPTIONS &

# run the autosplitter on the file
    db2split -c orders.cfg -i $npipe -o $dsspath/orders.$i
    (( i=$i+1 ));
}
done;

# lineitem table
(( i=1 ));
while [[ $i -le $numchunk ]];
do {
 create named pipe for this node's chunk of dbgen
    npipe=$dsspath"/lineitem.tbl"
    rm $npipe
    mkfifo $npipe
    DSS_OPTIONS="-F -s "$TPCD_SF" -T L -S $i"
    print "options=$DSS_OPTIONS";
    $DSS_DBGEN/dbgen $DSS_OPTIONS &

# run the autosplitter on the file
    db2split -c lineitem.cfg -i $npipe -o $dsspath/lineitem.$i
    (( i=$i+1 ));
}
done;

dsspath="/dbgen_data";
export DSS_PATH=$dsspath;
# nation table
    DSS_OPTIONS="-F -s "$TPCD_SF" -T n"
    print "options=$DSS_OPTIONS";
    $DSS_DBGEN/dbgen $DSS_OPTIONS &
# region table
    DSS_OPTIONS="-F -s "$TPCD_SF" -T r"
    print "options=$DSS_OPTIONS";
    $DSS_DBGEN/dbgen $DSS_OPTIONS &
```

B.3 doload.ksh

```
#! /bin/ksh
#    loadeverything
echo "loading customer at "`date`
/home/tpcd/custom.acme/loadcust.ksh
echo "loading supplier at "`date`
/home/tpcd/custom.acme/loadsupp.ksh
echo "loading part at "`date`
/home/tpcd/custom.acme/loadpart.ksh
echo "loading partsupp at "`date`
/home/tpcd/custom.acme/loadps.ksh
#/home/tpcd/custom.acme/loadps.8w.ksh
echo "loading orders at "`date`
/home/tpcd/custom.acme/loadord.ksh
#/home/tpcd/custom.acme/loadord.8w.ksh
echo "loading lineitem at "`date`
/home/tpcd/custom.acme/loadline.ksh
#/home/tpcd/custom.acme/loadline.8w.ksh

echo "loading nation and region at "`date`

db2 connect to tpcd;
db2 "load from /home/tpcd/HEADERS/nation.tbl of del modified by coldel|
fastparse noheader replace into TPCD.nation nonrecoverable";
db2 commit work;
db2 "load from /home/tpcd/HEADERS/region.tbl of del modified by coldel|
fastparse noheader replace into TPCD.region nonrecoverable";
db2 commit work;
echo "finished loading at "`date`
echo "sanity checking database";
db2 -f sanity.sql
echo "sanity checking done"

db2 connect reset;
db2 terminate;
```

B.4 loadline.ksh

```
#! /bin/ksh
#    my load lineitem
RAHOSTFILE=/home/tpcd/sqllib/db2nodes.cfg db2_all '||"]}typeset -i
ln=##;typeset -Z5 LN5=$ln; path="/dbgen"; db2 connect to tpcd;str="db2
\"load from $path/lineitem.1.${LN5}, $path/lineitem.2.${LN5},
$path/lineitem.3.${LN5}, $path/lineitem.4.${LN5},
$path/lineitem.5.${LN5}, $path/lineitem.6.${LN5},
$path/lineitem.7.${LN5}, $path/lineitem.8.${LN5},
$path/lineitem.9.${LN5}, $path/lineitem.10.${LN5} of del modified by
coldel| fastparse dumpfile /tmp/tpcd/dmpline${ln}  messages
/tmp/tpcd/msgline${ln} remote file /tmp/tpcd/rmtline${ln} replace into
TPCD.lineitem nonrecoverable\" ";print -- "$str";eval "$str";db2 connect
reset;db2 terminate'
```

B.5 loadpart.ksh

```
#! /bin/ksh
#    my load part
RAHOSTFILE=/home/tpcd/sqllib/db2nodes.cfg db2_all '||"]}typeset -i
ln=##;typeset -Z5 LN5=$ln;db2 connect to tpcd;str="db2 \"load from
/dbgen/part.${LN5} of del modified by coldel| fastparse replace into
TPCD.part nonrecoverable\" </dev/null >/tmp/tpcd/loadpart.$LN5
2>&1";print -- "$str";eval "$str";db2 connect reset;db2 terminate'
```

B.6 loadord.ksh

```
#! /bin/ksh
#    my load orders
RAHOSTFILE=/home/tpcd/sqllib/db2nodes.cfg db2_all '||"]}typeset -i
ln=##;typeset -Z5 LN5=$ln;db2 connect to tpcd;str="db2 \"load from
/dbgen/orders.1.${LN5}, /dbgen/orders.2.${LN5}, /dbgen/orders.3.${LN5},
/dbgen/orders.4.${LN5}, /dbgen/orders.5.${LN5}, /dbgen/orders.6.${LN5},
/dbgen/orders.7.${LN5}, /dbgen/orders.8.${LN5}, /dbgen/orders.9.${LN5},
/dbgen/orders.10.${LN5} of del modified by coldel| fastparse dumpfile
/tmp/tpcd/dmpord${ln} messages /tmp/tpcd/msgord${ln} remote file
/tmp/tpcd/rmtord${ln} replace into TPCD.orders nonrecoverable\" ";print
-- "$str";eval "$str";db2 connect reset;db2 terminate'
```

B.7 loadps.ksh

```
#! /bin/ksh
#    my load partsupp
RAHOSTFILE=/home/tpcd/sqllib/db2nodes.cfg db2_all '||"]}typeset -i
ln=##;typeset -Z5 LN5=$ln;db2 connect to tpcd;str="db2 \"load from
/dbgen/partsupp.${LN5} of del modified by coldel| fastparse replace into
TPCD.partsupp nonrecoverable\" </dev/null >/tmp/tpcd/loadps.$LN5
2>&1";print -- "$str";eval "$str";db2 connect reset; db2 terminate'
```

B.8 loadcust.ksh

```
#! /bin/ksh
#    my load customer
RAHOSTFILE=/home/tpcd/sqllib/db2nodes.cfg db2_all '||"]}typeset -i
ln=##;typeset -Z5 LN5=$ln;db2 connect to tpcd;str="db2 \"load from
/dbgen/customer.${LN5} of del modified by coldel| fastparse replace into
TPCD.customer nonrecoverable\" </dev/null >/tmp/tpcd/loadcust.$LN5
2>&1";print -- "$str";eval "$str";db2 connect reset;db2 terminate'
```

B.9 loadsupp.ksh

```
#! /bin/ksh
#    my load supplier
RAHOSTFILE=/home/tpcd/sqllib/db2nodes.cfg db2_all '||"]}typeset -i
ln=##;typeset -Z5 LN5=$ln;db2 connect to tpcd;str="db2 \"load from
/dbgen/supplier.${LN5} of del modified by coldel| fastparse replace into
TPCD.supplier nonrecoverable\" </dev/null >/tmp/tpcd/loadsupp.$LN5
2>&1";print -- "$str";eval "$str";db2 connect reset; db2 terminate'
```

B.10 db2nodes.cfg

```
0 acme 0
1 acme 1
2 acme 2
3 acme 3
4 acme 4
5 acme 5
6 acme 6
7 acme 7
8 acme 8
9 acme 9
10 acme 10
11 acme 11
```

B.11 dss.dbconfig100.acme

```
update db cfg for tpcd using buffpage 100000;
update db cfg for tpcd using sortheap 10000;
update db cfg for tpcd using chngpgs_thresh 30;
update db cfg for tpcd using num_ioservers 13;
update db cfg for tpcd using num_iocleaners 12;
--update db cfg for tpcd using logbufsz 128;
--update db cfg for tpcd using logfilsiz 3500;
--update db cfg for tpcd using logsecond 5;
--update db cfg for tpcd using logprimary 10;
update database configuration for tpcd using dft_queryopt       7;
update database configuration for tpcd using app_ctl_heap_sz  384;
update database configuration for tpcd using maxfilop        1024;
update database configuration for tpcd using applheapsz       384;
update database configuration for tpcd using stmtheap        8192;
update database configuration for tpcd using maxappls         365;
update database configuration for tpcd using dbheap          2400;
update db cfg for tpcd using locklist                        3500;
update db cfg for tpcd using maxlocks                           6;
```

```
update db cfg for tpcd using softmax                          1000;
update db cfg for tpcd using util_heap_sz                     6000;
```

B.12 dss.dbmconfig100.acme

```
update dbm cfg using sheapthres 150000;
update dbm cfg using numdb 1;
update database  manager configuration using fcm_num_buffers 16384;
update database manager configuration using fcm_num_rqb      2048;
update database manager configuration using cpuspeed -1;
update database manager configuration using comm_bandwidth 2.0;
update database manager configuration using num_poolagents 10;
update database manager configuration using num_initagents 10;
update database manager configuration using max_querydegree any;
update database manager configuration using intra_parallel no;
update dbm cfg using diaglevel 3;
```

B.13 dss.ddl100GB.tbsp.acme.12w.mirror.addtmp

```
--CONNECT TO TPCD;

DROP TABLESPACE TPCDTEMP;
-- create the dms managed tablespace
-- 60 lvs of 1600MB each on 60 disks
-- each lv has 100 PPs of size 16MB ==> 2048000 pages
-- Total pages in temp tablespace: 24576000 ==> ~96 GB
CREATE TEMPORARY TABLESPACE TPCDTEMP MANAGED BY database
  USING (

         device '/dev/rlv.hdisk18.A1' 409600,
         device '/dev/rlv.hdisk24.A1' 409600,
         device '/dev/rlv.hdisk22.A1' 409600,
         device '/dev/rlv.hdisk14.A1' 409600,
         device '/dev/rlv.hdisk215.A1' 409600) on NODE (0)
  USING (

         device '/dev/rlv.hdisk35.A1' 409600,
         device '/dev/rlv.hdisk32.A1' 409600,
         device '/dev/rlv.hdisk30.A1' 409600,
         device '/dev/rlv.hdisk38.A1' 409600,
         device '/dev/rlv.hdisk214.A1' 409600) on NODE (1)
  USING (

         device '/dev/rlv.hdisk61.A1' 409600,
         device '/dev/rlv.hdisk70.A1' 409600,
         device '/dev/rlv.hdisk66.A1' 409600,
         device '/dev/rlv.hdisk67.A1' 409600,
         device '/dev/rlv.hdisk205.A1' 409600) on NODE (2)
  USING (

         device '/dev/rlv.hdisk127.A1' 409600,
         device '/dev/rlv.hdisk130.A1' 409600,
         device '/dev/rlv.hdisk135.A1' 409600,
         device '/dev/rlv.hdisk133.A1' 409600,
         device '/dev/rlv.hdisk207.A1' 409600) on NODE (3)
  USING (

         device '/dev/rlv.hdisk79.A1' 409600,
         device '/dev/rlv.hdisk87.A1' 409600,
         device '/dev/rlv.hdisk77.A1' 409600,
         device '/dev/rlv.hdisk80.A1' 409600,
         device '/dev/rlv.hdisk204.A1' 409600) on NODE (4)
  USING (

         device '/dev/rlv.hdisk138.A1' 409600,
         device '/dev/rlv.hdisk145.A1' 409600,
         device '/dev/rlv.hdisk147.A1' 409600,
         device '/dev/rlv.hdisk148.A1' 409600,
         device '/dev/rlv.hdisk202.A1' 409600) on NODE (5)
  USING (

         device '/dev/rlv.hdisk184.A1' 409600,
         device '/dev/rlv.hdisk171.A1' 409600,
         device '/dev/rlv.hdisk182.A1' 409600,
         device '/dev/rlv.hdisk181.A1' 409600,
         device '/dev/rlv.hdisk216.A1' 409600) on NODE (6)
  USING (

         device '/dev/rlv.hdisk44.A1' 409600,
         device '/dev/rlv.hdisk57.A1' 409600,
         device '/dev/rlv.hdisk45.A1' 409600,
         device '/dev/rlv.hdisk54.A1' 409600,
         device '/dev/rlv.hdisk210.A1' 409600) on NODE (7)
  USING (

         device '/dev/rlv.hdisk110.A1' 409600,
         device '/dev/rlv.hdisk120.A1' 409600,
         device '/dev/rlv.hdisk108.A1' 409600,
         device '/dev/rlv.hdisk106.A1' 409600,
         device '/dev/rlv.hdisk208.A1' 409600) on NODE (8)
  USING (

         device '/dev/rlv.hdisk157.A1' 409600,
         device '/dev/rlv.hdisk165.A1' 409600,
         device '/dev/rlv.hdisk156.A1' 409600,
         device '/dev/rlv.hdisk167.A1' 409600,
         device '/dev/rlv.hdisk206.A1' 409600) on NODE (9)
  USING (

         device '/dev/rlv.hdisk201.A1' 409600,
         device '/dev/rlv.hdisk191.A1' 409600,
         device '/dev/rlv.hdisk193.A1' 409600,
         device '/dev/rlv.hdisk199.A1' 409600,
         device '/dev/rlv.hdisk212.A1' 409600) on NODE (10)
  USING (

         device '/dev/rlv.hdisk98.A1' 409600,
         device '/dev/rlv.hdisk91.A1' 409600,
         device '/dev/rlv.hdisk97.A1' 409600,
         device '/dev/rlv.hdisk99.A1' 409600,
         device '/dev/rlv.hdisk211.A1' 409600) on NODE (11)
        EXTENTSIZE 32 PREFETCHSIZE 128;

COMMIT WORK;
drop tablespace tempspace1;

-- create the dms managed tablespace for data
-- 144 lvs of 1072MB each on 144 disks
-- each lv has 67 PPs of size 16MB ==> 274432 pages
-- Total pages in data tablespace: 39518208 ==> ~154 GB
CREATE TABLESPACE TPCDDATA MANAGED BY DATABASE
  USING (
         device '/dev/rlv.hdisk13.B1' 274432,
         device '/dev/rlv.hdisk25.B1' 274432,
         device '/dev/rlv.hdisk12.B1' 274432,
         device '/dev/rlv.hdisk20.B1' 274432,
         device '/dev/rlv.hdisk11.B1' 274432,
         device '/dev/rlv.hdisk21.B1' 274432,
         device '/dev/rlv.hdisk23.B1' 274432,
         device '/dev/rlv.hdisk19.B1' 274432,
         device '/dev/rlv.hdisk17.B1' 274432,
         device '/dev/rlv.hdisk10.B1' 274432,
         device '/dev/rlv.hdisk15.B1' 274432,
         device '/dev/rlv.hdisk16.B1' 274432) on NODE (0)
  USING (
         device '/dev/rlv.hdisk31.B1' 274432,
         device '/dev/rlv.hdisk37.B1' 274432,
         device '/dev/rlv.hdisk26.B1' 274432,
         device '/dev/rlv.hdisk40.B1' 274432,
         device '/dev/rlv.hdisk41.B1' 274432,
         device '/dev/rlv.hdisk39.B1' 274432,
         device '/dev/rlv.hdisk27.B1' 274432,
         device '/dev/rlv.hdisk33.B1' 274432,
         device '/dev/rlv.hdisk29.B1' 274432,
         device '/dev/rlv.hdisk36.B1' 274432,
         device '/dev/rlv.hdisk34.B1' 274432,
         device '/dev/rlv.hdisk28.B1' 274432) on NODE (1)
  USING (
         device '/dev/rlv.hdisk68.B1' 274432,
         device '/dev/rlv.hdisk64.B1' 274432,
         device '/dev/rlv.hdisk65.B1' 274432,
         device '/dev/rlv.hdisk73.B1' 274432,
         device '/dev/rlv.hdisk59.B1' 274432,
         device '/dev/rlv.hdisk72.B1' 274432,
         device '/dev/rlv.hdisk60.B1' 274432,
         device '/dev/rlv.hdisk69.B1' 274432,
         device '/dev/rlv.hdisk58.B1' 274432,
         device '/dev/rlv.hdisk71.B1' 274432,
         device '/dev/rlv.hdisk62.B1' 274432,
         device '/dev/rlv.hdisk63.B1' 274432) on NODE (2)
  USING (
         device '/dev/rlv.hdisk123.B1' 274432,
         device '/dev/rlv.hdisk137.B1' 274432,
         device '/dev/rlv.hdisk134.B1' 274432,
         device '/dev/rlv.hdisk125.B1' 274432,
         device '/dev/rlv.hdisk129.B1' 274432,
         device '/dev/rlv.hdisk131.B1' 274432,
         device '/dev/rlv.hdisk122.B1' 274432,
         device '/dev/rlv.hdisk126.B1' 274432,
         device '/dev/rlv.hdisk132.B1' 274432,
         device '/dev/rlv.hdisk136.B1' 274432,
         device '/dev/rlv.hdisk124.B1' 274432,
         device '/dev/rlv.hdisk128.B1' 274432) on NODE (3)
  USING (
         device '/dev/rlv.hdisk75.B1' 274432,
         device '/dev/rlv.hdisk81.B1' 274432,
         device '/dev/rlv.hdisk89.B1' 274432,
         device '/dev/rlv.hdisk82.B1' 274432,
         device '/dev/rlv.hdisk88.B1' 274432,
         device '/dev/rlv.hdisk76.B1' 274432,
         device '/dev/rlv.hdisk83.B1' 274432,
         device '/dev/rlv.hdisk78.B1' 274432,
         device '/dev/rlv.hdisk74.B1' 274432,
         device '/dev/rlv.hdisk85.B1' 274432,
         device '/dev/rlv.hdisk84.B1' 274432,
         device '/dev/rlv.hdisk86.B1' 274432) on NODE (4)
  USING (
         device '/dev/rlv.hdisk146.B1' 274432,
         device '/dev/rlv.hdisk139.B1' 274432,
         device '/dev/rlv.hdisk152.B1' 274432,
         device '/dev/rlv.hdisk142.B1' 274432,
         device '/dev/rlv.hdisk151.B1' 274432,
         device '/dev/rlv.hdisk153.B1' 274432,
         device '/dev/rlv.hdisk140.B1' 274432,
         device '/dev/rlv.hdisk143.B1' 274432,
         device '/dev/rlv.hdisk149.B1' 274432,
         device '/dev/rlv.hdisk150.B1' 274432,
         device '/dev/rlv.hdisk144.B1' 274432,
         device '/dev/rlv.hdisk141.B1' 274432) on NODE (5)
  USING (
         device '/dev/rlv.hdisk175.B1' 274432,
         device '/dev/rlv.hdisk178.B1' 274432,
         device '/dev/rlv.hdisk174.B1' 274432,
         device '/dev/rlv.hdisk180.B1' 274432,
         device '/dev/rlv.hdisk183.B1' 274432,
         device '/dev/rlv.hdisk185.B1' 274432,
         device '/dev/rlv.hdisk172.B1' 274432,
         device '/dev/rlv.hdisk177.B1' 274432,
         device '/dev/rlv.hdisk176.B1' 274432,
         device '/dev/rlv.hdisk179.B1' 274432,
         device '/dev/rlv.hdisk173.B1' 274432,
         device '/dev/rlv.hdisk170.B1' 274432) on NODE (6)
  USING (
         device '/dev/rlv.hdisk42.B1' 274432,
         device '/dev/rlv.hdisk51.B1' 274432,
         device '/dev/rlv.hdisk53.B1' 274432,
         device '/dev/rlv.hdisk52.B1' 274432,
         device '/dev/rlv.hdisk43.B1' 274432,
         device '/dev/rlv.hdisk49.B1' 274432,
         device '/dev/rlv.hdisk47.B1' 274432,
         device '/dev/rlv.hdisk56.B1' 274432,
         device '/dev/rlv.hdisk55.B1' 274432,
         device '/dev/rlv.hdisk50.B1' 274432,
         device '/dev/rlv.hdisk48.B1' 274432,
```

```
        device '/dev/rlv.hdisk46.B1' 274432) on NODE (7)
  USING (
        device '/dev/rlv.hdisk116.B1' 274432,
        device '/dev/rlv.hdisk112.B1' 274432,
        device '/dev/rlv.hdisk113.B1' 274432,
        device '/dev/rlv.hdisk118.B1' 274432,
        device '/dev/rlv.hdisk117.B1' 274432,
        device '/dev/rlv.hdisk119.B1' 274432,
        device '/dev/rlv.hdisk115.B1' 274432,
        device '/dev/rlv.hdisk114.B1' 274432,
        device '/dev/rlv.hdisk121.B1' 274432,
        device '/dev/rlv.hdisk111.B1' 274432,
        device '/dev/rlv.hdisk109.B1' 274432,
        device '/dev/rlv.hdisk107.B1' 274432) on NODE (8)
  USING (
        device '/dev/rlv.hdisk161.B1' 274432,
        device '/dev/rlv.hdisk154.B1' 274432,
        device '/dev/rlv.hdisk166.B1' 274432,
        device '/dev/rlv.hdisk162.B1' 274432,
        device '/dev/rlv.hdisk163.B1' 274432,
        device '/dev/rlv.hdisk168.B1' 274432,
        device '/dev/rlv.hdisk169.B1' 274432,
        device '/dev/rlv.hdisk160.B1' 274432,
        device '/dev/rlv.hdisk159.B1' 274432,
        device '/dev/rlv.hdisk158.B1' 274432,
        device '/dev/rlv.hdisk155.B1' 274432,
        device '/dev/rlv.hdisk164.B1' 274432) on NODE (9)
  USING (
        device '/dev/rlv.hdisk190.B1' 274432,
        device '/dev/rlv.hdisk189.B1' 274432,
        device '/dev/rlv.hdisk186.B1' 274432,
        device '/dev/rlv.hdisk188.B1' 274432,
        device '/dev/rlv.hdisk187.B1' 274432,
        device '/dev/rlv.hdisk197.B1' 274432,
        device '/dev/rlv.hdisk195.B1' 274432,
        device '/dev/rlv.hdisk194.B1' 274432,
        device '/dev/rlv.hdisk192.B1' 274432,
        device '/dev/rlv.hdisk196.B1' 274432,
        device '/dev/rlv.hdisk200.B1' 274432,
        device '/dev/rlv.hdisk198.B1' 274432) on NODE (10)
  USING (
        device '/dev/rlv.hdisk102.B1' 274432,
        device '/dev/rlv.hdisk90.B1' 274432,
        device '/dev/rlv.hdisk96.B1' 274432,
        device '/dev/rlv.hdisk104.B1' 274432,
        device '/dev/rlv.hdisk95.B1' 274432,
        device '/dev/rlv.hdisk105.B1' 274432,
        device '/dev/rlv.hdisk94.B1' 274432,
        device '/dev/rlv.hdisk101.B1' 274432,
        device '/dev/rlv.hdisk100.B1' 274432,
        device '/dev/rlv.hdisk103.B1' 274432,
        device '/dev/rlv.hdisk93.B1' 274432,
        device '/dev/rlv.hdisk92.B1' 274432) on NODE (11)
       EXTENTSIZE 32 PREFETCHSIZE 384;

COMMIT WORK;

-- create the dms managed tablespace for indices
-- 144 lvs of 1072MB each on 144 disks
-- each lv has 67 PPs of size 16MB ==> 274432 pages
-- Total pages in data tablespace: 39518208 ==> ~154 GB
CREATE TABLESPACE TPCDINDEX MANAGED BY DATABASE
  USING (
        device '/dev/rlv.hdisk13.C1' 274432,
        device '/dev/rlv.hdisk25.C1' 274432,
        device '/dev/rlv.hdisk12.C1' 274432,
        device '/dev/rlv.hdisk20.C1' 274432,
        device '/dev/rlv.hdisk11.C1' 274432,
        device '/dev/rlv.hdisk21.C1' 274432,
        device '/dev/rlv.hdisk23.C1' 274432,
        device '/dev/rlv.hdisk19.C1' 274432,
        device '/dev/rlv.hdisk17.C1' 274432,
        device '/dev/rlv.hdisk10.C1' 274432,
        device '/dev/rlv.hdisk15.C1' 274432,
        device '/dev/rlv.hdisk16.C1' 274432) on NODE (0)
  USING (
        device '/dev/rlv.hdisk31.C1' 274432,
        device '/dev/rlv.hdisk37.C1' 274432,
        device '/dev/rlv.hdisk26.C1' 274432,
        device '/dev/rlv.hdisk40.C1' 274432,
        device '/dev/rlv.hdisk41.C1' 274432,
        device '/dev/rlv.hdisk39.C1' 274432,
        device '/dev/rlv.hdisk27.C1' 274432,
        device '/dev/rlv.hdisk33.C1' 274432,
        device '/dev/rlv.hdisk29.C1' 274432,
        device '/dev/rlv.hdisk36.C1' 274432,
        device '/dev/rlv.hdisk34.C1' 274432,
        device '/dev/rlv.hdisk28.C1' 274432) on NODE (1)
  USING (
        device '/dev/rlv.hdisk68.C1' 274432,
        device '/dev/rlv.hdisk64.C1' 274432,
        device '/dev/rlv.hdisk65.C1' 274432,
        device '/dev/rlv.hdisk73.C1' 274432,
        device '/dev/rlv.hdisk59.C1' 274432,
        device '/dev/rlv.hdisk72.C1' 274432,
        device '/dev/rlv.hdisk60.C1' 274432,
        device '/dev/rlv.hdisk69.C1' 274432,
        device '/dev/rlv.hdisk58.C1' 274432,
        device '/dev/rlv.hdisk71.C1' 274432,
        device '/dev/rlv.hdisk62.C1' 274432,
        device '/dev/rlv.hdisk63.C1' 274432) on NODE (2)
  USING (
        device '/dev/rlv.hdisk123.C1' 274432,
        device '/dev/rlv.hdisk137.C1' 274432,
        device '/dev/rlv.hdisk134.C1' 274432,
        device '/dev/rlv.hdisk125.C1' 274432,
        device '/dev/rlv.hdisk129.C1' 274432,
        device '/dev/rlv.hdisk131.C1' 274432,
        device '/dev/rlv.hdisk122.C1' 274432,
        device '/dev/rlv.hdisk126.C1' 274432,
        device '/dev/rlv.hdisk132.C1' 274432,
        device '/dev/rlv.hdisk136.C1' 274432,
        device '/dev/rlv.hdisk124.C1' 274432,
        device '/dev/rlv.hdisk128.C1' 274432) on NODE (3)
  USING (
        device '/dev/rlv.hdisk75.C1' 274432,
        device '/dev/rlv.hdisk81.C1' 274432,
        device '/dev/rlv.hdisk89.C1' 274432,
        device '/dev/rlv.hdisk82.C1' 274432,
        device '/dev/rlv.hdisk88.C1' 274432,
        device '/dev/rlv.hdisk76.C1' 274432,
        device '/dev/rlv.hdisk83.C1' 274432,
        device '/dev/rlv.hdisk78.C1' 274432,
        device '/dev/rlv.hdisk74.C1' 274432,
        device '/dev/rlv.hdisk85.C1' 274432,
        device '/dev/rlv.hdisk84.C1' 274432,
        device '/dev/rlv.hdisk86.C1' 274432) on NODE (4)
  USING (
        device '/dev/rlv.hdisk146.C1' 274432,
        device '/dev/rlv.hdisk139.C1' 274432,
        device '/dev/rlv.hdisk152.C1' 274432,
        device '/dev/rlv.hdisk142.C1' 274432,
        device '/dev/rlv.hdisk151.C1' 274432,
        device '/dev/rlv.hdisk153.C1' 274432,
        device '/dev/rlv.hdisk140.C1' 274432,
        device '/dev/rlv.hdisk143.C1' 274432,
        device '/dev/rlv.hdisk149.C1' 274432,
        device '/dev/rlv.hdisk150.C1' 274432,
        device '/dev/rlv.hdisk144.C1' 274432,
        device '/dev/rlv.hdisk141.C1' 274432) on NODE (5)
  USING (
        device '/dev/rlv.hdisk175.C1' 274432,
        device '/dev/rlv.hdisk178.C1' 274432,
        device '/dev/rlv.hdisk174.C1' 274432,
        device '/dev/rlv.hdisk180.C1' 274432,
        device '/dev/rlv.hdisk183.C1' 274432,
        device '/dev/rlv.hdisk185.C1' 274432,
        device '/dev/rlv.hdisk172.C1' 274432,
        device '/dev/rlv.hdisk177.C1' 274432,
        device '/dev/rlv.hdisk176.C1' 274432,
        device '/dev/rlv.hdisk179.C1' 274432,
        device '/dev/rlv.hdisk173.C1' 274432,
        device '/dev/rlv.hdisk170.C1' 274432) on NODE (6)
  USING (
        device '/dev/rlv.hdisk42.C1' 274432,
        device '/dev/rlv.hdisk51.C1' 274432,
        device '/dev/rlv.hdisk53.C1' 274432,
        device '/dev/rlv.hdisk52.C1' 274432,
        device '/dev/rlv.hdisk43.C1' 274432,
        device '/dev/rlv.hdisk49.C1' 274432,
        device '/dev/rlv.hdisk47.C1' 274432,
        device '/dev/rlv.hdisk56.C1' 274432,
        device '/dev/rlv.hdisk55.C1' 274432,
        device '/dev/rlv.hdisk50.C1' 274432,
        device '/dev/rlv.hdisk48.C1' 274432,
        device '/dev/rlv.hdisk46.C1' 274432) on NODE (7)
  USING (
        device '/dev/rlv.hdisk116.C1' 274432,
        device '/dev/rlv.hdisk112.C1' 274432,
        device '/dev/rlv.hdisk113.C1' 274432,
        device '/dev/rlv.hdisk118.C1' 274432,
        device '/dev/rlv.hdisk117.C1' 274432,
        device '/dev/rlv.hdisk119.C1' 274432,
        device '/dev/rlv.hdisk115.C1' 274432,
        device '/dev/rlv.hdisk114.C1' 274432,
        device '/dev/rlv.hdisk121.C1' 274432,
        device '/dev/rlv.hdisk111.C1' 274432,
        device '/dev/rlv.hdisk109.C1' 274432,
        device '/dev/rlv.hdisk107.C1' 274432) on NODE (8)
  USING (
        device '/dev/rlv.hdisk161.C1' 274432,
        device '/dev/rlv.hdisk154.C1' 274432,
        device '/dev/rlv.hdisk166.C1' 274432,
        device '/dev/rlv.hdisk162.C1' 274432,
        device '/dev/rlv.hdisk163.C1' 274432,
        device '/dev/rlv.hdisk168.C1' 274432,
        device '/dev/rlv.hdisk169.C1' 274432,
        device '/dev/rlv.hdisk160.C1' 274432,
        device '/dev/rlv.hdisk159.C1' 274432,
        device '/dev/rlv.hdisk158.C1' 274432,
        device '/dev/rlv.hdisk155.C1' 274432,
        device '/dev/rlv.hdisk164.C1' 274432) on NODE (9)
  USING (
        device '/dev/rlv.hdisk190.C1' 274432,
        device '/dev/rlv.hdisk189.C1' 274432,
        device '/dev/rlv.hdisk186.C1' 274432,
        device '/dev/rlv.hdisk188.C1' 274432,
        device '/dev/rlv.hdisk187.C1' 274432,
        device '/dev/rlv.hdisk197.C1' 274432,
        device '/dev/rlv.hdisk195.C1' 274432,
        device '/dev/rlv.hdisk194.C1' 274432,
        device '/dev/rlv.hdisk192.C1' 274432,
        device '/dev/rlv.hdisk196.C1' 274432,
        device '/dev/rlv.hdisk200.C1' 274432,
        device '/dev/rlv.hdisk198.C1' 274432) on NODE (10)
  USING (
        device '/dev/rlv.hdisk102.C1' 274432,
        device '/dev/rlv.hdisk90.C1' 274432,
        device '/dev/rlv.hdisk96.C1' 274432,
        device '/dev/rlv.hdisk104.C1' 274432,
        device '/dev/rlv.hdisk95.C1' 274432,
        device '/dev/rlv.hdisk105.C1' 274432,
        device '/dev/rlv.hdisk94.C1' 274432,
        device '/dev/rlv.hdisk101.C1' 274432,
        device '/dev/rlv.hdisk100.C1' 274432,
        device '/dev/rlv.hdisk103.C1' 274432,
        device '/dev/rlv.hdisk93.C1' 274432,
        device '/dev/rlv.hdisk92.C1' 274432) on NODE (11)
       EXTENTSIZE 32 PREFETCHSIZE 384;

COMMIT WORK;

create nodegroup NODE0 on NODE (0);

commit work;
create regular tablespace tsnode_1 in nodegroup node0 managed by system
using ('/home/tpcd/tpcd100/tsnode_1') on node (0);

commit work;

alter bufferpool ibmdefaultbp size -1;
commit work;

create tablespace UFTEMP1 managed by system
```

```
     using ('/UF100GB/uf1_0') on node (0)
     using ('/UF100GB/uf1_1') on node (1)
     using ('/UF100GB/uf1_2') on node (2)
     using ('/UF100GB/uf1_3') on node (3)
     using ('/UF100GB/uf1_4') on node (4)
     using ('/UF100GB/uf1_5') on node (5)
     using ('/UF100GB/uf1_6') on node (6)
     using ('/UF100GB/uf1_7') on node (7)
     using ('/UF100GB/uf1_8') on node (8)
     using ('/UF100GB/uf1_9') on node (9)
     using ('/UF100GB/uf1_10') on node (10)
     using ('/UF100GB/uf1_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP2 managed by system
     using ('/UF100GB/uf2_0') on node (0)
     using ('/UF100GB/uf2_1') on node (1)
     using ('/UF100GB/uf2_2') on node (2)
     using ('/UF100GB/uf2_3') on node (3)
     using ('/UF100GB/uf2_4') on node (4)
     using ('/UF100GB/uf2_5') on node (5)
     using ('/UF100GB/uf2_6') on node (6)
     using ('/UF100GB/uf2_7') on node (7)
     using ('/UF100GB/uf2_8') on node (8)
     using ('/UF100GB/uf2_9') on node (9)
     using ('/UF100GB/uf2_10') on node (10)
     using ('/UF100GB/uf2_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP3 managed by system
     using ('/UF100GB/uf3_0') on node (0)
     using ('/UF100GB/uf3_1') on node (1)
     using ('/UF100GB/uf3_2') on node (2)
     using ('/UF100GB/uf3_3') on node (3)
     using ('/UF100GB/uf3_4') on node (4)
     using ('/UF100GB/uf3_5') on node (5)
     using ('/UF100GB/uf3_6') on node (6)
     using ('/UF100GB/uf3_7') on node (7)
     using ('/UF100GB/uf3_8') on node (8)
     using ('/UF100GB/uf3_9') on node (9)
     using ('/UF100GB/uf3_10') on node (10)
     using ('/UF100GB/uf3_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP4 managed by system
     using ('/UF100GB/uf4_0') on node (0)
     using ('/UF100GB/uf4_1') on node (1)
     using ('/UF100GB/uf4_2') on node (2)
     using ('/UF100GB/uf4_3') on node (3)
     using ('/UF100GB/uf4_4') on node (4)
     using ('/UF100GB/uf4_5') on node (5)
     using ('/UF100GB/uf4_6') on node (6)
     using ('/UF100GB/uf4_7') on node (7)
     using ('/UF100GB/uf4_8') on node (8)
     using ('/UF100GB/uf4_9') on node (9)
     using ('/UF100GB/uf4_10') on node (10)
     using ('/UF100GB/uf4_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;

create tablespace UFTEMP5 managed by system
     using ('/UF100GB/uf5_0') on node (0)
     using ('/UF100GB/uf5_1') on node (1)
     using ('/UF100GB/uf5_2') on node (2)
     using ('/UF100GB/uf5_3') on node (3)
     using ('/UF100GB/uf5_4') on node (4)
     using ('/UF100GB/uf5_5') on node (5)
     using ('/UF100GB/uf5_6') on node (6)
     using ('/UF100GB/uf5_7') on node (7)
     using ('/UF100GB/uf5_8') on node (8)
     using ('/UF100GB/uf5_9') on node (9)
     using ('/UF100GB/uf5_10') on node (10)
     using ('/UF100GB/uf5_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP6 managed by system
     using ('/UF100GB/uf6_0') on node (0)
     using ('/UF100GB/uf6_1') on node (1)
     using ('/UF100GB/uf6_2') on node (2)
     using ('/UF100GB/uf6_3') on node (3)
     using ('/UF100GB/uf6_4') on node (4)
     using ('/UF100GB/uf6_5') on node (5)
     using ('/UF100GB/uf6_6') on node (6)
     using ('/UF100GB/uf6_7') on node (7)
     using ('/UF100GB/uf6_8') on node (8)
     using ('/UF100GB/uf6_9') on node (9)
     using ('/UF100GB/uf6_10') on node (10)
     using ('/UF100GB/uf6_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP7 managed by system
     using ('/UF100GB/uf7_0') on node (0)
     using ('/UF100GB/uf7_1') on node (1)
     using ('/UF100GB/uf7_2') on node (2)
     using ('/UF100GB/uf7_3') on node (3)
     using ('/UF100GB/uf7_4') on node (4)
     using ('/UF100GB/uf7_5') on node (5)
     using ('/UF100GB/uf7_6') on node (6)
     using ('/UF100GB/uf7_7') on node (7)
     using ('/UF100GB/uf7_8') on node (8)
     using ('/UF100GB/uf7_9') on node (9)
     using ('/UF100GB/uf7_10') on node (10)
     using ('/UF100GB/uf7_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP8 managed by system
     using ('/UF100GB/uf8_0') on node (0)
     using ('/UF100GB/uf8_1') on node (1)
     using ('/UF100GB/uf8_2') on node (2)
     using ('/UF100GB/uf8_3') on node (3)
     using ('/UF100GB/uf8_4') on node (4)
     using ('/UF100GB/uf8_5') on node (5)
     using ('/UF100GB/uf8_6') on node (6)
     using ('/UF100GB/uf8_7') on node (7)
     using ('/UF100GB/uf8_8') on node (8)
     using ('/UF100GB/uf8_9') on node (9)
     using ('/UF100GB/uf8_10') on node (10)
     using ('/UF100GB/uf8_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;

create tablespace UFTEMP9 managed by system
     using ('/UF100GB/uf9_0') on node (0)
     using ('/UF100GB/uf9_1') on node (1)
     using ('/UF100GB/uf9_2') on node (2)
     using ('/UF100GB/uf9_3') on node (3)
     using ('/UF100GB/uf9_4') on node (4)
     using ('/UF100GB/uf9_5') on node (5)
     using ('/UF100GB/uf9_6') on node (6)
     using ('/UF100GB/uf9_7') on node (7)
     using ('/UF100GB/uf9_8') on node (8)
     using ('/UF100GB/uf9_9') on node (9)
     using ('/UF100GB/uf9_10') on node (10)
     using ('/UF100GB/uf9_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP10 managed by system
     using ('/UF100GB/uf10_0') on node (0)
     using ('/UF100GB/uf10_1') on node (1)
     using ('/UF100GB/uf10_2') on node (2)
     using ('/UF100GB/uf10_3') on node (3)
     using ('/UF100GB/uf10_4') on node (4)
     using ('/UF100GB/uf10_5') on node (5)
     using ('/UF100GB/uf10_6') on node (6)
     using ('/UF100GB/uf10_7') on node (7)
     using ('/UF100GB/uf10_8') on node (8)
     using ('/UF100GB/uf10_9') on node (9)
     using ('/UF100GB/uf10_10') on node (10)
     using ('/UF100GB/uf10_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP11 managed by system
     using ('/UF100GB/uf11_0') on node (0)
     using ('/UF100GB/uf11_1') on node (1)
     using ('/UF100GB/uf11_2') on node (2)
     using ('/UF100GB/uf11_3') on node (3)
     using ('/UF100GB/uf11_4') on node (4)
     using ('/UF100GB/uf11_5') on node (5)
     using ('/UF100GB/uf11_6') on node (6)
     using ('/UF100GB/uf11_7') on node (7)
     using ('/UF100GB/uf11_8') on node (8)
     using ('/UF100GB/uf11_9') on node (9)
     using ('/UF100GB/uf11_10') on node (10)
     using ('/UF100GB/uf11_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP12 managed by system
     using ('/UF100GB/uf12_0') on node (0)
     using ('/UF100GB/uf12_1') on node (1)
     using ('/UF100GB/uf12_2') on node (2)
     using ('/UF100GB/uf12_3') on node (3)
     using ('/UF100GB/uf12_4') on node (4)
     using ('/UF100GB/uf12_5') on node (5)
     using ('/UF100GB/uf12_6') on node (6)
     using ('/UF100GB/uf12_7') on node (7)
     using ('/UF100GB/uf12_8') on node (8)
     using ('/UF100GB/uf12_9') on node (9)
     using ('/UF100GB/uf12_10') on node (10)
     using ('/UF100GB/uf12_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP13 managed by system
     using ('/UF100GB/uf13_0') on node (0)
     using ('/UF100GB/uf13_1') on node (1)
     using ('/UF100GB/uf13_2') on node (2)
     using ('/UF100GB/uf13_3') on node (3)
     using ('/UF100GB/uf13_4') on node (4)
     using ('/UF100GB/uf13_5') on node (5)
     using ('/UF100GB/uf13_6') on node (6)
     using ('/UF100GB/uf13_7') on node (7)
     using ('/UF100GB/uf13_8') on node (8)
     using ('/UF100GB/uf13_9') on node (9)
     using ('/UF100GB/uf13_10') on node (10)
     using ('/UF100GB/uf13_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP14 managed by system
     using ('/UF100GB/uf14_0') on node (0)
     using ('/UF100GB/uf14_1') on node (1)
     using ('/UF100GB/uf14_2') on node (2)
     using ('/UF100GB/uf14_3') on node (3)
     using ('/UF100GB/uf14_4') on node (4)
     using ('/UF100GB/uf14_5') on node (5)
     using ('/UF100GB/uf14_6') on node (6)
     using ('/UF100GB/uf14_7') on node (7)
     using ('/UF100GB/uf14_8') on node (8)
     using ('/UF100GB/uf14_9') on node (9)
     using ('/UF100GB/uf14_10') on node (10)
     using ('/UF100GB/uf14_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP15 managed by system
     using ('/UF100GB/uf15_0') on node (0)
     using ('/UF100GB/uf15_1') on node (1)
     using ('/UF100GB/uf15_2') on node (2)
     using ('/UF100GB/uf15_3') on node (3)
     using ('/UF100GB/uf15_4') on node (4)
     using ('/UF100GB/uf15_5') on node (5)
     using ('/UF100GB/uf15_6') on node (6)
     using ('/UF100GB/uf15_7') on node (7)
     using ('/UF100GB/uf15_8') on node (8)
     using ('/UF100GB/uf15_9') on node (9)
     using ('/UF100GB/uf15_10') on node (10)
     using ('/UF100GB/uf15_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP16 managed by system
     using ('/UF100GB/uf16_0') on node (0)
     using ('/UF100GB/uf16_1') on node (1)
     using ('/UF100GB/uf16_2') on node (2)
     using ('/UF100GB/uf16_3') on node (3)
     using ('/UF100GB/uf16_4') on node (4)
     using ('/UF100GB/uf16_5') on node (5)
     using ('/UF100GB/uf16_6') on node (6)
```

```
     using ('/UF100GB/uf16_7') on node (7)
     using ('/UF100GB/uf16_8') on node (8)
     using ('/UF100GB/uf16_9') on node (9)
     using ('/UF100GB/uf16_10') on node (10)
     using ('/UF100GB/uf16_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP17 managed by system
     using ('/UF100GB/uf17_0') on node (0)
     using ('/UF100GB/uf17_1') on node (1)
     using ('/UF100GB/uf17_2') on node (2)
     using ('/UF100GB/uf17_3') on node (3)
     using ('/UF100GB/uf17_4') on node (4)
     using ('/UF100GB/uf17_5') on node (5)
     using ('/UF100GB/uf17_6') on node (6)
     using ('/UF100GB/uf17_7') on node (7)
     using ('/UF100GB/uf17_8') on node (8)
     using ('/UF100GB/uf17_9') on node (9)
     using ('/UF100GB/uf17_10') on node (10)
     using ('/UF100GB/uf17_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP18 managed by system
     using ('/UF100GB/uf18_0') on node (0)
     using ('/UF100GB/uf18_1') on node (1)
     using ('/UF100GB/uf18_2') on node (2)
     using ('/UF100GB/uf18_3') on node (3)
     using ('/UF100GB/uf18_4') on node (4)
     using ('/UF100GB/uf18_5') on node (5)
     using ('/UF100GB/uf18_6') on node (6)
     using ('/UF100GB/uf18_7') on node (7)
     using ('/UF100GB/uf18_8') on node (8)
     using ('/UF100GB/uf18_9') on node (9)
     using ('/UF100GB/uf18_10') on node (10)
     using ('/UF100GB/uf18_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP19 managed by system
     using ('/UF100GB/uf19_0') on node (0)
     using ('/UF100GB/uf19_1') on node (1)
     using ('/UF100GB/uf19_2') on node (2)
     using ('/UF100GB/uf19_3') on node (3)
     using ('/UF100GB/uf19_4') on node (4)
     using ('/UF100GB/uf19_5') on node (5)
     using ('/UF100GB/uf19_6') on node (6)
     using ('/UF100GB/uf19_7') on node (7)
     using ('/UF100GB/uf19_8') on node (8)
     using ('/UF100GB/uf19_9') on node (9)
     using ('/UF100GB/uf19_10') on node (10)
     using ('/UF100GB/uf19_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP20 managed by system
     using ('/UF100GB/uf20_0') on node (0)
     using ('/UF100GB/uf20_1') on node (1)
     using ('/UF100GB/uf20_2') on node (2)
     using ('/UF100GB/uf20_3') on node (3)
     using ('/UF100GB/uf20_4') on node (4)
     using ('/UF100GB/uf20_5') on node (5)
     using ('/UF100GB/uf20_6') on node (6)
     using ('/UF100GB/uf20_7') on node (7)
     using ('/UF100GB/uf20_8') on node (8)
     using ('/UF100GB/uf20_9') on node (9)
     using ('/UF100GB/uf20_10') on node (10)
     using ('/UF100GB/uf20_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;

create tablespace UFTEMP21 managed by system
     using ('/UF100GB/uf21_0') on node (0)
     using ('/UF100GB/uf21_1') on node (1)
     using ('/UF100GB/uf21_2') on node (2)
     using ('/UF100GB/uf21_3') on node (3)
     using ('/UF100GB/uf21_4') on node (4)
     using ('/UF100GB/uf21_5') on node (5)
     using ('/UF100GB/uf21_6') on node (6)
     using ('/UF100GB/uf21_7') on node (7)
     using ('/UF100GB/uf21_8') on node (8)
     using ('/UF100GB/uf21_9') on node (9)
     using ('/UF100GB/uf21_10') on node (10)
     using ('/UF100GB/uf21_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP22 managed by system
     using ('/UF100GB/uf22_0') on node (0)
     using ('/UF100GB/uf22_1') on node (1)
     using ('/UF100GB/uf22_2') on node (2)
     using ('/UF100GB/uf22_3') on node (3)
     using ('/UF100GB/uf22_4') on node (4)
     using ('/UF100GB/uf22_5') on node (5)
     using ('/UF100GB/uf22_6') on node (6)
     using ('/UF100GB/uf22_7') on node (7)
     using ('/UF100GB/uf22_8') on node (8)
     using ('/UF100GB/uf22_9') on node (9)
     using ('/UF100GB/uf22_10') on node (10)
     using ('/UF100GB/uf22_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP23 managed by system
     using ('/UF100GB/uf23_0') on node (0)
     using ('/UF100GB/uf23_1') on node (1)
     using ('/UF100GB/uf23_2') on node (2)
     using ('/UF100GB/uf23_3') on node (3)
     using ('/UF100GB/uf23_4') on node (4)
     using ('/UF100GB/uf23_5') on node (5)
     using ('/UF100GB/uf23_6') on node (6)
     using ('/UF100GB/uf23_7') on node (7)
     using ('/UF100GB/uf23_8') on node (8)
     using ('/UF100GB/uf23_9') on node (9)
     using ('/UF100GB/uf23_10') on node (10)
     using ('/UF100GB/uf23_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP24 managed by system
     using ('/UF100GB/uf24_0') on node (0)
     using ('/UF100GB/uf24_1') on node (1)
     using ('/UF100GB/uf24_2') on node (2)
     using ('/UF100GB/uf24_3') on node (3)
     using ('/UF100GB/uf24_4') on node (4)
     using ('/UF100GB/uf24_5') on node (5)
     using ('/UF100GB/uf24_6') on node (6)
     using ('/UF100GB/uf24_7') on node (7)
     using ('/UF100GB/uf24_8') on node (8)
     using ('/UF100GB/uf24_9') on node (9)
     using ('/UF100GB/uf24_10') on node (10)
     using ('/UF100GB/uf24_11') on node (11)
    extentsize 32 prefetchsize 32;
commit work;

--CONNECT RESET;
--TERMINATE;
```

B.14 dss.ddl100GB.tbl.acme

```
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 TSNODE_1;

CREATE TABLE TPCD.REGION  ( R_REGIONKEY  INTEGER NOT NULL,
                            R_NAME       CHAR(25) NOT NULL,
                            R_COMMENT    VARCHAR(152) NOT NULL WITH
DEFAULT )
                            IN TSNODE_1;

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 TPCDDATA INDEX IN TPCDINDEX
                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 TPCDDATA INDEX IN TPCDINDEX
                 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 TPCDDATA INDEX IN TPCDINDEX
                PARTITIONING KEY(PS_PARTKEY) USING HASHING ;

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

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 TPCDDATA INDEX IN TPCDINDEX
                PARTITIONING KEY(L_ORDERKEY) USING HASHING;

COMMIT WORK;
```

B.15 dss.index.acme

```
CREATE UNIQUE INDEX "TPCD  "."C_MS_CK" ON "TPCD    "."CUSTOMER"
                ("C_MKTSEGMENT" ASC,
                 "C_CUSTKEY" ASC)
                PCTFREE 0 ;
commit work;
values(current timestamp);
CREATE UNIQUE INDEX "TPCD    "."C_NAT_CKEY_REG" ON "TPCD    "."CUSTOMER"
                ("C_NATIONKEY" ASC,
                 "C_CUSTKEY" ASC)
                PCTFREE 0 ;
commit work;

values(current timestamp);
CREATE UNIQUE INDEX TPCD.C_CK ON TPCD.CUSTOMER (C_CUSTKEY) PCTFREE 0;
commit;

values(current timestamp);
CREATE INDEX "TPCD    "."L_OKCDRD" ON "TPCD    "."LINEITEM"
                ("L_ORDERKEY" ASC,
                 "L_COMMITDATE" ASC,
                 "L_RECEIPTDATE" ASC)
                PCTFREE 4 ;
commit work;

values(current timestamp);
CREATE INDEX TPCD.L_SMRDCDSDOK ON TPCD.LINEITEM
                (L_SHIPMODE ASC,
                 L_RECEIPTDATE ASC,
                 L_COMMITDATE ASC,
                 L_SHIPDATE ASC,
                 L_ORDERKEY ASC)
               PCTFREE 5;
commit work;

values(current timestamp);
CREATE INDEX "TPCD    "."L_SDDSEPSKPK" ON "TPCD    "."LINEITEM"
                ("L_SHIPDATE" ASC,
                 "L_DISCOUNT" ASC,
                 "L_EXTENDEDPRICE" ASC,
                 "L_SUPPKEY" ASC,
                 "L_PARTKEY" ASC)
                PCTFREE 6 ;
commit work;

values(current timestamp);
CREATE INDEX "TPCD    "."PXL@OKSDRFSKEPDC" ON "TPCD    "."LINEITEM"
                ("L_ORDERKEY" ASC,
                 "L_SHIPDATE" ASC,
                 "L_RETURNFLAG" ASC,
                 "L_SUPPKEY" ASC,
                 "L_EXTENDEDPRICE" ASC,
                 "L_DISCOUNT" ASC)
                PCTFREE 6 ;
commit work;
values(current timestamp);
CREATE INDEX "TPCD    "."L_PKSKOKEPDSQN" ON "TPCD    "."LINEITEM"
                ("L_PARTKEY" ASC,
                 "L_SUPPKEY" ASC,
                 "L_ORDERKEY" ASC,
                 "L_EXTENDEDPRICE" ASC,
                 "L_DISCOUNT" ASC,
                 "L_QUANTITY" ASC)
                PCTFREE 6 ;
commit work;


values(current timestamp);
CREATE UNIQUE INDEX "TPCD    "."N_NAMNATKEY" ON "TPCD    "."NATION"
                ("N_NAME" ASC,
                 "N_NATIONKEY" ASC)
                PCTFREE 0 ;
commit work;

values(current timestamp);
CREATE UNIQUE INDEX "TPCD    "."N_NATKEYNAM" ON "TPCD    "."NATION"
                ("N_NATIONKEY" ASC) include(
                 "N_NAME" ASC)
                PCTFREE 0 ;
commit work;

values(current timestamp);
CREATE UNIQUE INDEX "TPCD    "."N_REGKEYNATKEYNAM" ON "TPCD
"."NATION"
                ("N_REGIONKEY" ASC,
                 "N_NATIONKEY" ASC) include (
                 "N_NAME" ASC)
                PCTFREE 0 ;
commit work;

values(current timestamp);
CREATE INDEX "TPCD  "."O_CK_OD_OK_SP" ON "TPCD    "."ORDERS"
                ("O_CUSTKEY" ASC,
                 "O_ORDERDATE" ASC,
                 "O_ORDERKEY" ASC,
                 "O_SHIPPRIORITY" ASC)
                PCTFREE 4 ;
commit work;

values(current timestamp);
CREATE UNIQUE INDEX "TPCD  "."O_OK_OD_OP" ON "TPCD    "."ORDERS"
                ("O_ORDERKEY" ASC) include(
                 "O_ORDERDATE" ASC,
                 "O_ORDERPRIORITY" ASC)
                PCTFREE 4 ;
commit work;

values(current timestamp);
CREATE INDEX "TPCD    "."O_CLERK" ON "TPCD    "."ORDERS"
                ("O_CLERK" ASC)
                PCTFREE 4 ;
commit work;
values(current timestamp);
CREATE  INDEX "TPCD    "."O_OD_OK_OP_CK" ON "TPCD    "."ORDERS"
                ("O_ORDERDATE" ASC,
                 "O_ORDERKEY" ASC,
                 "O_ORDERPRIORITY" ASC,
                 "O_CUSTKEY" ASC)
                PCTFREE 4 ;
commit work;

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

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

values(current timestamp);
CREATE UNIQUE INDEX "TPCD    "."P_SIZE_PK_BR_TY" ON "TPCD    "."PART"
                ("P_SIZE" ASC,
                 "P_PARTKEY" ASC ) include (
                 "P_BRAND" ASC,
                 "P_TYPE" ASC )
                PCTFREE 0 ;
commit work;

values(current timestamp);
CREATE UNIQUE INDEX "TPCD    "."SXP_BRC2PK" ON "TPCD    "."PART"
                ("P_BRAND" ASC,
                 "P_CONTAINER" ASC,
                 "P_PARTKEY" ASC)
                PCTFREE 0 ;
commit work;

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

values(current timestamp);
CREATE UNIQUE INDEX "TPCD    "."UXP_SZTYPKMF" ON "TPCD    "."PART"
                ("P_SIZE" ASC,
                 "P_TYPE" ASC,
                 "P_PARTKEY" ASC) include (
                 "P_MFGR" ASC)
                PCTFREE 0 ;
commit work;

values(current timestamp);
CREATE UNIQUE INDEX "TPCD    "."UXPS_PK2KSC" ON "TPCD    "."PARTSUPP"
                ("PS_PARTKEY" ASC,
                 "PS_SUPPKEY" ASC) include(
                 "PS_SUPPLYCOST" ASC)
                PCTFREE 0 ;
commit work;

values(current timestamp);
CREATE UNIQUE INDEX "TPCD    "."UXPS_SK2PKSCAQ" ON "TPCD    "."PARTSUPP"
                ("PS_SUPPKEY" ASC,
                 "PS_PARTKEY" ASC) include(
                 "PS_SUPPLYCOST" ASC,
                 "PS_AVAILQTY" ASC)
                PCTFREE 0 ;
commit work;

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

values(current timestamp);
CREATE UNIQUE INDEX "TPCD    "."S_NAT_SKEY_REG" ON "TPCD    "."SUPPLIER"
                ("S_NATIONKEY" ASC,
                 "S_SUPPKEY" ASC)
                PCTFREE 0 ;
commit work;

values(current timestamp);
CREATE UNIQUE INDEX "TPCD    "."UXS_SKNK" ON "TPCD    "."SUPPLIER"
                ("S_SUPPKEY" ASC) include(
                 "S_NATIONKEY" ASC)
                PCTFREE 0 ;
commit work;

values(current timestamp);
```

B.16 dss.ri3

```
values(current timestamp);
alter table TPCD.NATION add constraint pk primary key (N_NATIONKEY);
commit work;

values(current timestamp);
alter table TPCD.CUSTOMER add constraint pk primary key (C_CUSTKEY);
commit work;
values(current timestamp);

lock table tpcd.nation in exclusive mode;
lock table tpcd.customer in exclusive mode;

select cast(raise_error('70001', 'constraint violation') as int) from
 (select n_nationkey from (customer left outer join nation on
c_nationkey =
n_nationkey)) x(c) where c is null;
```

```
set constraints for tpcd.customer off;
values(current timestamp);
alter table TPCD.CUSTOMER add constraint fk foreign key (C_NATIONKEY)
references TPCD.NATION;
set constraints for tpcd.customer all immediate unchecked;
commit work;

values(current timestamp);

lock table tpcd.customer in exclusive mode;
lock table tpcd.orders in exclusive mode;

select cast(raise_error('70001', 'constraint violation') as int) from
 (select c_custkey from (orders left outer join customer on o_custkey =
c_custkey)) x(c) where c is null;

set constraints for tpcd.orders off;
values(current timestamp);
alter table TPCD.ORDERS add constraint fk foreign key (O_CUSTKEY)
references TPCD.CUSTOMER;
values(current timestamp);
set constraints for tpcd.orders all immediate unchecked;
commit work;
values(current timestamp);
```

B.17 dss.AST

```
connect reset;

change isolation to RR;
connect to tpcd;

values ('l_summary timestamp: start               ', current timestamp);
create summary table tpcd.l_summary
as
(select
  l_shipdate, l_returnflag, l_linestatus, l_discount, l_quantity,
  sum(l_quantity)                                  as s1,
  sum(l_extendedprice)                             as s2,
  sum(l_extendedprice*(1-l_discount))              as s3,
  sum(l_extendedprice*(1-l_discount)*(1+l_tax))  as s4,
  sum(l_discount)                                  as s5,
  sum(l_extendedprice * l_discount)                as revenue,
  count_big(*)                                     as count

from tpcd.lineitem
group by l_shipdate, l_returnflag, l_linestatus, l_discount, l_quantity)
data initially deferred
refresh immediate
partitioning key (l_shipdate)
in tpcddata index in tpcdindex
not logged initially;

refresh table tpcd.l_summary;
commit work;

values ('l_summary timestamp: after create/refresh', current timestamp);

create index tpcd.idx_l_sum on tpcd.l_summary (l_shipdate, l_returnflag,

                                        l_linestatus, l_discount,
l_quantity)
            pctfree 0;
commit work;

values ('l_summary timestamp: after create index  ', current timestamp);

runstats on table tpcd.l_summary and indexes all;
commit;

values ('l_summary timestamp: after runstats      ', current timestamp);
connect reset;
terminate;
```

B.18 dss.runstats

```
--CONNECT TO TPCD;
!date;
RUNSTATS ON TABLE TPCD.NATION    AND DETAILED INDEXES ALL;
commit;
!date;
RUNSTATS ON TABLE TPCD.REGION    AND DETAILED INDEXES ALL;
commit;
!date;
RUNSTATS ON TABLE TPCD.SUPPLIER  AND DETAILED INDEXES ALL;
commit;
!date;
RUNSTATS ON TABLE TPCD.PART      AND DETAILED INDEXES ALL;
commit;
!date;
RUNSTATS ON TABLE TPCD.PARTSUPP  AND DETAILED INDEXES ALL;
commit;
!date;
RUNSTATS ON TABLE TPCD.CUSTOMER  AND DETAILED INDEXES ALL;
commit;
!date;
RUNSTATS ON TABLE TPCD.ORDERS    AND DETAILED INDEXES ALL;
commit;
!date;
RUNSTATS ON TABLE TPCD.LINEITEM  AND DETAILED INDEXES ALL;
COMMIT WORK;
!date;

--CONNECT RESET;
--TERMINATE;
```

B.19 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=aix                  # aix, nt, ....
TPCD_DBNAME=TPCD                   # name to create database under
TPCD_VERSION=1                     # version of TPCD
TPCD_AUDIT_DIR=/home/tpcd/tpcd     # top level directory of tar file for
                                   # all the tpcd scripts
TPCD_PRODUCT=v5                    # v5 or pe
TPCD_MODE=mln                      # uni/smp/mln/mpp
TPCD_PHYS_NODE=1                   # number of physical nodes
TPCD_LN_PER_PN=12                  # number of logical nodes per
                                   # physical node
TPCD_SF=100                        # size of the database (1=1GB,...) to
                                   # get test size databases use:
                                   #     0.012 = 12MB
                                   #     0.1   = 100MB
TPCD_BUILD_DATABASE=NULL           # whether to build a new database ???
                                   # NOT CURRENTLY USED
TPCD_BUILD_STAGE=NULL              # start just before crt tbsp
                                   # flat files.
TPCD_DBPATH=/home/tpcd/tpcd100     # path for database (defaults to home)
TPCD_DDLPATH=/home/tpcd/custom.acme
                                  # path for all ddl files and customized
                                  # scripts (load script), config files,etc
TPCD_TBSP_DDL=dss.ddl100GB.tbsp.acme.12w.mirror.addtmp
                                 # ddl file for tablespaces
TPCD_DDL=dss.ddl100GB.tbl.acme     # ddl file for tables
TPCD_QUAL_TBSP_DDL=NULL            # ddl file for tablespaces for qual
TPCD_QUAL_DDL=NULL                 # ddl file for qualification database
                                   # tablespaces and tables should be
                                   # identical to regular ddl except
                                   # container names
TPCD_INDEXDDL=dss.index.acme       # ddl file for indexes
TPCD_AST=dss.AST                   # sql for AST creation/population
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=dss.ri3                # 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_RUNSTATS=dss.runstats         # ddl file for runstats.  If you have
                                   # created indices before the load (ie
                                   # TPCD_EARLYINDEX=yes), have specified
                                   # to gather stats on the load command
                                   # (either through your own load script
                                   # or by using TPCD_LOADSTATS=yes, AND
                                   # you have specified a file for
TPCD_EXTRAINDEX
                                   # then this runstats file should
                                   # include the runstats commands
                                   # specifically for the extra indices.
TPCD_RUNSTATSHORT=NULL             # 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/tpcd/tpcd/appendix/dbgen
                                   # path name to data generation code

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

TPCD_TAILOR_DIR=/home/tpcd/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_CONFIGFILE=dss.loaddbconfig
                                   # 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=dss.loaddbmconfig
                                   # 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=NULL      # 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=NULL # 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/tpcd                # path for LOAD temp files
                                   # defaults to /u/<instance>/sqllib/tmp
                                   # used in load script only
TPCD_SORTBUF=4096                  # sortbuf size for LOAD
                                   # used in load script only
TPCD_LOAD_PARALLELISM=0            # degree of parallelism to use on load
```

```
                                # 0 = use the "intelligent default"
                                # that the load will chose at run time
                                # used in load script only
TPCD_COPY_DIR=/dev/null         # directory where copy image is
                                # created on load command CURRENTLY
                                # UNUSED used in load script only
TPCD_FASTPARSE=yes              # use fastparse on load
                                # used in load script only

                                # Backup and logfile specific
parameters follow:
TPCD_BACKUP_DIR=/home/tpcd/backupdir
                                # directory where backup files are
                                # placed
TPCD_LOGPRIMARY=50              # NULL/value = how many primary log
                                # files to configure.  If NULL is
                                # specified then the default is not
                                # changed.
TPCD_LOGFILSIZ=3500             # NULL/value = how 4KB pages to use
                                # for logfilsiz db cfg parameter.  If
                                # NULL is specified then the default
                                # is not changed
TPCD_LOGSECOND=5                # NULL/value = how many secondary log
                                # files to configure.  If NULL is
                                # specified then the default is not
                                # changed.
TPCD_LOG_DIR=/db2log2/tpcdmln.log # directory where log files stored..
                                # NULL leaves them in the dbpath
TPCD_LOG_QUAL_DIR=NULL          # directory where qual log files
stored
                                # NULL leaves them in the dbpath
TPCD_LOG=yes                    # yes/no - turn LOG_RETAIN on/off
                                # i.e. are backups needed to be taken
                                # CONFIG specific parameters
TPCD_CONFIGFILE=dss.dbconfig100.acme
                                # name of configuration file in ddl
                                # path that will be used for the
                                # benchmark run
TPCD_DBM_CONFIG=dss.dbmconfig100.acme
                                # name of config file for database
                                # manager cfg parms
TPCD_QUALCONFIGFILE=NULL        # name of database cfg file in ddl
                                # path for qualification database
TPCD_DBM_QUALCONFIG=NULL        # name of config file for database
                                # manager cfg parms
TPCD_MACHINE=acme               # set to NULL if using load config
                                # file big/medium/small  size of
                                # machine used to determine buffpage,
                                # sortheap,sheapthres
                                # and ioservers parms for load, create
                                # index and runstats
                                # NOTE that this parameter is ignored
                                # if a TPCD_LOAD_CONFIGFILE
TPCD_SMPDEGREE=1                # 1...# of degrees of parallelism to
                                # run with
TPCD_AGENTPRI=NULL              # set agentpri to this value (default
                                # is SYSTEM)
TPCD_ACTIVATE=yes               # activate the database upon build
                                # completion
                                # run specific parameters
TPCD_AUDIT=yes                  # no/yes
                                # no - don't set up qualification db
                                # stuff
                                # yes - set up qualification db
                                #       queries
                                #     - build the update function
                                #       tables
                                #       and data before we get into
                                #       the timing of the creation of
                                #       the tables and the load.
#TPCD_TMP_DIR=/home/tpcd/UFtempdir # place to put temp working files
TPCD_TMP_DIR=/UFtempdir         # place to put temp working files

TPCD_QUERY_TEMPLATE_DIR=standard  # 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=5                # number of streams for the throughput
                                # test
TPCD_FLATFILES=/home/tpcd/100GBUFfiles
                                # where to generate flat files
                                # for update functions
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=24           # number of chunks to split the update
                                # function into.
TPCD_CONCURRENT_INSERTS=12      # number of insert chunks that are run
                                # concurrently.  This number should be
                                # evenly divisible by
TPCD_SPLIT_UPDATES
TPCD_CONCURRENT_INSERTS_LOAD=8  # number of insert chunks that are
                                # loaded concurrently.  This number
should be
                                # should be evenly divisible by
                                # TPCD_SPLIT_UPDATES
                                # this controls the load portion of th
                                # insert routine for partitioned
                                # databases
TPCD_SPLIT_DELETES=480          # number of portions to split the
                                # delete function into.
                                # this variable is only valid in
                                # UNI/SMP mode.
TPCD_CONCURRENT_DELETES=4       # number of insert chunks that are run
                                # concurrently.  This number should be
                                # evenly divisible by
                                # TPCD_SPLIT_UPDATES
TPCD_GEN_UPDATEPAIRS=18         # number of pairs of update function
                                # data to generate
                                # if 0 the update data generation and
                                # setup will not be done.  use this if
                                # you don't want to run the update
                                # functions   (Update functions not
                                # fully tested in new env't yet)
TPCD_GENERATE_SEED_FILE=yes     # yes/no  These are the seed files for
                                # generating the query substitution
                                # values
                                # yes - generate a seed file base on
                                #       year/month/day (for audited
                                #       runs)
                                # no  - use qgen's default seeds
TPCD_RUN_ON_MULTIPLE_NODES=NULL # pe V1.2 only - will we be running
                                # each query stream of throughput
                                # starting at different nodes or from
                                # same node
TPCD_STATS_INTERVAL=30          # timing interval for vmstats/iostats
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
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=100              # max num of keys to delete at a time
                                # for UF2, use "default" for default.
TPCD_LOAD_SCRIPT=doload.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=NULL      # script to run for loading tables in
                                # TPCD_DDLPATH directory under mln/mpp
                                # for QUAL db
                                #  acid test specific information
TPCD_DB2LOG=/home/tpcd/sqllib/db2dump
                                # directory wehre the db2diag.log can
                                # be found for the durability tests
```

B.20 dss.loaddbconfig

```
update db cfg for tpcd using buffpage 80000;
update db cfg for tpcd using sortheap 50000;
update db cfg for tpcd using chngpgs_thresh 30;
update db cfg for tpcd using num_ioservers 21;
update db cfg for tpcd using num_iocleaners 4;
update db cfg for tpcd using util_heap_sz 120000;
update db cfg for tpcd using num_freqvalues 5;
```

B.21 dss.loaddbmconfig

```
update dbm cfg using sheapthres 81000;
update dbm cfg using intra_parallel no;
```

B.22 createUF2tbls

```
CREATE TABLE TPCDTEMP.ORDERS_DEL ( APP_ID INTEGER NOT NULL, O_ORDERKEY
INTEGER NOT NULL) PARTITIONING KEY (O_ORDERKEY) IN UFTEMP1;
alter table tpcdtemp.orders_del locksize table;
commit work;
```

B.23 buildUFTbl

```
#!/usr/bin/perl
# usage  buildUFTbl

($myName = $0) =~ s@.*/@@; $usage="
Usage: buildUFTbl\n";

die $usage if (@ARGV > 0);

# 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";
}
```

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

$dbname=$ENV{"TPCD_DBNAME"};
$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$uftemp=$ENV{"TPCD_UFTEMP"};
$delim=$ENV{"TPCD_PATH_DELIM"};
$inschunk=$ENV{"TPCD_SPLIT_UPDATES"};
$delchunk=$ENV{"TPCD_SPLIT_DELETES"};

# note this is a very slow way to do this, but I don't know any other
way
# to get a loop going of the number of tables

$crtblfile="crtbl.sql";
$chktblfile="chktbl.sql";
if ( -e $crtblfile )
{
  &rm("$crtblfile");
  &rm("$chktblfile");
}

open(CRTTBL, ">$crtblfile") || die "Can't open $crtblfile: $!\n";

print CRTTBL "CONNECT to $dbname;\n";

for (${chunk}=0,$tbsp=1; ${chunk} < ${inschunk}; ${chunk}++,$tbsp++)
{
    print CRTTBL "CREATE TABLE TPCDTEMP.ORDERS_${chunk} ( 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) IN
$uftemp$tbsp;\n";

    print CRTTBL "CREATE TABLE TPCDTEMP.LINEITEM_${chunk} ( 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) IN $uftemp$tbsp;\n";

    print CRTTBL "COMMIT WORK;\n";

    print CRTTBL "ALTER TABLE TPCDTEMP.ORDERS_${chunk} locksize
table;\n";
    print CRTTBL "ALTER TABLE TPCDTEMP.LINEITEM_${chunk} locksize
table;\n";
    print CRTTBL "COMMIT WORK;\n";
}

print CRTTBL "connect reset;\n";
print CRTTBL "terminate;\n";
close(CRTTBL);

system("db2start");
print "Now creating holding areas for the update functions in the
database ...\n";
if ( -e "$auditDir${delim}auditruns" )
{

$outputMsgFile="$auditDir${delim}auditruns${delim}rawdata${delim}crtbl.m
sg";
}
else
{
    $outputMsgFile="crtbl.msg";
}
print "Verification info can be found in $outputMsgFile \n";
system("db2 -vtf $crtblfile > $outputMsgFile");

print "Verifying that the temporary tables were created successfully
...\n";
open(CHKTBL, ">$chktblfile") || die "Can't open $chktblfile: $!\n";
print CHKTBL "CONNECT to $dbname;\n";
print CHKTBL "select creator,name,tbspace from sysibm.systables where
creator = 'TPCDTEMP';\n";
print CHKTBL "connect reset;\n";
print CHKTBL "terminate;\n";
close(CHKTBL);
system("db2 -vtf $chktblfile");

1;
```

Appendix C: Query Text and Output

C.1 Qualification Query Output

Qualification Query 1

```
QRYQUAL [1-17]
**************************
QUERY 1
**************************
-----------------------------------------------
--#SET ROWS_OUT -1 ROWS_FETCH -1

Tag: Q1     Stream: -1   Sequence number: 1

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 DAYS
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                    3773034.000        5319329289.680
     5053976845.784        5256336547.676               25.510
   35964.013                0.050          147907
N            F                  100245.000         141459686.100
      134380852.769         139710306.872               25.625
   36160.451                0.050            3912
N            O                  7464940.000       10518546073.980
     9992072944.461       10392414192.063               25.542
   35990.126                0.050          292262
R            F                  3779140.000        5328886172.990
     5062370635.934        5265431221.821               25.549
   36025.461                0.050          147920


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

Qualification Query 2

```
**************************
QUERY 2
**************************
-----------------------------------------------

Tag: Q2     Stream: -1   Sequence number: 7

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

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------------------
           9828.210  Supplier#000000647         UNITED KINGDOM
        13120  Manufacturer#5             jB16PyPyB7Bl52jMjSPw3mS
           33-258-202-4782  z1QhSiMj11Bm7COlLwh6Ql0B1R2Mg4CLn
LhiP0wiMzy72hlkP7l5in2y6RS6N130lz51nSRL5gOg5S26hPCCQN2L
           9508.370  Supplier#000000070         FRANCE
         3563  Manufacturer#1          M5C616R5h5SlMR3zzmLkSw24j2
           16-821-608-1166  m7z0CPShmBkhlChBAi3LkQ2CLw
mhl6QP362RPS3044CB2y41yhOhjlBin0CL7yhxmhS4hBM07kQ1yyjOjz3C

           9508.370  Supplier#000000070         FRANCE
        17268  Manufacturer#4             M5C616R5h5SlMR3zzmLkSw24j2
           16-821-608-1166  m7z0CPShmBkhlChBAi3LkQ2CLw
mhl6QP362RPS3044CB2y41yhOhjlBin0CL7yhxmhS4hBM07kQ1yyjOjz3C

           9453.010  Supplier#000000802         ROMANIA
        10021  Manufacturer#5
5yARQNSLNRAlOlBnkNQCik3SOlyClk7nmRhA2h0   29-342-882-6463
65y3RQ2i0OP6Nz7mS hC
PxwLy7L1jQy6Ol63xO3iBCz52Rm1zm0MziCMLij2n6wky51mBOwx Qh52iz QB1545Amxyj

           9453.010  Supplier#000000802         ROMANIA
        13275  Manufacturer#4
5yARQNSLNRAlOlBnkNQCik3SOlyClk7nmRhA2h0   29-342-882-6463
65y3RQ2i0OP6Nz7mS hC
PxwLy7L1jQy6Ol63xO3iBCz52Rm1zm0MziCMLij2n6wky51mBOwx Qh52iz QB1545Amxyj

           9192.100  Supplier#000000115         UNITED KINGDOM
        13325  Manufacturer#1
h0m3lzlSPMw2B0ny7LNyNCkjRRn7iyMlLBLA      33-597-248-1220  1QzQjhSyx
ixm2lgz2Ry7075RL3MS5z36x56hxmR0wLN0LBxm164LzCMmALzOAjn4kz7i4wjOlCONl1C51
M7nCMx66SBRAQA
           9032.150  Supplier#000000959         GERMANY
         4958  Manufacturer#4              205LNCzxMCnQ5gnz4n S3ynP6Mhnw
           17-108-642-3106  Px z7kOx56l7jQz NwBBQhky yM7kLgxRQw5zw6
426Bm55lC6 OkQ7hQPLixjM7y47BNP16CRi0kjk354lgxh
           8702.020  Supplier#000000333         RUSSIA
        11810  Manufacturer#3
5iwkgN5n2BN15OmQk2602h0N6NzxPyiPN5lnj     32-508-202-6136
SgimAjmn3wL7Rlxmh3LCwOPnhjyl 7xxzxAN 4ACx43y65NwQ7P

           8615.500  Supplier#000000812         FRANCE
        10551  Manufacturer#2              h4i2M2O0
ky1g2mlBOmxjzj0hA2h6nkSNhP       16-585-724-6633
57i0NAyR0RP2jOh54C6B22OlSL

           8615.500  Supplier#000000812         FRANCE
        13811  Manufacturer#4              h4i2M2O0
ky1g2mlBOmxjzj0hA2h6nkSNhP       16-585-724-6633
57i0NAyR0RP2jOh54C6B22OlSL

Truncated at 10 rows

Number of rows retrieved is:  44
-----------------------------------------------
```

Qualification Query 3

```
**************************
QUERY 3
**************************
-----------------------------------------------

Tag: Q3     Stream: -1   Sequence number: 13

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
-------------------------------------------------------------------
     260930            320547.253  1995-03-12               0
     402497            298879.532  1995-02-12               0
     457859            296490.675  1995-01-17               0
     509889            294068.874  1995-02-03               0
      58117            292632.833  1995-02-21               0
     538311            279665.996  1995-03-07               0
     588421            275477.117  1995-03-03               0
     416167            273765.453  1995-02-22               0
      97830            273227.061  1995-03-04               0
      90276            272233.917  1995-03-04               0


Number of rows retrieved is:  10
-----------------------------------------------
```

Qualification Query 4

```
**************************
QUERY 4
**************************
-----------------------------------------------

Tag: Q4     Stream: -1   Sequence number: 2

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 MONTHS
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                 999
2-HIGH                  1002
3-MEDIUM                1021
4-NOT SPECIFIED          997
5-LOW                   1089


Number of rows retrieved is:  5
-----------------------------------------
```

Qualification Query 5

```
**************************
QUERY 5
**************************
-----------------------------------------

Tag: Q5     Stream: -1   Sequence number: 14

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
-----------------------------------------------
CHINA                            7349391.471
INDONESIA                        6485853.403
INDIA                            5505346.820
JAPAN                            5388883.594
VIETNAM                          4728846.602


Number of rows retrieved is:  5
-----------------------------------------
```

Qualification Query 6

```
**************************
QUERY 6
**************************
-----------------------------------------

Tag: Q6     Stream: -1   Sequence number: 6

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
----------------------
       11450588.043


Number of rows retrieved is:  1
-----------------------------------------
```

Qualification Query 7

```
**************************
QUERY 7
**************************
-----------------------------------------

Tag: Q7     Stream: -1   Sequence number: 16

SELECT
SUPP_NATION,
CUST_NATION,
YEAR,
SUM(VOLUME) AS REVENUE
FROM
(SELECT
N1.N_NAME AS SUPP_NATION,
N2.N_NAME AS CUST_NATION,
YEAR(L_SHIPDATE) AS 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, YEAR
ORDER BY SUPP_NATION, CUST_NATION, YEAR

SUPP_NATION               CUST_NATION               YEAR
REVENUE
-------------------------------------------------------------------------------
-----------------
FRANCE                    GERMANY                   1995
   4611421.440
FRANCE                    GERMANY                   1996
   4828420.372
GERMANY                   FRANCE                    1995
   6755766.841
GERMANY                   FRANCE                    1996
   5810951.396


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

Qualification Query 8

```
**************************
QUERY 8
**************************
-----------------------------------------

Tag: Q8     Stream: -1   Sequence number: 10

SELECT
YEAR AS YEAR,
SUM(CASE WHEN NATION = 'BRAZIL'
THEN VOLUME
ELSE 0
END) / SUM(VOLUME) AS MKT_SHARE
FROM
(SELECT
YEAR(O_ORDERDATE) AS 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 YEAR
ORDER BY YEAR

YEAR         MKT_SHARE
-----------------------------------
       1995                 0.055
       1996                 0.089


Number of rows retrieved is:  2
-----------------------------------------
```

Qualification Query 9

```
**************************
QUERY 9
**************************
-----------------------------------------

Tag: Q9     Stream: -1   Sequence number: 17

SELECT
NATION,
YEAR,
SUM(AMOUNT) AS SUM_PROFIT
FROM
(SELECT
N_NAME AS NATION,
YEAR(O_ORDERDATE) AS 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, YEAR
ORDER BY NATION, YEAR DESC

NATION                    YEAR         SUM_PROFIT
-------------------------------------------------------------
ALGERIA                   1998           1946316.005
ALGERIA                   1997           2973825.692
ALGERIA                   1996           3308881.516
ALGERIA                   1995           3092227.299
ALGERIA                   1994           3406958.710
ALGERIA                   1993           3140744.026
ALGERIA                   1992           3330704.407
ARGENTINA                 1998           3045410.008
ARGENTINA                 1997           4255378.593
ARGENTINA                 1996           4651751.937

Truncated at 10 rows

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

Qualification Query 10

```
**************************
QUERY 10
**************************
-------------------------------------------

Tag: Q10    Stream: -1   Sequence number: 4

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 MONTHS
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

-------------------------------------------------------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-
       9722  Customer#000009722                  464618.258
  474.040  CANADA                    l Mwzn4NAk6j
       13-518-602-8070  5L 50Oy
RSgBAzPxmOSi5wk6xxOR7kh2nnPlgy7LBng2hOw5B01
RmCM120L24Pkg7PS1zwC11BCnz4L6i15PkixP26l66
      12800  Customer#000012800                  444265.642
 1900.840  PERU                      57zjB3CQx4P4OB2R2MBi2mwhSllM4mn 4
nC6     27-142-205-3552  0hwglS77RB56Rx436lQ0N16CxhOPnmyhgwz
5z64wnj1kiC4jL350mM41y71hNxBllPjyA4hiN1wzjjM7SCxAN244mk2A

       1025  Customer#000001025                  442028.022
 3363.460  INDIA                     lkiSn154M5ROi
       18-588-456-4616  0Bl45z233Rniw0OO064nPBgP16kimO0y74iLh73g1N4 m310
jQ yQzPA50iC 3MA75g2Bj162Nw4P
      13028  Customer#000013028                  441692.240
 -452.660  UNITED KINGDOM            yP714ORSNgNN2LA3L5B
       33-253-660-2127  xPkmnhL2BkhkNyww4khlxwwAymN
h11PSjBCNMi50LkyOhO6CC 5nzOQCALzliOk2R66w l05hRPO3iSP

       3694  Customer#000003694                  438180.070
 2960.440  UNITED KINGDOM            2CCklmCBOCC
       33-421-331-3127  MzLxQxLlLx3MPxlAwg1B5kg61zxkPnk
xiAm6PhMMAAQ2nzN3S6zzgP x70w0lhhPx4QRzlMMy0204lAl3mBO7jh2jAP0N60wg367z

        976  Customer#000000976                  435897.632
 7772.850  ROMANIA                   QzR 56Px1kgS  wANnAz02RS 30n Pm
       29-436-660-4732  kzn32776 gwzkMzzzO4yxOAnkR7hR4R4x2SMwilz3x6h
nN7OnNLRMml3 kz5SLwi1yklOxiwS4g0wmA5A 4hmgBSwRRiQ1

       8206  Customer#000008206                  429905.110
 6046.360  ARGENTINA                 P yMg30BBBBx NMgC03AmzN2
       11-571-859-1370  hLi122RMPmLC36Oy0kxO71zz2wCR0QQCl7z26hlQ3mM

      13532  Customer#000013532                  427731.804
 -924.180  KENYA
6ij7M5PBMx2kwwyz62Oj4SL5S0mRCwl3m1Rmw     24-525-332-7244
7ih7yRz214zO67AiNPx64nO5l5k yj6i3jLA5PCL15Q4QlA3ll60iM1P iBxCixg6
1hCh2RCnjOzk5R OnO 1OhhC3m463lm5
      12745  Customer#000012745                  422327.693
 9691.330  CHINA                     SgS1LMC4gB2NM3wh
       28-985-189-6174  jl72wjSw0 S6 7L4Cgxw PkyO5Nl2LL7LBR

       2344  Customer#000002344                  411240.109
 5597.220  MOROCCO                   O3PC7ikBgw OAzPAlm2P
426zm3BnBN6Q1O 6N    25-593-745-7663  5NBn0wRNngLw2z5kyn1AhL0ASyg6SMhM
i2kMOyxARAnlO0Q5j4CBNARix7ABlMAC

Truncated at 10 rows

Number of rows retrieved is:  20
-------------------------------------------
```

Qualification Query 11

```
**************************
QUERY 11
**************************
-------------------------------------------

Tag: Q11    Stream: -1   Sequence number: 5

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) * 0.0010000000
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
-----------------------------------
      12098          16227681.210
       5134          15709338.520
      13334          15023662.410
      17052          14351644.200
       3452          14070870.140
      12552          13332469.180
       1084          13170428.290
       5797          13038622.720
      12633          12892561.610
        403          12856217.340
       1833          12024581.720
       2084          11502875.360
      17349          11354213.050
      18427          11282385.240
       2860          11262529.950
      17852          10934711.930
       9871          10889253.680
      12231          10841131.390
       6366          10759786.810
      12146          10257362.660
       5043          10226395.880
      12969          10125777.930

Number of rows retrieved is:  22
-------------------------------------------
```

Qualification Query 12

```
**************************
QUERY 12
**************************
-------------------------------------------

Tag: Q12    Stream: -1   Sequence number: 11

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                   654             950
SHIP                   684            1004

Number of rows retrieved is:  2
-------------------------------------------
```

Qualification Query 13

```
**************************
QUERY 13
**************************
-------------------------------------------

Tag: Q13    Stream: -1   Sequence number: 15

SELECT YEAR,
SUM(REVENUE) AS REVENUE
FROM
(SELECT YEAR(O_ORDERDATE) AS YEAR,
L_EXTENDEDPRICE*(1-L_DISCOUNT) AS REVENUE
FROM TPCD.LINEITEM, TPCD.ORDERS
WHERE O_ORDERKEY = L_ORDERKEY
AND O_CLERK    = 'Clerk#000000088'
AND L_RETURNFLAG = 'R') AS PERFORMANCE
GROUP BY YEAR
ORDER BY YEAR

YEAR         REVENUE
-----------------------------------
       1992          1262855.731
       1993           964121.033
       1994          1750395.294
       1995           198820.299

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

Qualification Query 14

```
**************************
QUERY 14
**************************
-------------------------------------------
```

```
Tag: Q14    Stream: -1   Sequence number: 9

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
---------------------
              16.729


Number of rows retrieved is:  1
-------------------------------------------
```

Qualification Query 15

```
**************************
QUERY 15
**************************
-------------------------------------------

Tag: Q15c   Stream: -1   Sequence number: 3

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 MONTHS
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
-------------------------------------------------------------------------
-------------------------------------------
        389  Supplier#000000389         PB1Lx0xx6LMz3h7Rx63m6j3QmMx
         34-885-883-5717           1418538.214


Number of rows retrieved is:  1
-------------------------------------------
```

Qualification Query 16

```
**************************
QUERY 16
**************************
-------------------------------------------

Tag: Q16    Stream: -1   Sequence number: 8

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 '%Better Business Bureau%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#14    SMALL ANODIZED NICKEL              45             12
Brand#22    SMALL BURNISHED BRASS              19             12
Brand#25    PROMO POLISHED COPPER              14             12
Brand#35    LARGE ANODIZED STEEL               45             12
Brand#35    PROMO BRUSHED COPPER                9             12
Brand#51    ECONOMY ANODIZED STEEL              9             12
Brand#53    LARGE BRUSHED NICKEL               45             12
Brand#11    ECONOMY POLISHED COPPER            14              8
Brand#11    LARGE PLATED STEEL                 23              8
Brand#11    PROMO POLISHED STEEL               23              8

Truncated at 10 rows

Number of rows retrieved is:  2762
-------------------------------------------
```

Qualification Query 17

```
**************************
QUERY 17
**************************
-------------------------------------------

Tag: Q17    Stream: -1   Sequence number: 12

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
----------------------
           24436.880


Number of rows retrieved is:  1
-------------------------------------------
```

C.2 First 10 rows of Test Database Tables

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

R_REGIONKEY R_NAME                    R_COMMENT


----------- -------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
--------
          0 AFRICA
xSx31zz31Cl1z4OAnmm05AjiOxC3AMMNOgC0kACgwngg3glP7LLLywlQy7R


          1 AMERICA
kgyh3LSnC72k6zlAz0LP3k2L4QB1QL1O673OjO1SPj0ngQ7CO100SBgmgRQ4lgPCMk21A425
iklyAR4yBRAwR4Cm5miNw 4jl13mMnxw17B

          2 ASIA                      NSg6xlMlA1lzm6mOR0Ajx
nhRA77NgRxBwL1M6Py RjySB3RLwkyPkwMM2R1BQ
xAzkOgkjml10gAghinP5inmNmR76MlijMS3S2zxONR15

          3 EUROPE
zlSL7Qwg12hMBL5lhlz0M45QkjShwSyiO04MLOh7wn1ARLQPyPAyAiil576l1Li7AlnR1S
RQ4SLny7B2Ryj5P66MLhn NxhwB4C3ig0SO

          4 MIDDLE EAST               RllxmhPLz3Cy2mNlg4QMBnNASM ACki
MPki7Oi


  5 record(s) selected.


SELECT * FROM TPCD.NATION FETCH FIRST 10 ROWS ONLY

N_NATIONKEY N_NAME                    N_REGIONKEY N_COMMENT


----------- -------------------------- -----------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
--------
          0 ALGERIA                             0 2Cxhl7
L1iwk6hMh300izngN32CPwCikyLk6khMzSRA

          5 ETHIOPIA                            0 NS7n LSOP
Oz5n1AlB2S02nN0lMh4SBxP iRhBO 047R26 2BlM

         14 KENYA                               0
yA0jNSxmS25NRiCy1B6yhSL4NLMROQ3kLyAPx43wil27kN1QmPLmPw7ywh2kyj2iSBwSjx26
zClx

         15 MOROCCO                             0
7mC2hnARnj21nwllAOhR2nwRmgz22AkP6xj3iQiPNB0PjxBMzS3g
4y0L4Alx1gNCAP2kjQMBR2xwiiSNLgCMQx74RmhMy4OONhPCNQQ 3y

         16 MOZAMBIQUE                          0
LB6SkSOmkznRLS4z5P2BOmC23RnA6h7mn0gPAAN7nkxy6jO0k3w3R32R6A1ASk0LMyngO36j
CL1gl5wQw4AMPC

          1 ARGENTINA                           1
zQn3Okwz1wLn7PLS3OhCgn56kP5PyRikgi1B7lL


          2 BRAZIL                              1
gLmS0nACAmnBCj2klki7RCPNgPxnCOjNg4k OiAg57COSOm1NwCnOyLx40R SC
y20gPPAkNk5hxRhR5OmgS1iPQQzNAxPL30n67OgyC l617Sh4LS

          3 CANADA                              1 4yMO AhnQ5Lh
wzQAM662Aw1ByCl7CxmzRwNR5nAlO4 x

         17 PERU                                1 35h63
yS7gjRh22iyBlhy5wjLMQnln74k0j5O ygx1miQgA66hyhP3740PQOgRkwhy1i3LSwzjB
3yjh0QnyMMN A7Qn 6NOC2jg65QRLAgS

         24 UNITED STATES                       1
QQ41AxzBSQACRA1wCLyONCi5yzCyljQR144lnNLm2 Q7PRyk BRzP


  10 record(s) selected.


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

```
P_PARTKEY   P_NAME
P_MFGR                    P_BRAND    P_TYPE                     P_SIZE
  P_CONTAINER P_RETAILPRICE           P_COMMENT
----------- ---------------------------------------------------
------------------------- ---------- -------------------------
----------- ----------- ------------------------- ------------------------
         10 slate dark white lavender purple
Manufacturer#5            Brand#54   LARGE BURNISHED STEEL
44 LG CAN        +9.10010000000000E+002 wPP74M1Lwj1
         14 grey beige tomato cornflower
Manufacturer#1            Brand#13   SMALL POLISHED STEEL
28 JUMBO BOX     +9.14010000000000E+002 PA4yml7LMzm3COm4COQ
         37 indian plum olive royal spring
Manufacturer#4            Brand#45   LARGE POLISHED TIN
48 JUMBO BOX     +9.37030000000000E+002 j45 j
         80 sandy plum goldenrod peru wheat
Manufacturer#4            Brand#44   PROMO PLATED BRASS
28 MED CAN       +9.80080000000000E+002 zn25ywOxRB
         89 seashell midnight pale rose sky
Manufacturer#5            Brand#53   STANDARD BURNISHED STEEL
7 MED JAR        +9.89080000000000E+002 Chg145
        126 violet almond medium pale blue
Manufacturer#4            Brand#45   MEDIUM BRUSHED NICKEL
4 LG BAG         +1.02612000000000E+003 1wn5yNBA
        167 royal powder plum white sky
Manufacturer#3            Brand#32   LARGE ANODIZED STEEL
46 WRAP BOX      +1.06716000000000E+003 z7S2 m
        169 beige deep misty olive
Manufacturer#5            Brand#55   STANDARD POLISHED BRASS
10 JUMBO CASE    +1.06916000000000E+003 BLnli3n Oy3Bl6A
        170 gainsboro misty red pale olive
Manufacturer#3            Brand#33   LARGE POLISHED COPPER
28 LG DRUM       +1.07017000000000E+003 M C2PBnkSgjgxj5CBjS
        178 chiffon navy puff tomato linen
Manufacturer#1            Brand#13   STANDARD POLISHED TIN
10 LG JAR        +1.07817000000000E+003 m14xnMQOBzwkz 5lhBg

  10 record(s) selected.


SELECT * FROM TPCD.SUPPLIER FETCH FIRST 10 ROWS ONLY

S_SUPPKEY   S_NAME                    S_ADDRESS
      S_NATIONKEY S_PHONE         S_ACCTBAL               S_COMMENT


----------- -------------------------
---------------------------------------- ----------- ---------------
-----------------------
-------------------------------------------------------------------------------
----------------------------
         10 Supplier#000000010        wN1S4mQ0g7Px5Lj34xw6kS4Li4NzB4mO
               24 34-852-489-8585   +3.89191000000000E+003
5xwg6AOz0NzhONL6kC43zR3AhzO6njCiwPg7k6MxwP1mN2 Rg 5Q426

         14 Supplier#000000014        l4iQ47yygAilnQiS2yg2
               15 25-656-247-5058   +9.18982000000000E+003
B34knh6m3SR45g42Lnl7xBixy0QQ0yClLCS0xjSlAl3P4AnxBy4zk

         37 Supplier#000000037        mRPgxC4nPnLjNmCPP23yCjL7j7j
                0 10-470-144-1330   +3.01747000000000E+003 n17BjLwNQMNMk
 M5i4SNRRnjOwy

         80 Supplier#000000080        LO2CSB1kSOx35hg1y7ny3
               21 31-646-289-1906   -4.04500000000000E+001 QN5m2gCL
SCRz7  74BmQjOOiQz5QC0

         89 Supplier#000000089        P2Bx3wm36zASOCg
                9 19-259-876-1014   +1.63802000000000E+003
A2LLwAP7M7mykmAi6k4Rj1OB71x4MiQl6Cy07xj207BP

        126 Supplier#000000126        SOlyR6klBOnx0QyOkhQ0wwzMgSj
zg57k1kxS             14 24-728-670-3468   +6.82986000000000E+003
N0Cy3C32177P3hBByjmz5zj6SnwmSgCkz gB  g 0Olxk

        167 Supplier#000000167        BQlnNlyhQA  gByOy
               20 30-725-194-1727   +5.73187000000000E+003
L0337y27MR2mQQwzPx36263kxxQmh55mhAhh577ji4726O01
Qzw5C5z7hAPiBiyjAiPi7mz6OjS3n3CmymlgNyPh7h1MQ
        169 Supplier#000000169        xRAyR01Pn70LRNn
               13 23-698-509-1073   -9.27500000000000E+002
x12R3S2O2O7k14kgii3N052gQNm5Siyg

        170 Supplier#000000170        PCxjjzNQihLNxgLw0SiMmQ
               23 33-803-340-5398   +7.39278000000000E+003
M116S1xzg54iC3k7OPLQi3Cimhghz2BClQk g5Ag12QSBlhglANnw4MR MBS 72A

        178 Supplier#000000178        ww R0Pz 6O4P34LSy7jj4R6
               16 26-471-122-2582   +4.69327000000000E+003
762LSnPNBj4zB4mijnC2ljkB0hRiROBm1gCzC
0M0SAQiM1z2Ajw4QSn2j3Slk5lLyBh0MNQLCP06O

  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
M462AQAARxAO yOilNgQx0lC 6n7kA5CmNQmS3xC1QSQ7hwlM46hxR OOMCSMR n0
QOwk6QzM1hO1AjO6CkxixRS1xRgCNCQhNh532n3Bmy7zlyi3

         10      250011        3335   +6.73270000000000E+002
hm72RBRkAOhgSC7 BAk6NR501O
6zSS0kPjSlywCihCO6x5SzQPk7nkS6C42mmLM0gCiQnC0Q3xSRClPmN4
OkjmROh52jNzk5QPNx1mBgA5zh0R kzjP0gAAN 6g2AQCBl1wk0j jil4ClgNQzCL14A6iC
P4RSNQ
         10      500011        5691   +1.64000000000000E+002
wlA2mRAQw6x7MCC6MwPP1gP02A
ACN5yBOM4Cn4jzS11M1CCnj4h6gQR41jSwgNwzPQn7Sj23O676OSwhikxRO7xmC6
LxznzPgzPm6Nmh2APOliw2yLjh2hlig44y2MgMNgPj3RyBny7jNwQCS3iAwMgzS3MBRACiC4
Qnmwy3SyjOOC5l6
         10      750011         841   +3.74020000000000E+002 CmnMwAg2x 3
650QMSnMSzLhPClB0hnxx057wkgiAAghPw
hPlyyijj2Owm7gzQRlSh62P7yOzw1ml30Ny4igSmQ7NPljz0w2y
B0RNSL7j211wOMzSMLm2R0PlC hllCL61

         14          15        5278   +6.50070000000000E+002
OgOCmlxnSySiBB7nO13m5S1Cy33PCgMC25PymCwAQAgkM3l1CgBn0 gRinN07x3L2SM0hCC
nPhhx iPl hxhAkCAByh2BLnlPS1kLwB2xj350h4LBBR1OBz2nm7Czw 61yjjlhSi5Rh

         14      250015        5334   +8.89500000000000E+002 66B5Sx
nAPhOwM1Ay5iiMPB2AzQk206M34Pij7M53Q6xN30AQP4QBPSz16yg4NxM2QNnjQMBhBy6i7C
Sk6P2ML1Mg77wA0iBLR5BA2

         14      500015        3676   +8.93390000000000E+002
xmlxnMl0BNkgRLLPkA 4kCjOL16A4NC0MRMMh
6Ox2khBLL0mMjxlOLQyLMB02w3lNyMBzl3ihPLnkM7BShQwBx0QlmwyhP466i3S4jnQPzPzB
wAQgR5wzl57hLRn3OjQwm2gnLNwy23Qy

         14      750015        4947   +3.10130000000000E+002
57xxill0zQ0x4zLLg4 yiBnxOO
jlAngNPmPzh7ml35QgN31zzzBi41nlkSzg6Cl1Czg62lwAi72wm53wLO7CBL4xwS7lx5OPQ2
xg6Llmy3R1n7zzOAzNA2khA7zxB6x1zRmykg

         37          38        7171   +8.24960000000000E+002
Cwnj0mnCjjRRxRLg5yA 4Nw62L1BwlyRNyyS7xMnyNlxM mmNnAAyhQOO5wn70nR
hz7OSyMwhySkN3Oz6i20NmwAA7zO3h7gl kS1R0hgNw
xC64mO42glQ7j2R41kNBSyRRj04gwmS4R40 hjQ5BNk3AB57jyALQNx2xnQi6zhzlM

         37      250038        5542   +1.26590000000000E+002
zjj0BL323NOiPCjNwjmQNhjS nCNk4j3C5hj7725RiRCOP6
4mNLQA0hggyhSNPiCgC7jhw33x1MMlSBy1mLO1SgQQ1jn34j11jP



  10 record(s) selected.


SELECT * FROM TPCD.CUSTOMER FETCH FIRST 10 ROWS ONLY

C_CUSTKEY   C_NAME                    C_ADDRESS
      C_NATIONKEY C_PHONE         C_ACCTBAL               C_MKTSEGMENT
C_COMMENT

----------- -------------------------
---------------------------------------- ----------- ---------------
------------------------ ------------
-------------------------------------------------------------------------------
----------------------------------------
         10 Customer#000000010        L3jg3xAwi6A0B103B0Aymm
                5 15-741-346-9870   +2.75354000000000E+003 HOUSEHOLD
7Lm LiCwwxQMykgNOR6kzCyP1B21QyA57hBlSOPnx6m53iSOP6w44M3CP
MnP7Alky4OwkOwSh20341
         14 Customer#000000014        l2AjL Q3jizLz4L1 n2y lyBPx62L1PkOS
3               1 11-845-129-3851   +5.26630000000000E+003 FURNITURE
2QO6MN6my4235j4kMww7Rn yLk00zj0mm ywi3lgg0B2Q75yShl7wm6xy5xgSzS3Nmx2
2w76wk jij
         37 Customer#000000037        RminMM52SinCxBy0ljj45R
                8 18-385-235-7162   -9.17750000000000E+002 FURNITURE
37jk5AxRm2l2wQ4AknhRmhN 2 CPB2PimMRM2Blh7SkMQjCxz2yAQ0PQkMLPygy2
2winzCy517iLnCjlxkwQix1Ogh2nylyC7Ly1jO136z266
         80 Customer#000000080        kmhA2zARBR4CNxl  z2iN3
                0 10-267-172-7101   +7.38353000000000E+003 FURNITURE
g700xy0A2lzQnl5m7gPQxiBw5Q1iQ1LSCN7xn2iwL3LMR3h

         89 Customer#000000089        xhg3n6y B7Pj7AiOkL31g glij2Qx
               14 24-394-451-5404   +1.53076000000000E+003 FURNITURE
4mnkMCjLkPP6AhR0B4QgxM5RMwMjP1PS2yLN51MMilyBPjP5g24w

        126 Customer#000000126        lLCnS61Mxk165PP6n7jBwlkjBN16
               22 32-755-914-7592   +1.00139000000000E+003 HOUSEHOLD
NP1g
N2RkO14QPjS7Lim6kSj3hPBLBQAhNjjgCS5COSiwi6R227kQi74RB7gwCj3R3AkAjRj3
33OzNAkjBCQ246k
        167 Customer#000000167        z65NhmhiOhy1zO4201whgLgCn
               5 15-288-395-5501   +1.46809000000000E+003 AUTOMOBILE
wkOhzgAL1BC7N2BMnSgiCzRMzSmnBLMim51C

        169 Customer#000000169        6i357 1O55lx
               18 28-362-499-3728   +4.48383000000000E+003 FURNITURE
15BCRyOPmM PwN1 iRxA36P
PyBnh5gCB51zjynm3M505NjLyi37nPh50626QBNyLz5L7QRgRkO2QnzQnPNCkkO27SQzR6P

        170 Customer#000000170        A64mn 0wlM0Q0njyP7OLNNmg7CM0
               15 25-879-984-9818   +7.68789000000000E+003 BUILDING
zy4mQ3PRngizQOw0lkxy6kw4OzxxnQPwwh5MPkS6BClmw7zPkz 26xRmymiALQ
14Ai0S7xSMNjgiSNhR0glAhnwAwLyj66QlSyCzzLl4niQ
        178 Customer#000000178        gPNzxz PAzjz716SwS
               21 31-436-268-6327   +2.27250000000000E+003 FURNITURE
B73L65 OC1ll56MnngOxklPOwAPN1m1SRxMMOPj0
j2135mlPMwBgROAM3jzOALPmSnkPAmSMyOim2z0z30OhjN2kw k4wCxPAC364km2B2m


  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

----------- ----------- ------------- ----------------------
---------- --------------- --------------- --------------
-------------------------------------------------------------------------------
-------
         37     3211259 F               +1.21339030000000E+005
06/03/1992  3-MEDIUM        Clerk#000045557               0
OOQ5iMMS3lCk3i132lmw2h06QQ74hP7nj66MQLOMj2

        167    12269245 F               +9.91701000000000E+004
01/04/1993  4-NOT SPECIFIED Clerk#000073068               0
wg0h3l41PjBQP41PMMLi3CkRwOMBPNmQmO4Q k7 zixxnikzyPALwwllyBj27N

        197     2992804 P               +1.83017470000000E+005
04/07/1995  2-HIGH          Clerk#000096831               0 BAyQj
```

```
3Q5hySBkC7P5Myzg2PnRMLln4QiiQm4N LLi05 7C

        421     12512053 F               +1.49311600000000E+004
02/22/1992  5-LOW           Clerk#000040478              0 2nCCNNnByRMx2
AOhhkjSQyBLwg4QN2M0xNPynm1yiOL55 3A7Lik3kO2inwjg
        550     1633414 O                +2.19428800000000E+004
08/02/1995  1-URGENT        Clerk#000020362              0 4A mO3 AAN
BwMN3yhRQMxkzw3P4hMn6lwO
        578     3829804 O                +8.07462600000000E+004
01/10/1997  5-LOW           Clerk#000028024              0
jih0wPBLkClCnwMMi3C4 gO1g4MmlRCQyByBgzR AAAOS245Q4QCz47Oyz7RgkA7N mRMl02

        612     13095157 F               +2.44825910000000E+005
10/21/1992  3-MEDIUM        Clerk#000075869              0
34imSQi5yP42PkiijQ45RnMRgk72g4Skz0kxy l2B3Mz7gPhQyL36i2LLz2SQC13

        647     10267574 O               +1.76881730000000E+005
08/07/1997  1-URGENT        Clerk#000026918              0 Mw
z6jR3yn7yS5Sn71OPz0wN6yj1lL yyygPPPQ3 gn3 36hNCk1w0N246N

        741     3511036 O                +6.86657000000000E+003
07/07/1998  2-HIGH          Clerk#000029426              0 1LnBy37yC24
P7k M1yCACOS5iwwyjk4xjhg2
        834     11632558 F               +6.41156900000000E+004
05/23/1994  3-MEDIUM        Clerk#000080463              0
A2hl2NL77n14OPk14RBA1QnSwBjROAknONiilwCg


  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

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

         37    16307205      307206             1
+2.80000000000000E+001   +3.39189200000000E+004   +3.00000000000000E-002
  +4.00000000000000E-002 R            F            08/13/1992 08/31/1992
  09/12/1992    NONE                      FOB        5A65x4R0Lzw75jzSzSN

         37    15189381      939427             2
+3.80000000000000E+001   +5.58459400000000E+004   +8.00000000000000E-002
  +1.00000000000000E-002 R            F            07/05/1992 07/18/1992
  07/12/1992    DELIVER IN PERSON         TRUCK
L7PzwC7Q1R4zR5jBg6hm6BBNgx2OyBLQS1
         37     7925015      175023             3
+3.50000000000000E+001   +3.63867000000000E+004   +6.00000000000000E-002
  +3.00000000000000E-002 A            F            06/09/1992 07/17/1992
  06/15/1992    TAKE BACK RETURN          FOB        0O4Pzk3
7QA4AizM7wjNxA Mn1L7 wngwkO RkLj
        167    13408154      158194             1
+4.20000000000000E+001   +4.45821600000000E+004   +2.00000000000000E-002
  +2.00000000000000E-002 A            F            02/10/1993 03/11/1993
  02/11/1993    NONE                      RAIL
04myNzQ1BNA3y5xhnB02B2B
        167    10833701       83712             2
+3.60000000000000E+001   +5.88297600000000E+004   +9.00000000000000E-002
  +2.00000000000000E-002 A            F            04/27/1993 02/08/1993
  05/23/1993    DELIVER IN PERSON         AIR        3P03gCy33SQym7LzlNx

        197    10572671       72692             1
+6.00000000000000E+000   +1.04589000000000E+004   +9.00000000000000E-002
  +0.00000000000000E+000 N            O            07/12/1995 05/15/1995
  08/01/1995    NONE                      FOB        Q7k izOlRnQjmBgj
6ik2w Ni32Q
        197    14873533      623576             2
+2.70000000000000E+001   +4.06563300000000E+004   +4.00000000000000E-002
  +5.00000000000000E-002 R            F            05/14/1995 05/21/1995
  06/03/1995    DELIVER IN PERSON         AIR        Ow n2NCSyhSg Pj 1O
ng7h
        197    19465761      215819             3
+6.00000000000000E+000   +1.03547400000000E+004   +3.00000000000000E-002
  +5.00000000000000E-002 N            O            07/29/1995 05/12/1995
  08/15/1995    NONE                      FOB
SNhm1P5SiRnLAROO3x2B5hz P
        197     1042592      292594             4
+3.70000000000000E+001   +5.67779800000000E+004   +2.00000000000000E-002
  +7.00000000000000E-002 N            O            07/05/1995 05/14/1995
  07/08/1995    TAKE BACK RETURN          MAIL
yNSxl1A6CSyiBR3AAxB4xi7RlxBj2n25
        197     2843036      843037             5
+2.30000000000000E+001   +2.25144700000000E+004   +1.00000000000000E-002
  +4.00000000000000E-002 N            O            06/29/1995 07/06/1995
  07/14/1995    NONE                      FOB
2h1klPhSgPl0Qh6g6O457l0wk1l RgLiwRx7

  10 record(s) selected.
```

C.3 Query Substitution Parameters

```
Power stream    Seed = 304789933
-- TPC-D Parameter Substitution (Version 1.3.1)
-- using 304789933 as a seed to the RNG
Q1  DELTA         104
Q2  SIZE          37
    TYPE          STEEL
    REGION        ASIA
Q3  SEGMENT       HOUSEHOLD
    DATE          1995-03-14
Q4  DATE          1996-08-01
Q5  REGION        EUROPE
    DATE          1995-01-01
Q6  DATE          1996-01-01
    DISCOUNT      0.05
    QUANTITY      25
Q7  NATION1       CHINA
    NATION2       IRAN
Q8  NATION        CHINA
    REGION        ASIA
    TYPE          MEDIUM PLATED STEEL
Q9  COLOR         puff
Q10 DATE          1994-07-01
Q11 NATION        CHINA
    FRACTION      0.0000010000
Q12 SHIPMODE1     FOB
    SHIPMODE2     RAIL
    DATE          1995-01-01
Q13 CLERK         Clerk#000000731
Q14 DATE          1996-09-01
Q15 DATE          1996-08-01
Q16 BRAND         Brand#43
    TYPE          LARGE PLATED
    SIZE1         2
    SIZE2         7
    SIZE3         26
    SIZE4         13
    SIZE5         9
    SIZE6         22
    SIZE7         41
    SIZE8         33
Q17 BRAND         Brand#43
    CONTAINER     MED CAN
Throughput Stream = 1    Seed = 542987759
-- TPC-D Parameter Substitution (Version 1.3.1)
-- using 542987759 as a seed to the RNG
Q1  DELTA         114
Q2  SIZE          45
    TYPE          COPPER
    REGION        AMERICA
Q3  SEGMENT       MACHINERY
    DATE          1995-03-23
Q4  DATE          1997-06-01
Q5  REGION        MIDDLE EAST
    DATE          1996-01-01
Q6  DATE          1997-01-01
    DISCOUNT      0.07
    QUANTITY      24
Q7  NATION1       RUSSIA
    NATION2       CHINA
Q8  NATION        RUSSIA
    REGION        EUROPE
    TYPE          ECONOMY BURNISHED COPPER
Q9  COLOR         spring
Q10 DATE          1994-11-01
Q11 NATION        RUSSIA
    FRACTION      0.0000010000
Q12 SHIPMODE1     SHIP
    SHIPMODE2     FOB
    DATE          1994-01-01
Q13 CLERK         Clerk#000000897
Q14 DATE          1997-07-01
Q15 DATE          1997-06-01
Q16 BRAND         Brand#54
    TYPE          SMALL POLISHED
    SIZE1         6
    SIZE2         47
    SIZE3         32
    SIZE4         43
    SIZE5         5
    SIZE6         41
    SIZE7         21
    SIZE8         1
Q17 BRAND         Brand#54
    CONTAINER     LG JAR
Throughput Stream = 2    Seed = 300838752
-- TPC-D Parameter Substitution (Version 1.3.1)
-- using 300838752 as a seed to the RNG
Q1  DELTA         81
Q2  SIZE          18
    TYPE          COPPER
    REGION        AFRICA
Q3  SEGMENT       BUILDING
    DATE          1995-03-20
Q4  DATE          1994-10-01
Q5  REGION        AMERICA
    DATE          1996-01-01
Q6  DATE          1994-01-01
    DISCOUNT      0.07
    QUANTITY      24
Q7  NATION1       INDIA
    NATION2       MOROCCO
Q8  NATION        INDIA
    REGION        ASIA
    TYPE          LARGE POLISHED COPPER
Q9  COLOR         goldenrod
Q10 DATE          1993-10-01
Q11 NATION        INDIA
    FRACTION      0.0000010000
Q12 SHIPMODE1     RAIL
    SHIPMODE2     MAIL
    DATE          1993-01-01
Q13 CLERK         Clerk#000000354
Q14 DATE          1994-11-01
Q15 DATE          1994-10-01
Q16 BRAND         Brand#24
    TYPE          SMALL ANODIZED
    SIZE1         49
    SIZE2         16
    SIZE3         20
    SIZE4         8
    SIZE5         34
    SIZE6         30
    SIZE7         29
    SIZE8         5
Q17 BRAND         Brand#24
    CONTAINER     SM DRUM
Throughput Stream = 3    Seed = 1046734243
-- TPC-D Parameter Substitution (Version 1.3.1)
-- using 1046734243 as a seed to the RNG
```

```
Q1  DELTA        69
Q2  SIZE         8
    TYPE         NICKEL
    REGION       ASIA
Q3  SEGMENT      AUTOMOBILE
    DATE         1995-03-23
Q4  DATE         1993-10-01
Q5  REGION       AFRICA
    DATE         1996-01-01
Q6  DATE         1993-01-01
    DISCOUNT     0.07
    QUANTITY     25
Q7  NATION1      CANADA
    NATION2      CHINA
Q8  NATION       CANADA
    REGION       AMERICA
    TYPE         ECONOMY PLATED NICKEL
Q9  COLOR        chartreuse
Q10 DATE         1993-05-01
Q11 NATION       CANADA
    FRACTION     0.0000010000
Q12 SHIPMODE1    AIR
    SHIPMODE2    FOB
    DATE         1995-01-01
Q13 CLERK        Clerk#000000150
Q14 DATE         1993-10-01
Q15 DATE         1993-10-01
Q16 BRAND        Brand#14
    TYPE         LARGE ANODIZED
    SIZE1        7
    SIZE2        44
    SIZE3        39
    SIZE4        1
    SIZE5        27
    SIZE6        48
    SIZE7        5
    SIZE8        45
Q17 BRAND        Brand#14
    CONTAINER    MED PACK
Throughput Stream = 4    Seed = 731427844
-- TPC-D Parameter Substitution (Version 1.3.1)
-- using 731427844 as a seed to the RNG
Q1  DELTA        95
Q2  SIZE         29
    TYPE         NICKEL
    REGION       AMERICA
Q3  SEGMENT      FURNITURE
    DATE         1995-03-07
Q4  DATE         1995-11-01
Q5  REGION       ASIA
    DATE         1994-01-01
Q6  DATE         1995-01-01
    DISCOUNT     0.03
    QUANTITY     24
Q7  NATION1      KENYA
    NATION2      ETHIOPIA
Q8  NATION       KENYA
    REGION       AFRICA
    TYPE         SMALL BURNISHED NICKEL
Q9  COLOR        misty
Q10 DATE         1994-03-01
Q11 NATION       KENYA
    FRACTION     0.0000010000
Q12 SHIPMODE1    MAIL
    SHIPMODE2    AIR
    DATE         1994-01-01
Q13 CLERK        Clerk#000000580
Q14 DATE         1995-12-01
Q15 DATE         1995-11-01
Q16 BRAND        Brand#32
    TYPE         SMALL BURNISHED
    SIZE1        28
    SIZE2        35
    SIZE3        36
    SIZE4        8
    SIZE5        14
    SIZE6        23
    SIZE7        19
    SIZE8        50
Q17 BRAND        Brand#32
    CONTAINER    LG BOX
Throughput Stream = 5    Seed = 1768551525
-- TPC-D Parameter Substitution (Version 1.3.1)
-- using 1768551525 as a seed to the RNG
Q1  DELTA        114
Q2  SIZE         45
    TYPE         BRASS
    REGION       AMERICA
Q3  SEGMENT      MACHINERY
    DATE         1995-03-24
Q4  DATE         1997-05-01
Q5  REGION       MIDDLE EAST
    DATE         1996-01-01
Q6  DATE         1997-01-01
    DISCOUNT     0.07
    QUANTITY     24
Q7  NATION1      RUSSIA
    NATION2      CHINA
Q8  NATION       RUSSIA
    REGION       EUROPE
    TYPE         ECONOMY PLATED BRASS
Q9  COLOR        spring
Q10 DATE         1994-11-01
Q11 NATION       RUSSIA
    FRACTION     0.0000010000
Q12 SHIPMODE1    SHIP
    SHIPMODE2    FOB
    DATE         1994-01-01
Q13 CLERK        Clerk#000000894
Q14 DATE         1997-07-01
Q15 DATE         1997-05-01
Q16 BRAND        Brand#54
    TYPE         SMALL POLISHED
    SIZE1        24
    SIZE2        27
    SIZE3        50
    SIZE4        47
    SIZE5        5
    SIZE6        40
    SIZE7        10
    SIZE8        36
Q17 BRAND        Brand#54
    CONTAINER    LG JAR
```

Appendix D: Driver Source Code

D.1 tpcdbatch.sqc

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

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

#include <time.h>
#include <ctype.h>
#ifndef SQLWINT
#include <sys/time.h>                                   /*@d33143aha*/
#include <sys/ipc.h>
#include <sys/sem.h>
#include <sys/mode.h>
#include <sys/timeb.h>
#include <sys/types.h>
#else
#include <windows.h>
#include <sys\timeb.h>
#endif
#include <errno.h>


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

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


/*********************************************************************/
/* Define synonyms here                                              */
/*********************************************************************/
#define TPCDBATCH_VERSION "5.0"

#define TPCDBATCH_NONSQL      10
#define TPCDBATCH_SELECT      20
#define TPCDBATCH_NONSELECT   30
#define TPCDBATCH_EOBLOCK     40
#define TPCDBATCH_INSERT      50
#define TPCDBATCH_DELETE      60

#define TPCDBATCH_MAX_COLS    100

#define TPCDBATCH_CHAR char

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

/* #define TPCD_PREPARETIME 1 */      /* for separate prep/exec on uf */

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

#define PARALLEL_UPDATES 1

#ifdef PARALLEL_UPDATES
#define UF1OUTSTREAMPATTERN "%s%cuf1.%02d.%d.out"
#ifdef TPCD_NONPARTITIONED
#define UF2OUTSTREAMPATTERN "%s%cuf2.%02d.%d.out"
#else
#define UF2OUTSTREAMPATTERN "%s%cuf2.%02d.%d.%d.out"    /*add delchunk*/
#endif
#define BUFSIZE 1024
#endif


#define T_STAMP_FORM_1 1
#define T_STAMP_FORM_2 2
/* jen TIME_ACC start */
#define T_STAMP_FORM_3 3
#define T_STAMP_1LEN 17
#if defined (SQLUNIX) || defined (SQLAIX)
#define T_STAMP_3LEN 24
#elif (defined (SQLOS2) || defined(SQLWINT) || defined (SQLWIN) ||
defined(SQLDOS))
#define T_STAMP_3LEN 22
#else
#error Unknown operating system
#endif
/* TIME_ACC start */


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

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

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

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

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

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

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

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

/*********************************************************************/
/* 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                  sem_on;             /* semaphore stuff
  */
   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 */
   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;
   unsigned int        stmt_num;
   double              elapse_time;
   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                  */
   struct stmt_info    *next;
};


/*********************************************************************/
/* Structure containing command line options                         */
/*********************************************************************/
struct comm_line_opt
{

   char                 str_file_name[256];    /* output filename    */
```

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


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


/*****************************************************************/
/* Structure containing control flags passed between functions   */
/*****************************************************************/
struct ctrl_flags
{
   int eo_infile;
   int time_stamp;
   int eo_block;
   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);
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);
void dumpCa(struct sqlca*);

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( int updatePair, int copyOnOrOff );
void runUF2( int updatePair, int copyOnOrOff );

void runUF1_fn( int updatePair, int i, int copyOnOrOff );
void runUF2_fn( int updatePair, int i, int numChunks );

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);
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);


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 {                             /* LONG */
        short len;
        char data[32700];
       } stmt_str;                   /* LONG */
char    dbname[9] = "\0";
char    userid[9] = "\0";
char    passwd[9] = "\0";
char    sourcefile[256];             /* used for semaphores and table
functions?*/

EXEC SQL END DECLARE SECTION;


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

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

/* Other globals */
FILE               *instream, *outstream;  /* File pointers
*/
int                verbose = 0;            /* Verbose option flag
*/
int                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) */


/*****************************************************************/
/* Start main program processing.                                */
/*****************************************************************/
int main(int argc, char *argv[])
{
  struct comm_line_opt c_l_opt = { "\0","\0", 0, 1, 0, 0, 0 };
  /* command line options          */
  Timer_struct         start_time;          /* start point for elapsed
time */


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

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

  /* TIME_ACC start */
#if defined (SQLUNIX) || defined (SQLAIX)
  struct global_struct g_struct =
  { NULL, NULL, NULL, NULL, {0,0}, {0,0}, "\0", 0.1, "\0", 1, FALSE, 0,
    NULL, "\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", 1,
FALSE, 0,
    NULL, "\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);


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

    get_start_time(&start_time);
    strcpy(g_struct.s_info_ptr->start_stamp,
            get_time_stamp(T_STAMP_FORM_3,&start_time ));  /* TIME_ACC
*/

    /* 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 */
              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) == 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));

    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 */
#endif
   return;
}

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

/*********************************************************************/
/* Get environment variables.                                        */
/*********************************************************************/
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    */
   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, "--#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) {
 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,"\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);
            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);


            /* 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;
         test_semi = strstr (input_ln,";");
         if (test_semi == NULL) {   /* if there's no semi-colon keep on
going */
            strcat (stmt_str.data,input_ln);                  /* LONG */
            strcat (stmt_str.data," ");                       /* LONG */
            stmt_str.len = strlen( stmt_str.data );           /* LONG */
            eostmt = 0;
         }

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

   }  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);                         /* 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
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(long))) == NULL)
            mem_error("allocating INTEGER");
         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;

      if (*(sqlda->sqlvar[col].sqlind))
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],
                    *(long *)(sqlda->sqlvar[col].sqldata));
            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:
 ((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 (SQLUNIX) || defined (SQLAIX)                    /*TIMER*/
  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 (SQLUNIX) || defined (SQLAIX)
  status = gettimeofday(&end_time,NULL);                       /*TIMER*/
#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 next highest 0.1 seconds
as
    * required by the TPC-D spec.
    */
   result = (double)(((long)((result + 0.099999) * 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 */
  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   */
      if ((sqlca.sqlerrml)   /*  there's one or more tokens   */
          && (sqlca.sqlerrml < sizeof(sqlca.sqlerrmc)) /* and field not
full */
          )
      {
         char *tokptr;
         int  tok1;
         *(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 */
 /** Determine if the error is critical or a connection can be made **/

      EXEC SQL CONNECT ;
     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         -l  Input file containing list of statement
numbers");
   printf("\n         -r  Create set of output files containing query
results");
   printf("\n                Default - off");
   printf("\n         -v  Verbose.  Sends information to stderr
during");
   printf("\n             query processing");
   printf("\n                Default - off");
   printf("\n         -b  Process groups of statements as blocks ");
   printf("\n             instead of individually.");
   printf("\n                Default - off");
   printf("\n         -u  Update streams: p    - for power test");
   printf("\n                             t    - for throughput test
without");
   printf("\n                                    UFs (run this instead
of t2)");
   printf("\n                             t1   - for throughput test
step 1");
   printf("\n                                    only running queries");
   printf("\n                             t2   - for throughput test
step 2");
   printf("\n                                    running update
functions");
   printf("\n         -s  Scale factor");
   printf("\n                Default - 0.1");
   printf("\n         -n  Stream number");
   printf("\n                Default - 0");
   printf("\n         -m  Maximum number of keys to delete at a time");
   printf("\n                Default - 400");
   printf("\n         -h  Display this help screen");

   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' :
          case 'F' :

strcpy(g_struct->c_l_opt->infile,argv[++loopvar]);
                        break;

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

          case 'r' :
          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' :
          case 'D' :
                        strcpy(dbname,argv[++loopvar]);
                        break;

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

          case 'u' :
          case 'U' :
            g_struct->c_l_opt->update=-1; /* init to invalid number */
            if (!strcmp(uppercase(argv[++loopvar]),"P"))
               g_struct->c_l_opt->update=1;
            if (!strcmp(uppercase(argv[loopvar]),"T1"))
                  g_struct->c_l_opt->update=0;
            if (!strcmp(uppercase(argv[loopvar]),"T2"))
                  g_struct->c_l_opt->update=2;
            if (!strcmp(uppercase(argv[loopvar]),"T"))
                  g_struct->c_l_opt->update=5;
            break;

          case 'b' :
          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' :
          case 'N' :
            g_struct->c_l_opt->intStreamNum = atoi(argv[++loopvar]);
            break;

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

          case 'h':
          case 'H' :
           display_usage();
            break;

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

#ifdef PARALLEL_UPDATES
          case 'i':
             updatePair = atoi (argv[++loopvar]);
             break;

          case 'j':
             function = atoi (argv[++loopvar]);
             break;

          case 'k':
             updateStream = atoi (argv [++loopvar]);
             break;

          case 'x':                                 /*DEL -x is chunk*/
             deleteChunk = atoi (argv[++loopvar]);/* to delete for this*/
             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, copyOnOrOff);
          else
             if (function == 2)
             runUF2_fn (updatePair, updateStream, deleteChunk);
          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);

    }

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

}



/************************************************/
/* Converts DECIMAL values to ASCII text        */
/************************************************/
int sqlrxd2a(
                                                  /* 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;
   int tmpint, src_nibble;
   int count, j, limit[3];

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


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


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

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

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

   }

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


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

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

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

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

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

/* determine if we are running from a single node (or a serial setup) */
/* and therefore want to use semaphores to control the processing of  */
/* the update functions, or if we are going to be running on multiple */
/* nodes for the throughput test and want to use the control in the   */
   /* calling script */

   if (!strcmp(uppercase(env_tpcd_run_on_multiple_nodes),"YES")) {
     /* okay, we don't want the semaphores */
     g_struct->sem_on = 0;
   } else {
     g_struct->sem_on = 1;  /* use the semaphores */
   }

/* 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 */
      }
       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
* 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 */
      /* 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;
   }
/*********************************************************************/
/* Connect to the target database.  Start DBM if not already done so.*/
/*********************************************************************/
   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 */
   strcpy(g_struct->file_time_stamp,

get_time_stamp(T_STAMP_FORM_2,&(g_struct->stream_start_time)));
/*TIME_ACC*/

   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*/
   if ( g_struct->c_l_opt->intStreamNum > 0 )
   {
      if (g_struct->c_l_opt->update == 2 ||
          g_struct->c_l_opt->update == 5)
      {
         /* update function stream */
         sprintf(file_name,"%s%sstrcntuf.%s",g_struct->run_dir,
                   env_tpcd_path_delim, g_struct->file_time_stamp);
      }
      else
      {
         /* 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);
      }
      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 == 2 ||
          g_struct->c_l_opt->update == 5)
      {
        /* 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*/
      }
      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 */

get_time_stamp(T_STAMP_FORM_3,&(g_struct->stream_start_time))); /*
TIME_ACC */
      }
   }

/* 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 (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
!= 5 ))
        /* on or onl, but not  */
   {
      updateFP = fopen(g_struct->update_num_file,"r");
```

```
      if (updateFP != NULL )
      {
         fscanf(updateFP,"%d",&updatePairStart);
         fclose(updateFP);
         if ( g_struct->c_l_opt->update == 1 )   /* 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:
         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_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:
         /* 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 ((m+n), col_name_length);
         /* Special handling for DECIMAL */
         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 */
   struct tm *tp;
   size_t timeLength = 0;

   /* TIME_ACC 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 stop*/

   if ((form == T_STAMP_FORM_1) || (form == T_STAMP_FORM_3))
   {
      timeLength = strftime(newtime,50,"%x %X",tp);
      /* TIME_ACC 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,".%0.4d",temp_stamp.millitm);
#else
#error Unknown operating system
#endif
         /* TIME_ACC 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 Ts   = 0;                    /* different TPC-D metrics */
   double QppD = 0;
   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  */
   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 */
   get_time_stamp(T_STAMP_FORM_3,&(g_struct->stream_start_time))); /*
TIME_ACC */
   /* 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*/
   get_time_stamp(T_STAMP_FORM_3,&(g_struct->stream_end_time)));  /*
TIME_ACC */
   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)
               if (s_info_ptr->elapse_time < (max->elapse_time/1000))
                  s_info_ptr->adjusted_time = max->elapse_time/1000;
         }                              /* 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 */
                  T_STAMP_1LEN,T_STAMP_1LEN,s_info_ptr->end_stamp); /*
TIME_ACC */
     /* 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;
   }
   /* Calculate the arithmetic and geometric means */
   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 = 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 */
   if (g_struct->c_l_opt->update != 2)
   {
      fprintf (outstream, "\nArith. mean %15.3f %15.3f\n",\
               arith_mean,adjusted_a_mean);
      fprintf (outstream, "Geom.  mean %15.3f %15.3f\n",\
               geo_mean,adjusted_g_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*/
               T_STAMP_1LEN,T_STAMP_1LEN,max->end_stamp);  /*TIME_ACC*/
      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*/
               T_STAMP_1LEN,T_STAMP_1LEN,min->end_stamp);  /*TIME_ACC*/
   }
   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 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;
#elif (defined (SQLOS2) || defined(SQLWINT) || defined (SQLWIN) ||
defined(SQLDOS))
      Ts = g_struct->stream_end_time.time -
g_struct->stream_start_time.time + 1;
#else
#error Unknown operating system
#endif
      /* TIME_ACC stop*/
      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 */
      if (db_size_frac_part < .01)
            fprintf (outstream,
                  "QppD@%-.0f%-5.5s = %10.3f\n\nTs %11 =
%10.0f\n\nQthD@%-.0f%-5.5s = %10.3f\n\nQphD@%-.0f%-5.5s = %10.3f\n",
                     db_size,db_size_qualifier,QppD,
Ts,db_size,db_size_qualifier,
                     QthD,db_size,db_size_qualifier, QphD);
      else
            fprintf (outstream,
                  "QppD@%-.2f%-5.5s = %10.3f\n\nTs %11 =
%10.0f\n\nQthD@%-.2f%-5.5s = %10.3f\n\nQphD@%-.2f%-5.5s = %10.3f\n",
                     db_size,db_size_qualifier,QppD,
Ts,db_size,db_size_qualifier,
                     QthD,db_size,db_size_qualifier, QphD);
   }
}
/************************************************************/
/* free up all the elements of the sqlda after done processing */
/************************************************************/
void free_sqlda (struct sqlda *sqlda, int select_status)
{
   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 */
}
/* ************ PARALLEL_UPDATES version of runUF1 and runUF2 only */

/*********************************************/
/* processing to run the insert update function */
/*********************************************/
void runUF1 ( int updatePair, int copyOnOrOff )
{
#ifndef TPCD_NONPARTITIONED
   char statement[3000];
   char sourcedir[256];
#endif

   int split_updates = 2;     /* no. of ways update records are split */
   int concurrent_inserts = 2;/* no of concurrent updates to be */
                              /* run at once*/
   int loop_updates = 1;      /* no of updates to be run in one  */
                              /* "concurrent" invocation.  should*/
                              /* be split_updates/concurrent_inserts*/
   int i, j;
   char myoutstreamfile[256];
   FILE *myoutstream;
   char *buffer;
#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
   fprintf( outstream,"UF1 for update pair %d starting\n",updatePair);
   fprintf (stderr, "UF1 for update pair %d starting at
%*.*\n",updatePair,
            T_STAMP_1LEN,T_STAMP_1LEN,
            get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));
#ifndef TPCD_NONPARTITIONED
   /* for PE we want to call load_uf1 script that will do the
      appropriate load across the nodes. For SE, the load and insert are
      all done by the runUF1_fn function */
   /* first must determine the directory to find the script in */
   if (getenv("TPCD_PELOAD_DIR") != NULL)
      strcpy(sourcedir,getenv("TPCD_PELOAD_DIR"));
   else
      strcpy(sourcedir,".");
   sprintf(statement, "%s/load_uf1 %d 1",sourcedir,updatePair);
   if (system(statement))
   {
      fprintf (outstream,
            "load_uf1 failed for UF1, examine UF1.log for cause.
Exiting.\n");
      if (verbose)
         fprintf (stderr,
               "load_uf1 failed for UF1, examine UF1.log for cause.
Exiting.\n");
      exit (-1);
   }
   fprintf (outstream, "load_uf1 finished for UF1.\n");
   fprintf (stderr, "load_uf1 finished for UF1. at %*.*\n",
```

```
            T_STAMP_1LEN,T_STAMP_1LEN,
            get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));

#endif  /* not TPCD_NONPARTITIONED */

   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 first flat file to generate the semaphore key
*/
   if (getenv("TPCD_FLATFILES") != NULL)
   {
#ifndef TPCD_NONPARTITIONED
      /* this is assuming that you will be running this from 0th node */
      sprintf(sourcefile, "%s%corder.tbl.u%d.00000.0",
              getenv("TPCD_FLATFILES"), PATH_DELIM,updatePair);
#else
      sprintf(sourcefile, "%s%corder.tbl.u%d.0",
              getenv("TPCD_FLATFILES"), PATH_DELIM, updatePair);
#endif
   }
   else
   {
      fprintf (stderr, "TPCD_FLATFILES is not defined.\n");
      exit (-1);
   }
   su_semkey = ftok (sourcefile, 'J');
   if ( (su_semid = semget (su_semkey, 1, IPC_CREAT|S_IRUSR|S_IWUSR)) <
0)
   {
      fprintf (stderr, "Can't get semaphore! semget failed: errno =
%d\n",
               errno);
      exit (-1);
   }
#else  /* SQLWINT */
   sprintf (UF1_semfile, "%s.%s.UF1.semfile", env_tpcd_dbname,
env_user);
   fprintf(stderr,"UF1 semfile = %s\n",UF1_semfile);
   su_hSem = CreateSemaphore(NULL, 0,
                             concurrent_inserts,
                            (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*/
   for (i=0; i < concurrent_inserts; i++)
   {
#ifndef SQLWINT
      if ((childpid[i] = fork()) == 0)
      {
         runUF1_fn (updatePair, i, copyOnOrOff);
      }
      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 );
      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)
   {
      fprintf(stderr,
              "Failure to wait on insert semaphone with %d of
children\n",
              concurrent_inserts);
      exit(1);
   }
  semctl (su_semid, 0, IPC_RMID, 0);
#else
   for (i = 0; i < concurrent_inserts; i++)
  {
      if (verbose)
      {
         fprintf(stderr,"About to wait again ...Sets to wait for %d\n",
                 concurrent_inserts - i);
     }
      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);
   }
   /* fixing multiple header problem
      wlc 081397 */

   /* combine the output files from the different streams */
   if ((buffer = malloc (BUFSIZE)) == NULL)
   {
      exit (-1);
   }
   for (i=0; i < concurrent_inserts; i++)
  {
      sprintf (myoutstreamfile, UF1OUTSTREAMPATTERN, env_tpcd_tmp_dir,
PATH_DELIM,
               updatePair, i);
      if ( (myoutstream = fopen (myoutstreamfile, READMODE)) == NULL) {
         fprintf (stderr,
                  "\nRunUF1: The output file '%s' for update pair %d set
%d could not be opened. runUF1\n",
                  myoutstreamfile,updatePair,i);
         fprintf (outstream,
                  "\nRunUF1: The output file '%s' for update pair %d set
%d could not be opened. runUF1\n",
                  myoutstreamfile,updatePair,i);
         exit (-1);
      }
 /* copy the while output file for stream i into the real output file */
      do
      {
         j=fread (buffer, sizeof (char), BUFSIZE, myoutstream);
         fwrite (buffer, sizeof (char), j, outstream);
      } while (!feof (myoutstream));
      fclose (myoutstream);
   }
   fprintf( outstream,"UF1 for update pair %d complete\n",updatePair);
}
void runUF1_fn ( int updatePair, int i, int copyOnOrOff )
{
   char sourcefile[256];
   int                  rc = 0;
   sqlu_media_list      dataFileList;
   sqlu_media_list      *pLobPathList;
   struct sqldcol       dcoldata;
   struct sqlchar       *pActionString;
   char                 filetype[] = SQL_DEL;
   struct sqlchar       *pFileTypeMod;
   int split_updates = 2;     /* no. of ways update records are split */
   int concurrent_inserts = 2;/* no of concurrent updates to be */
                              /* run at once*/
   int loop_updates = 1;      /* no of updates to be run in one  */
                              /* "concurrent" invocation.  should*/
                              /* be split_updates / concurrent_inserts*/
   int startChunk = 0;        /* number of first chunk to insert for */
                              /* this child */
   int stopChunk = 0;         /* number of last chunk to insert for */
                              /* this child */
   int chunk = 0;             /* counter for within the set of chunks*/
   long insertedLineitem = 0;
   long insertedOrders = 0;
   long savedInsertedOrders = 0;
   long sqlcode;
   int maxwait;
   char statement[3000];   /* jen temp */
#ifndef SQLWINT
   int              su_semid;/*semaphore for controlling split updates*/
   key_t            su_semkey;          /* key to generate semid */
#else
   HANDLE               su_hSem;
   char                 UF1_semfile[256];
#endif
   char myoutstreamfile[256];
   FILE *myoutstream;
#ifdef TPCD_NONPARTITIONED
   char                 localMsgFile1[256];
   char                 localMsgFile2[256];
   char                 remoteMsgFile1[256];
   char                 remoteMsgFile2[256];
   short                callerAction = SQLU_INITIAL;
   struct sqluload_in   LoadInfoIn;
   struct sqluload_out  LoadInfoOut;
   sqlu_media_list      *pDirectory = NULL; /* use default sqllib/tmp*/
   sqlu_media_list      CopyTarget;
   long                 *pNullInd = NULL;   /* only required for ASC */
   void                 *pReserved = NULL;
   struct sqluimpt_in   ImportInfoIn;           /* IMPORT */
   struct sqluimpt_out  ImportInfoOut;          /* IMPORT */
#endif
#ifdef SQLWINT
/*must reread the environment variable since they are not inherited by*/
/* children */
   if (getenv("TPCD_UPDATE_IMPORT") != NULL)
   {
      if (!strcmp(uppercase(getenv("TPCD_UPDATE_IMPORT")),"TRUE"))
      {
         iImportStagingTbl = 1;   /* use import */
      }
   }
#endif
    if (getenv ("TPCD_SPLIT_UPDATES") != NULL)
      split_updates = atoi (getenv ("TPCD_SPLIT_UPDATES"));
   if (getenv ("TPCD_CONCURRENT_INSERTS") != NULL)
      concurrent_inserts = atoi (getenv ("TPCD_CONCURRENT_INSERTS"));
   loop_updates = split_updates / concurrent_inserts;
/* determine the starting and stopping point of the chunks that this  */
/* invocation will apply. i is starting chunk number with range 0  */
/* through (concurrent_inserts -1)                                    */
   startChunk = i * loop_updates;
   stopChunk = startChunk + (loop_updates - 1);
   sprintf (myoutstreamfile, UF1OUTSTREAMPATTERN, env_tpcd_tmp_dir,
PATH_DELIM,
```

```
            updatePair, i);
   if ( (myoutstream = fopen (myoutstreamfile, WRITEMODE)) == NULL)
   {
      fprintf (stderr, "\nThe output file '%s' for update pair %d set %d
could not be opened. runUF1_fn\n",
               myoutstreamfile,updatePair,i);
      rc=-1;
      goto UF1_exit;
   }
  fprintf( myoutstream,"\nUF1 for update pair %d set %d starting at
%*.*s\n",
            updatePair, i,
            T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
            get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));    /*
TIME_ACC jen*/
   if (!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,"\nStart the DB2 Database Manager first\n");
      exit (-1);
   }
   error_check();

#ifdef TPCD_NONPARTITIONED
   fprintf( myoutstream,
            "Staging table load for update pair %d set %d starting at
%*.*s\n",
            updatePair, i,
             T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
            get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));   /*
TIME_ACC jen*/

  /* we must loop around for the sequential chunks that we are */
  /* applying to load all data chunks                          */
   for ( chunk = startChunk; chunk <= stopChunk; chunk++ )
   {
  /* substitions done through this loop of code from i to chunk */
  /* no s&d codes for this change...to messy to follow */
  /* for version 2 we want to get the name of the directory where the */
  /* data files reside (TPCD_FLATFILES) and use them as input to the  */
  /* load api.  The load api will be used to load the data from the   */
  /* flat files into the temporary tables */
      if (getenv("TPCD_FLATFILES") != NULL)
         sprintf(sourcefile, "%s%corder.tbl.u%d.%d",
                 getenv("TPCD_FLATFILES"),
                 PATH_DELIM, updatePair, chunk);

      sprintf (localMsgFile1, "%s%ctpcdloadorder.msg.%d",
               env_tpcd_tmp_dir, PATH_DELIM, chunk);
      sprintf (localMsgFile2, "%s%ctpcdloadline.msg.%d",
               env_tpcd_tmp_dir, PATH_DELIM, chunk);
      sprintf (remoteMsgFile1, "%s%ctpcdloadorder.rmt.%d",
               env_tpcd_tmp_dir, PATH_DELIM, chunk);
      sprintf (remoteMsgFile2, "%s%ctpcdloadline.rmt.%d",
               env_tpcd_tmp_dir, PATH_DELIM, chunk);
      /* IMPORT (begin) moved common code up */
      /* method - use default ... other fields ignored */
      dcoldata.dcolmeth = SQL_METH_D;
      /* action string  (what do to with loaded data) */
      pActionString = (struct sqlchar *)malloc(256);
      sprintf(pActionString->data,"REPLACE INTO TPCDTEMP.ORDERS_%d",
chunk); /*REUSE_STAGE */
      pActionString->length = strlen(pActionString->data);
      /* file modifcation */
      pFileTypeMod = (struct sqlchar *)malloc(256);
      strcpy(pFileTypeMod->data, "COLDEL|");
      pFileTypeMod->length = strlen(pFileTypeMod->data);
      /* IMPORT end  moved common code up */
      if ( iImportStagingTbl == 0 )       /* IMPORT isolate load /
import code*/
      {
         /* set up the datafile structure */
         dataFileList.media_type = SQLU_SERVER_LOCATION;
         dataFileList.sessions   = 1;
         dataFileList.target.location =
            (sqlu_location_entry *)
malloc(sizeof(sqlu_location_entry)*dataFileList.sessions);
strcpy(dataFileList.target.location->location_entry,sourcefile);
         pLobPathList = NULL;
         /* sqluload input structure */
         LoadInfoIn.sizeOfStruct = SQLULOAD_IN_SIZE;
         LoadInfoIn.savecnt = 0;   /*default*/
         LoadInfoIn.restartcnt = 0;
         LoadInfoIn.rowcnt = 0;    /* load all rows */
         LoadInfoIn.warningcnt = 0;  /* don't stop for warnings? */
         LoadInfoIn.data_buffer_size = 500;
         LoadInfoIn.sort_buffer_size = 4;  /* use default size */
         LoadInfoIn.hold_quiesce = 0;      /* don't hold the quiesce */
         LoadInfoIn.restartphase = ' ';    /* ignored?? */
         LoadInfoIn.statsopt = ' ';        /* no stats  */
         /* sqluload output structure */
         LoadInfoOut.sizeOfStruct = SQLULOAD_OUT_SIZE;
         /* copy target list */
         CopyTarget.media_type = SQLU_LOCAL_MEDIA;
         CopyTarget.sessions = 1;
         CopyTarget.target.media =
            (sqlu_media_entry *)malloc(sizeof (sqlu_media_entry) *
CopyTarget.sessions);

sprintf(CopyTarget.target.media->media_entry,"%s",getenv("TPCD_COPY_DIR"
));
         if ( copyOnOrOff == TRUE )
         {
        /* yes, we want to specifiy a copy target, the log retain is on*/
            rc = sqluload(&dataFileList,
                          pLobPathList,
                          &dcoldata,
                          pActionString,
                          filetype,
                          pFileTypeMod,
                          localMsgFile1,
                          remoteMsgFile1,
                          callerAction,
                          &LoadInfoIn,
                          &LoadInfoOut,
                          pDirectory,
                          &CopyTarget,
                          pNullInd,
                          pReserved,
                          &sqlca );
         }
         else
         {
            /* don't specify a copy target, log retain is off */
            rc = sqluload(&dataFileList,
                          pLobPathList,
                          &dcoldata,
                          pActionString,
                          filetype,
                          pFileTypeMod,
                          localMsgFile1,
                          remoteMsgFile1,
                          callerAction,
                          &LoadInfoIn,
                          &LoadInfoOut,
                          pDirectory,
                          NULL,
                          pNullInd,
                          pReserved,
                          &sqlca );
         }
         if ( error_check() < 0 )
         {
            fprintf(myoutstream,
               "\nA serious error occurred loading into
TPCDTEMP.ORDERS_%d\n",  /*REUSE_STAGE*/
                    chunk);
        /*REUSE_STAGE*/
            rc=-1;
            goto UF1_exit;
         }
         /* report how many rows read from the flat file to the table */
         fprintf(myoutstream,
            "%u rows read from %s\nand loaded into TPCDTEMP.ORDERS_%d at
%*.*s\n", /*REUSE_STAGE*/

LoadInfoOut.rowsLoaded,dataFileList.target.location->location_entry,
            chunk,T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
            get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));
  /* TIME_ACC*/                /*REUSE_STAGE*/
      }   /* IMPORT end of load specific code */
      /* IMPORT begin */
      else
      {
         ImportInfoIn.sizeOfStruct   = SQLUIMPT_IN_SIZE;
         ImportInfoIn.commitcnt      = 0;
         ImportInfoIn.restartcnt     = 0;
         ImportInfoOut.sizeOfStruct  = SQLUIMPT_OUT_SIZE;
         rc = sqluimpr(sourcefile,
                       NULL,
                       &dcoldata,
                       pActionString,
                       filetype,
                       pFileTypeMod,
                       localMsgFile1,
                       callerAction,
                       &ImportInfoIn,
                       &ImportInfoOut,
                       pNullInd,
                       pReserved,
                       &sqlca);
         if ( error_check() < 0 )
         {
            fprintf(myoutstream,
               "\nA serious error occurred importing into
TPCDTEMP.ORDERS_%d\n",   /*REUSE_STAGE*/
                    chunk);
          /*REUSE_STAGE*/
            rc=-1;
            goto UF1_exit;
         }
         /* report how many rows read from the flat file to the table */
         fprintf(myoutstream,
              "%u rows read from %s\nand imported into
TPCDTEMP.ORDERS_%d at %*.*s\n",  /*REUSE_STAGE*/
               ImportInfoOut.rowsInserted, sourcefile,
               chunk, T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
               get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));
/* TIME_ACC*/                /*REUSE_STAGE*/
      } /* IMPORT end */

      /* now load the lineitem temporary table */
      if (getenv("TPCD_FLATFILES") != NULL)
         sprintf(sourcefile, "%s%slineitem.tbl.u%d.%d",
                 getenv("TPCD_FLATFILES"),
                 getenv("TPCD_PATH_DELIM"),
                 updatePair, chunk);
      else
         sprintf(sourcefile,".%slineitem.tbl.u%d.%d",
                 getenv("TPCD_PATH_DELIM"),updatePair, chunk);


      /* action string  (what do to with loaded data) */
      sprintf(pActionString->data,"REPLACE INTO TPCDTEMP.LINEITEM_%d",
/*REUSE_STAGE*/
                 chunk);
/*REUSE_STAGE*/
      pActionString->length = strlen(pActionString->data);

      if ( iImportStagingTbl == 0 )  /*IMPORT isolate load/import code*/
      {
         /* set up the datafile structure */
      strcpy(dataFileList.target.location->location_entry,sourcefile);
         if ( copyOnOrOff == TRUE )
         {
      /* yes, we want to specifiy a copy target, the log retain is on*/
            rc = sqluload(&dataFileList,
                          pLobPathList,
                          &dcoldata,
                          pActionString,
```

```
                        filetype,
                        pFileTypeMod,
                        localMsgFile2,
                        remoteMsgFile2,
                        callerAction,
                        &LoadInfoIn,
                        &LoadInfoOut,
                        pDirectory,
                        &CopyTarget,
                        pNullInd,
                        pReserved,
                        &sqlca );
         }
         else
         {
            /* no, don't specify a copy target */
            rc = sqluload(&dataFileList,
                        pLobPathList,
                        &dcoldata,
                        pActionString,
                        filetype,
                        pFileTypeMod,
                        localMsgFile2,
                        remoteMsgFile2,
                        callerAction,
                        &LoadInfoIn,
                        &LoadInfoOut,
                        pDirectory,
                        NULL,
                        pNullInd,
                        pReserved,
                        &sqlca );
         }
         if ( error_check() < 0 )
         {
            fprintf(myoutstream,
                    "\nA serious error occurred loading into
TPCDTEMP.LINEITEM_%d\n",  /*REUSE_STAGE*/
                    chunk);
       /*REUSE_STAGE*/
            rc=-1;
            goto UF1_exit;
         }
         /* report how many rows read from the flat file to the table */
         fprintf(myoutstream,
            "%u rows read from %s\nand loaded into TPCDTEMP.LINEITEM_%d
at %*.*s\n",  /*REUSE_STAGE*/
LoadInfoOut.rowsLoaded,dataFileList.target.location->location_entry,
            chunk, T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
            get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));
/* TIME_ACC jen*/                       /*REUSE_STAGE*/
      }
      /* IMPORT begin */
      else
      {
         rc = sqluimpr(sourcefile,
                        NULL,
                        &dcoldata,
                        pActionString,
                        filetype,
                        pFileTypeMod,
                        localMsgFile2,
                        callerAction,
                        &ImportInfoIn,
                        &ImportInfoOut,
                        pNullInd,
                        pReserved,
                        &sqlca);
         if ( error_check() < 0 )
         {
            fprintf(stderr,
              "\nA serious error occurred importing into
TPCDTEMP.LINEITEM_%d\n",  /*REUSE_STAGE*/
               i);
          /*REUSE_STAGE*/
            rc=-1;
            goto UF1_exit;
         }
         /* report how many rows read from the flat file to the table */
         fprintf(myoutstream,
           "%u rows read from %s\nand imported into TPCDTEMP.LINEITEM_%d
at %*.*s\n", /*REUSE_STAGE*/
           ImportInfoOut.rowsInserted, sourcefile,
           i, T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
           get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));
/* TIME_ACC jen*/                       /*REUSE_STAGE*/
      }  /* IMPORT end */

   } /* jenCI end loop processing for all chunks applied by this
invocation */
   fprintf( myoutstream,
          "Staging table load for update pair %d set %d completed at
%*.*s\n\n",
            updatePair, i, T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC
jen*/
            get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));  /*
TIME_ACC jen*/
#if 0
   /* TEMPORARY WORK AROUND FOR UNI WHILE LOAD BUG PROBLEM */
           sprintf(statement,"db2 connect to tpcd; db2 \"select
count(*) from tpcd.supplier\"; db2 connect reset; db2 terminate");
/*jen temp*/
           system(statement);  /*jen temp*/
#endif
#endif /* TPCD_NONPARTITIONED */

   for ( chunk = startChunk; chunk <= stopChunk; chunk++ )
/*jenCI*/
   {
/*jenCI*/
      /* wlc  062797 */
      sqlcode = SQL_RC_E911;
      maxwait = 1;

      /* loop to handle any deadlocks */
      while (sqlcode == SQL_RC_E911 && maxwait <= MAXWAIT && rc==0)
      {
         sqlcode = 0;
         sprintf(stmt_str.data,
                 "INSERT INTO TPCD.ORDERS SELECT
O_ORDERKEY,O_CUSTKEY,O_ORDERSTATUS,O_TOTALPRICE,O_ORDERDATE,O_ORDERPRIOR
ITY,O_CLERK,O_SHIPPRIORITY,O_COMMENT FROM TPCDTEMP.ORDERS_%d",
/*REUSE_STAGE*/
                 chunk);
         stmt_str.len = strlen( stmt_str.data );              /*LONG */
#ifdef TPCD_PREPARETIME
         /* temp code to test timing START*/
         fprintf( myoutstream,
                  "\nUF1 orders update pair %d chunk %d prep at
%*.*s\n",
                  updatePair,chunk,T_STAMP_1LEN,T_STAMP_1LEN, /*TIME_ACC
*/
                  get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));
/* TIME_ACC*/
         EXEC SQL PREPARE INSO FROM :stmt_str;
         fprintf( myoutstream,
                  "\nUF1 orders pair %d chunk %d prep finished at
%*.*s\n",
                  updatePair,chunk,T_STAMP_1LEN,T_STAMP_1LEN,
/*TIME_ACC*/
                  get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));
/* TIME_ACC*/
         if (sqlca.sqlcode < 0)
            sqlcode = error_check();/*Don't bother printing the SQL0100W
                                      error from going through the loop
                                      above one extra time */
         EXEC SQL EXECUTE INSO;
         fprintf( myoutstream,
                  "\nUF1 orders pair %d chunk %d execute INSO at
%*.*s\n",
                  updatePair,chunk,T_STAMP_1LEN,T_STAMP_1LEN,  /*
TIME_ACC*/
                  get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));
/* TIME_ACC*/
         /* temp code to test timing end*/
#else
         EXEC SQL EXECUTE IMMEDIATE :stmt_str;
#endif
         if (sqlca.sqlcode < 0)
            sqlcode = error_check();

         if (sqlcode == SQL_RC_E911)
         {          /* we've hit a deadlock */
            fprintf (myoutstream,
                     "\nDeadlock detected inserting from
tpcdtemp.orders_new for chunk %d for pair
%d..Retrying...\n",chunk,updatePair);
            SleepSome(UF_DEADLOCK_SLEEP);
            maxwait++;                                  /* jen DEADLOCK */
         }
         else if (sqlcode < 0)
         {
            fprintf(myoutstream,
                    "Insert into orders pair %d chunk %d failed
sqlcode=%d\n",
                    updatePair,chunk,sqlcode);
            rc = -1;
         }
         else
         {
            /* Everything worked with ORDERS, proceed with LINEITEM */
            /* report the number of row inserted */
            savedInsertedOrders = sqlca.sqlerrd[2];

            sqlcode = 0;

            sprintf(stmt_str.data,
                    "INSERT INTO TPCD.LINEITEM SELECT
L_ORDERKEY,L_PARTKEY,L_SUPPKEY,L_LINENUMBER,L_QUANTITY,L_EXTENDEDPRICE,L
_DISCOUNT,L_TAX,L_RETURNFLAG,L_LINESTATUS,L_SHIPDATE,L_COMMITDATE,L_RECE
IPTDATE,\
L_SHIPINSTRUCT,L_SHIPMODE,L_COMMENT FROM TPCDTEMP.LINEITEM_%d",
/*REUSE_STAGE*/
                    chunk);
/*REUSE_STAGE*/
            stmt_str.len = strlen( stmt_str.data );           /* LONG */
            sqlcode = 0;
#ifdef TPCD_PREPARETIME
            /* temp code to test timing START*/
            fprintf( myoutstream,
                     "\nUF1 lineitem update pair %d chunk %d prep
starting at %*.*s\n",
                     updatePair,chunk,T_STAMP_1LEN,T_STAMP_1LEN,  /*
TIME_ACC*/
                     get_time_stamp(T_STAMP_FORM_1,(Timer_struct
*)NULL));  /* TIME_ACC*/
            EXEC SQL PREPARE INSL FROM :stmt_str;
            fprintf( myoutstream,
                     "\nUF1 lineitem update pair %d chunk %d prep
starting at %*.*s\n",
                     updatePair,chunk,T_STAMP_1LEN,T_STAMP_1LEN,  /*
TIME_ACC*/
                     get_time_stamp(T_STAMP_FORM_1,(Timer_struct
*)NULL));  /* TIME_ACC*/
            if (sqlca.sqlcode < 0)
             sqlcode = error_check();/*Don't bother printing theSQL0100W
                                       error from going through the loop
                                       above one extra time */
            EXEC SQL EXECUTE INSL;
            fprintf( myoutstream,
                     "\nUF1 lineitem update pair %d chunk %d prep
starting at %*.*s\n",
                     updatePair,chunk,T_STAMP_1LEN,T_STAMP_1LEN,  /*
TIME_ACC*/
                     get_time_stamp(T_STAMP_FORM_1,(Timer_struct
*)NULL));  /* TIME_ACC*/
            /* temp code to test timing STOP */
#else
            EXEC SQL EXECUTE IMMEDIATE :stmt_str;
#endif
            if (sqlca.sqlcode < 0)
               sqlcode = error_check();/*Don't bother printing
theSQL0100W
                                         error from going through the loop
```

```
                                                above one extra time */
            if (sqlcode == SQL_RC_E911)
            {          /* we've hit a deadlock */
               fprintf (myoutstream,
                        "\nA deadlock has been detected inserting from
tpcdtemp.lineitem_%d...Retrying...\n", /*REUSE_STAGE*/
                        chunk);  /*REUSE_STAGE*/
               SleepSome (UF_DEADLOCK_SLEEP);
               maxwait++;                                  /* DEADLOCK */
            }
            else if (sqlcode < 0)
            {
               fprintf (myoutstream,
                        "\nAn error occurred inserting into
TPCD.LINEITEM\n");
               fprintf (myoutstream,
                        "for update pair number %d chunk %d
...Exiting\n",
                        updatePair, chunk);
            }
            else
            {
#ifdef UF1DEBUG
               fprintf (myoutstream, "lineitem insert succeeded\n");
               fflush(myoutstream);
#endif
               /* accumulate the number of rows inserted */
               /* Order count ONLY updated if both orders and lineitem
*/
               /* go through */
               insertedOrders += savedInsertedOrders;   /* kbs */
               insertedLineitem += sqlca.sqlerrd[2];

               rc=0;
               EXEC SQL COMMIT WORK;
               error_check();
            }
         } /* process lineitem INSERTs */
      } /* while loop for deadlocks */
   } /* while processing chunks */


   fprintf(myoutstream,
           "%ld rows inserted into TPCD.ORDERS at %*.*s\n",
           insertedOrders,
           T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
           get_time_stamp(T_STAMP_FORM_1,
                          (Timer_struct *)NULL));  /* TIME_ACC jen*/
   fprintf(myoutstream,
           "%ld rows inserted into TPCD.LINEITEM at %*.*s\n",
           insertedLineitem,
           T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
           get_time_stamp(T_STAMP_FORM_1,
                          (Timer_struct *)NULL));  /* TIME_ACC jen*/

   fprintf( myoutstream,
            "UF1 for update pair %d chunks %d through %d complete at
%*.*s\n\n",
            updatePair, startChunk,stopChunk,
            T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
            get_time_stamp(T_STAMP_FORM_1,
                           (Timer_struct *)NULL));  /* TIME_ACC jen*/

   if (sqlcode < 0)
   {
      if (sqlcode == SQL_RC_E911)
      {
         fprintf (myoutstream,"# of deadlocks exceeds %i\n", MAXWAIT);
      }
      rc=-1;
      EXEC SQL ROLLBACK WORK;
      error_check();                    /* @d22275 tjg */

      goto UF1_exit;
   }

UF1_conn_reset:
   EXEC SQL CONNECT RESET;
   error_check();                    /* @d22275 tjg */

UF1_exit:
   fclose (myoutstream);
   /* exiting, increment the semaphore */

   /* we used the first flat file to generate the semaphore key */
#ifndef SQLWINT
   /*    we will use the first flat file to generate the semaphore key
*/
   if (getenv("TPCD_FLATFILES") != NULL) {
#ifndef TPCD_NONPARTITIONED
      /* this is assuming that you will be running this from 0th node */
      sprintf(sourcefile, "%s%corder.tbl.u%d.00000.0",
              getenv("TPCD_FLATFILES"), PATH_DELIM,updatePair);
#else
      sprintf(sourcefile, "%s%corder.tbl.u%d.0",
              getenv("TPCD_FLATFILES"), PATH_DELIM,updatePair);
#endif
   }
   else
   {
      fprintf (stderr, "TPCD_FLATFILES is not defined.\n");
      exit (-1);
   }

   su_semkey = ftok (sourcefile, 'J');
   while ( (su_semid = semget (su_semkey, 1, 0)) < 0)
   {
      if (errno == ENOENT) {
         sleep(2);
      }
      else {
         fprintf(stderr,"update set %d: semget failed errno = %d\n",
                 i, errno);
         exit(1);
      }
   }
   if (sem_op (su_semid, 0, 1) != 0)
/*jen SEM*/
   {
      fprintf(stderr,"Failure to increment semaphone UF1 set %d\n",i);
      fprintf(stderr,"  semaphore sourcefile = %s  su_semid =
su_semid\n",sourcefile);
      exit(1);
   }
/*jenSEM*/

#else  /* SQLWINT */
   sprintf (UF1_semfile, "%s.%s.UF1.semfile",
            getenv("TPCD_DBNAME"), getenv("USER"));
   fprintf(stderr,"UF1 semfile = %s\n",UF1_semfile);
   while ((su_hSem = OpenSemaphore(SEMAPHORE_ALL_ACCESS |
                                   SEMAPHORE_MODIFY_STATE |
                                   SYNCHRONIZE,
                                   TRUE,
                                   UF1_semfile))
          == (HANDLE)(NULL))
   {
      /*
      ** if cannot open the semaphore, wait for 0.1 second
      */
      fprintf(stderr,"Retry Open semaphore %s\n", UF1_semfile);

      sleep(1);
   }

   if (! ReleaseSemaphore(su_hSem,
                          1,
                          (LPLONG)(NULL)))
   {
      fprintf(stderr, "ReleaseSemaphore failed, LastError: %d, quit\n",
              GetLastError());
      exit(-1);
   }
#endif /* SQLWINT */
   exit(rc);                     /* child exiting after finishing up */
}

/*************************************************/
/* processing to run the delete update function */
/*************************************************/
void runUF2 ( int updatePair, int copyOnOrOff )
{
   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, j, c;
   char myoutstreamfile[256];
   FILE *myoutstream;
   char *buffer;
#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

   fprintf( outstream,"UF2 for update pair %d starting\n",updatePair);

   /* 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;

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

   /* 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 */
   sprintf (statement, "%s/tools/load_uf2 %d 2", sourcedir, updatePair);
   if (system(statement))
   {
      fprintf (stderr, "load_uf2 failed for UF2, examine UF2.log for
cause. Exiting.\n");
      exit (-1);
   }
   fprintf (outstream, "load_uf2 finished for UF2.\n");

   /* Next we need to get ready to launch a bunch of concurrent
processes */
#ifndef SQLWINT
   /*    we will use the first flat file to generate the semaphore key
*/
   if (getenv("TPCD_FLATFILES") != NULL)
```

```
      sprintf(sourcefile, "%s%cdelete.%d.00000.new",
              getenv("TPCD_FLATFILES"), PATH_DELIM, updatePair);
   else
      sprintf(sourcefile,".%cdelete.%d.00000.new", PATH_DELIM,
              updatePair);

   su_semkey = ftok (sourcefile, 'J');
   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)
      {
         runUF2_fn (updatePair, i, chunks_per_concurrent_delete);
      }
      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,
              "echo start /b %s\\auditruns\\tpcdbatch.exe -z -d %s -i %d
-j 2 -k %d -x 0",
              env_tpcd_audit_dir, dbname, updatePair, i ); /* aph */
         /* the -x parm should be passed at 0...not 100% sure of this
jen */
         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++)
   {
      if (verbose)
      {
         fprintf(stderr,"About to wait again ...Sets to wait for %d\n",
                 split_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);

}
```

```
void runUF2_fn ( int updatePair, int thisConcurrentDelete, int numChunks
)
{
   int rc = 0;
   long sqlcode;
   int maxwait;
   int startChunk = thisConcurrentDelete*numChunks; /* where do we
start? */
   long deletedLineitem = 0;  /*kmw*/
   long deletedOrders = 0;    /*kmw*/
   long savedDeletedOrders = 0; /*kbs*/

#ifndef SQLWINT
   int                 su_semid; /* semaphore for controlling split
updates*/
   key_t               su_semkey; /* key to generate semid */
#else
   HANDLE              su_hSem;
   char                UF2_semfile[256];
#endif

   char myoutstreamfile[256];
   FILE *myoutstream, *src_fh=NULL;

   EXEC SQL BEGIN DECLARE SECTION;
     long chunk;
   EXEC SQL END DECLARE SECTION;

   /* Generate the unique filename for this concurrent delete process */

   sprintf (myoutstreamfile, UF2OUTSTREAMPATTERN, env_tpcd_tmp_dir,
PATH_DELIM,
            updatePair, thisConcurrentDelete);
   if ( (myoutstream = fopen (myoutstreamfile, WRITEMODE)) == NULL)
   {
      fprintf (stderr,
               "\nThe output file '%s' for update pair %d set %d could
not be opened runUF2_fn.\n",
               myoutstreamfile,updatePair,thisConcurrentDelete);
      rc=-1;
      goto UF2_exit;
   }

   outstream=myoutstream;    /* initialize outstream for error_check
dxxxxhar*/

   fprintf( myoutstream,
            "\nUF2 for update pair %d set %d starting at %*.*s\n",
            updatePair, thisConcurrentDelete, T_STAMP_1LEN,T_STAMP_1LEN,
 /* TIME_ACC jen*/
            get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));  /*
TIME_ACC jen*/

#ifdef UF2DEBUG
   fprintf (myoutstream, "before connect\n");
   fflush(myoutstream);
#endif

   if (!strcmp(userid,"\0"))   /** No authentication provided **/
      EXEC SQL CONNECT TO :dbname;
   else
      EXEC SQL CONNECT TO :dbname USER :userid USING :passwd;
   error_check();

#ifdef UF2DEBUG
   fprintf (myoutstream, "after connect\n");
   fflush(myoutstream);
#endif

   /* Start processing each chunk in my range */
   for ( chunk = startChunk; chunk <= startChunk+numChunks; chunk++ )
   {

      /* Set things up for the loop which will retry if there is a
deadlock */
      sqlcode = SQL_RC_E911;
      maxwait = 1;
      rc = 0;

      while (sqlcode == SQL_RC_E911 && maxwait <= MAXWAIT && rc == 0)
      {

#ifdef UF2DEBUG
   fprintf (myoutstream, "in loop before orders exec sql\n");
   fflush(myoutstream);
#endif
        sqlcode = 0;

        EXEC SQL DELETE FROM TPCD.ORDERS
        WHERE O_ORDERKEY IN ( SELECT O_ORDERKEY
                              FROM TPCDTEMP.ORDERS_DEL
                              WHERE APP_ID = :chunk);
        if (sqlca.sqlcode < 0)
           sqlcode = error_check();

        if (sqlcode == SQL_RC_E911)
        {          /* we've hit a deadlock */
           fprintf (myoutstream,
              "\nA deadlock detected while deleting from ORDERS: update
pair %d set %d chunk %d.  Retrying.\n",
              updatePair, thisConcurrentDelete, chunk);
           dumpCa(&sqlca);
           SleepSome(UF_DEADLOCK_SLEEP);
           maxwait++;      /* jen DEADLOCK */
        }
        else if (sqlcode < 0)
        {
           fprintf (myoutstream,"\n%s\n", stmt_str.data);
           fprintf (myoutstream,"\nsqlcode %d occurred deleting from
TPCD.ORDERS\n", sqlca.sqlcode);
           dumpCa(&sqlca);
           fprintf (myoutstream,
                    "for update pair number %d set %d chunk
%d..Exiting\n",
                    updatePair, thisConcurrentDelete,chunk);
           rc=-1;
        }
```

```
        else
        {
           /* accumulate the number of row deleted */
           savedDeletedOrders = sqlca.sqlerrd[2];   /*kbs*/

#ifdef UF2DEBUG
           fprintf (myoutstream, "in loop for update pair number %d set
%d chunk %d\n",
                    updatePair, thisConcurrentDelete,chunk);
           fflush(myoutstream);
#endif

           /* delete the lineitems now */

           EXEC SQL DELETE FROM TPCD.LINEITEM
                    WHERE L_ORDERKEY IN ( SELECT O_ORDERKEY
                                          FROM TPCDTEMP.ORDERS_DEL
                                          WHERE APP_ID = :chunk );
           if (sqlca.sqlcode < 0)
              sqlcode = error_check();

           if (sqlcode == SQL_RC_E911)
           {          /* we've hit a deadlock */
#ifdef UF2DEBUG
   fprintf (myoutstream, "lineitem deadlocked\n");
   fflush(myoutstream);
#endif
              fprintf (myoutstream,
              "\nA deadlock detected while deleting from LINEITEM:
update pair %d set %d chunk %d.  Retrying.\n",
              updatePair, thisConcurrentDelete, chunk);
              dumpCa(&sqlca);
              SleepSome(UF_DEADLOCK_SLEEP);
              maxwait++;      /* jen DEADLOCK */
           }
           else if (sqlcode < 0)
           {
#ifdef UF2DEBUG
   fprintf (myoutstream, "lineitem failed\n");
   fflush(myoutstream);
#endif
              fprintf (myoutstream,"\nAn error %d occurred deleting
from TPCD.LINEITEM\n",sqlca.sqlcode);
              dumpCa(&sqlca);
              fprintf (myoutstream,"for update pair number %d set %d
chunk %d..Exiting\n",
                       updatePair, thisConcurrentDelete,chunk);
              rc=-1;
           }
          else
           {
#ifdef UF2DEBUG
   fprintf (myoutstream, "lineitem succeeded\n");
   fflush(myoutstream);
#endif
              /* accumulate the number of row deleted */
              /* Order count ONLY updated if both orders and lineitem
*/
              /* go through */
              deletedOrders += savedDeletedOrders;   /* kbs */
              deletedLineitem += sqlca.sqlerrd[2];
              rc=0;
              EXEC SQL COMMIT WORK;
              error_check();
           }
        } /* process lineitem deletes */
     } /* while trying to delete one chunk loop */
   } /* while there are more chunks */

#ifdef UF2DEBUG
   fprintf (myoutstream, "after loop\n");
   fflush(myoutstream);
#endif
   /* report the number of row deleted */
   fprintf(myoutstream, "%ld rows deleted from TPCD.ORDERS at %*.*s\n",
            deletedOrders,T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
            get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));  /*
TIME_ACC jen*/
   fprintf(myoutstream, "%ld rows deleted from TPCD.LINEITEM at
%*.*s\n",
            deletedLineitem,T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
            get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));  /*
TIME_ACC jen*/

   if (sqlca.sqlcode < 0)
   {
      fprintf (myoutstream,"# of deadlocks %d exceeds %i\n",
maxwait,MAXWAIT);
      rc=-1;
      EXEC SQL ROLLBACK WORK;
      error_check();                  /* @d22275 tjg */
   }

UF2_conn_reset:                      /*971101jen*/
   EXEC SQL CONNECT RESET;
   error_check();                    /* @d22275 tjg */

UF2_exit:
   fclose (myoutstream);

   /* exiting, increment the semaphore */
#ifndef SQLWINT
   /* we used the first flat file to generate the semaphore key */
   if (getenv("TPCD_FLATFILES") != NULL)
      sprintf(sourcefile, "%s%cdelete.%d.00000.new",
               getenv("TPCD_FLATFILES"), PATH_DELIM, updatePair);
   else
      sprintf(sourcefile,".%cdelete.%d.00000.new", PATH_DELIM,
               updatePair);
   su_semkey = ftok (sourcefile, 'J');
   while ((su_semid = semget(su_semkey,1,0)) < 0)
   {
      if (errno == ENOENT)
         sleep(2);
      else {
         fprintf(stderr,"UF2 update stream %d: semget failed errno =
%d\n",
                  updatePair, errno);
         exit(1);
      }
   }
   if (sem_op (su_semid, 0, 1) != 0 )     /*jenSEM*/
   {                                       /*jenSEM*/
      fprintf(stderr,"Failure to increment semaphone UF2 set %d\n",
thisConcurrentDelete);
      exit(1);
   }
/*jenSEM*/

#else
   sprintf (UF2_semfile, "%s.%s.UF2.semfile",
            getenv("TPCD_DBNAME"), getenv("USER"));
   fprintf(stderr,"UF2 semfile = %s\n",UF2_semfile);
   while ((su_hSem = OpenSemaphore(SEMAPHORE_ALL_ACCESS |
                                   SEMAPHORE_MODIFY_STATE |
                                   SYNCHRONIZE,
                                   TRUE,
                                   UF2_semfile))
          == (HANDLE)(NULL)) {
      /*
      ** if cannot open the semaphore, wait for 0.1 second
      */
      fprintf(stderr,"Retry Open semaphore %s\n", UF2_semfile);

      SleepSome(1);
   }

   if (! ReleaseSemaphore(su_hSem,
                          1,
                          (LPLONG)(NULL)))
   {
      fprintf(stderr, "ReleaseSemaphore failed, LastError: %d, quit\n",
              GetLastError());
      exit(-1);
   }
#endif

   exit(rc);                     /* child exiting after finishing up */
}

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

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

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


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

   if (g_struct->stream_list == NULL)
      if ((g_struct->stream_list =
                   fopen(g_struct->c_l_opt->str_file_name, READMODE))
== NULL)
      {
         fprintf(stderr,"\nThe stream list file could not be opened.");
         fprintf(stderr,"Make sure that the filename is correct.\n");
         exit(-1);
      }

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

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

   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 ( ( g_struct->sem_on == 1 ) &&
            ( g_struct->c_l_opt->intStreamNum != 0 ) )
       {
          /* yes we are to be using semaphores */
          /* is this the 1st time into update function 1??? */
          if (currentUpdatePair == updatePairStart )
          {
#ifdef SQLWINT
             fprintf(stderr,"numstreams =
%d\n",g_struct->c_l_opt->intStreamNum);
             fprintf(stderr,"semfile = %s\n",g_struct->sem_file);
             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");
             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,"Done waiting! 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
             /* create a semaphore key...use the name of a file that */
             /* you know exists */

             semkey = ftok(g_struct->update_num_file,'J');
             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));
             }

             /* 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\n");
             semctl(semid,0,IPC_RMID,0);  /* we've finished waiting, now
*/
                                                /* remove the semaphore */
#endif
          }
          /* otherwise continue to run*/
       }

       if (g_struct->c_l_opt->update != 5) {
          runUF1(currentUpdatePair, g_struct->copy_on_load);
       }
       rc = FALSE;
       break;
     case TPCDBATCH_DELETE:
       if (g_struct->c_l_opt->update != 5) {
          runUF2(currentUpdatePair, g_struct->copy_on_load);
       }
       currentUpdatePair += 1;
       /* update the update.pair.num file to reflect the successfully
completed */
       /* update pair */
       if (g_struct->c_l_opt->update != 5)
       {    /*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))
         fprintf (outstream, "\n\nNumber of rows retrieved is: %6 %d",
                  rows_fetch);
         else
            fprintf (outstream, "\n\nNumber of rows retrieved is: %6
%d",
                     s_info_ptr->max_rows_fetch);
      }                                                          /* @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;

#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->a_commit) {
      EXEC SQL COMMIT WORK;
      error_check();                                          /* @d22275
tjg */
   }

   s_info_ptr->elapse_time = get_elapsed_time(start_time);

   if (g_struct->c_flags->time_stamp == TRUE)                /* @d25594
tjg */
      strcpy(s_info_ptr->end_stamp,
      get_time_stamp(T_STAMP_FORM_3,(time_t)NULL) );

   /* 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);


   fflush(outstream);

   /** 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 */
#else
   HANDLE               hSem;
#endif

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

   if (g_struct->c_l_opt->intStreamNum > 0)
   {
      /* print out the stream stop time in the stream count information
file*/
      /* for the throughput tests only (intStreamNum>0)*/
      if (g_struct->c_l_opt->update == 2 ||
          g_struct->c_l_opt->update == 5)
      {
         /* 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 streams */
         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);
   }


   EXEC SQL CONNECT RESET;
   error_check();                                                 /* @d22275
tjg */

   /* 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 ( ( g_struct->sem_on == 1 ) &&
        ( g_struct->c_l_opt->update == 0) &&
        ( g_struct->c_l_opt->intStreamNum > 0 ) )
   {



      /* create a semaphore key...use the name of a file that */
      /* you know exists */

#ifndef SQLWINT

      semkey = ftok(g_struct->update_num_file,'J');
      /* Okay, now's the time to increment the semaphore by the update
stream */
      while ((semid = semget(semkey,1,0)) < 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);
         }
      }
      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);
      }
      /* this stream is done...increment by one */
      if (sem_op(semid, 0, 1) != 0)                              /*jenSEM*/
      {                                                          /*jenSEM*/
         fprintf(stderr,
                 "Failed to increment semaphone for throughput stream
%d\n",
                 g_struct->c_l_opt->intStreamNum);
         fprintf(stderr,
                 "file for generation of semaphore is: %s\n",
                 g_struct->update_num_file);
         exit(1);
      }
/*jenSEM*/


#else
      while ((hSem = OpenSemaphore(SEMAPHORE_ALL_ACCESS |
                                   SEMAPHORE_MODIFY_STATE |
                                   SYNCHRONIZE,
                                  TRUE,
                                  g_struct->sem_file))
                     == (HANDLE)(NULL)) {
          /*
          ** if cannot open the semaphore, wait for 0.1 second
          */
          fprintf(stderr,"Retry Open semaphore
%s\n",g_struct->sem_file);

          Sleep(1000);
      }

      if (! ReleaseSemaphore(hSem,
                             1,
                             (LPLONG)(NULL))) {
          fprintf(stderr, "ReleaseSemaphore failed, LastError: %d,
quit\n",
                  GetLastError());
          exit(-1);
      }
#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);
}
```

D.2 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";

# 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";
}

#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"};

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");
}

# activate the database
system("db2 activate database $dbname");

#get log info
system("db2_all \'|}db2 get db cfg for tpcd | grep -i log >>
$runDir${delim}startLog.Info\' ");
system("ls -ltra /node??vg.log/NODE00* >>
$runDir${delim}startLog.Info");

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='TPCDBATC'\"; 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 ($gatherstats eq "on")
{
    # gather vm io and net stats
    if ($platform eq "aix")
    {
       # 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";
    }
}
if ( $runUF ne "UF" )
{
  print "Beginning power stream....no update functions\n";
  # run the 17 queries for the powertest (stream = 0) with NO update
functions
  $ret=system("$auditDir${delim}auditruns${delim}tpcdbatch -d $dbname -f
$runDir${delim}qtextpow.sql -r on -b on -s $sf -u p -m $inlistmax -n 0
-l $auditDir${delim}auditruns${delim}querytext${delim}streampow.list");
}
else
{
  print "Beginning power stream....with update functions\n";
  # run the 17 queries for the powertest (stream = 0) with the update
functions
  $ret=system("$auditDir${delim}auditruns${delim}tpcdbatch -d $dbname -f
$runDir${delim}qtextp.sql -r on -b on -s $sf -u p -m $inlistmax -n 0 -l
$auditDir${delim}auditruns${delim}querytext${delim}stream0.list");
}
if ($ret == 0)
{
    print "Power stream completed succesfully.\n";
}
else
{
    print "Power stream failed. ret=$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")
    {
       # kill the stats that were being gathered
        $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='TPCDBATC'\";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");
}

# deactivate the database
system("db2 deactivate database $dbname");

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("db2 get database configuration for $dbname >> $dbtunefile");
  system("db2 get database manager configuration >> $dbtunefile");
  system("db2set >> $dbtunefile");

}

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='TPCDBATC';\n";
  print VERTBL "connect reset;\n";
  print VERTBL "terminate;\n";
  close(VERTBL);
  system("db2 -vtf $file >> $logfile");
}
```

D.3 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";

$runUF="no";
if (@ARGV > 0)
{
  if ($ARGV[0] eq "UF")
  {
    $runUF=$ARGV[0];
    if ( @ARGV > 1)
    {
        $numStream=$ARGV[1];
    }
    else
    {
        $numStream=0;
    }
    shift;
  }
  else
  {
     $numStream=$ARGV[0];
  }
}
else
{
    $numStream=0;
}

if (length($ENV{"TPCD_AUDIT_DIR"}) <= 0)
{
  die "TPCD_AUDIT_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_RUN_DIR"}) <= 0)
{
  die "TPCD_RUN_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_DBNAME"}) <= 0)
{
  die "TPCD_DBNAME environment variable not set\n";
}
if (length($ENV{"TPCD_RUNNUMBER"}) <= 0)
{
  die "TPCD_RUNNUMBER environment variable not set\n";
}
if (length($ENV{"TPCD_NUMSTREAM"}) <= 0)
{
  die "TPCD_NUMSTREAM environment variable not set\n";
}
if (length($ENV{"TPCD_RUN_ON_MULTIPLE_NODES"}) <= 0)
{
  die "TPCD_RUN_ON_MULTIPLE_NODES environment variable not set\n";
}
if (length($ENV{"TPCD_SF"}) <= 0)
{
  die "TPCD_SF environment variable not set\n";
}
if (length($ENV{"TPCD_PRODUCT"}) <= 0)
{
  die "TPCD_PRODUCT environment variable not set\n";
}
if (length($ENV{"TPCD_PLATFORM"}) <= 0)
{
  die "TPCD_PLATFORM environment variable not set\n";
}
if (length($ENV{"TPCD_AUDIT"}) <= 0)
{
    die "Must set TPCD_AUDIT env't var.  Real audit timing sequence run
if yes\n";
}

#set up local variables
$runNum=$ENV{"TPCD_RUNNUMBER"};
if ( $numStream == 0 )
{
  $numStream=$ENV{"TPCD_NUMSTREAM"};
}
$runDir=$ENV{"TPCD_RUN_DIR"};
$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$dbname=$ENV{"TPCD_DBNAME"};
$sf=$ENV{"TPCD_SF"};
$product=$ENV{"TPCD_PRODUCT"};
$multinode=$ENV{"TPCD_RUN_ON_MULTIPLE_NODES"};
$platform=$ENV{"TPCD_PLATFORM"};
$delim=$ENV{"TPCD_PATH_DELIM"};
$RealAudit=$ENV{"TPCD_AUDIT"};
$inlistmax=$ENV{"TPCD_INLISTMAX"};
$gatherstats=$ENV{"TPCD_GATHER_STATS"};

# return 1 if the given pattern(parameter $_[0]) matches any file
sub existfile {
  if ($platform eq "aix")
  {
    `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;
}

# if we are in real audit mode then we must start the db manager now
if ( $RealAudit eq "yes" )
{
    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='TPCDBATC'\" >> $runDir${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.
if ( $platform eq "aix" )
{
   system("watchstreams &");
}
elsif ( $platform eq "nt" )
{
    system("start perl watchstreams");
}
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")
    {
       # gather vmstats and iostats (and net stats if in mpp mode)
       system("perl getstats t &");
    }
    else
    {
        print "Stats gather not set up for current platform
$platform\n";
    }
}

if ( $multinode ne "yes" )
{
  # we are running the query streams and update stream from the same
node or
  # from a serial db....use semaphores for control of the update stream

  # the auditruns directory is where we have already generated the sql
files
  # for the updates and the power tests

  $loopStream=1;
```

```
  for ( $loopStream = 1; $loopStream <= $numStream; $loopStream++)
  {
    print "starting stream $loopStream\n";
    system("echo Executing stream $loopStream out of $numStream.");
    # run the 17 queries
    if ( $platform eq "aix" )
    {
        system("$auditDir${delim}auditruns${delim}tpcdbatch -d $dbname
-f $runDir${delim}qtextt$loopStream.sql -r on -b on -s $sf -u t1 -m
$inlistmax -n $loopStream -l
$auditDir${delim}auditruns${delim}querytext${delim}stream$loopStream.lis
t &");
    }
    elsif ( $platform eq "nt" )
    {
        system("start /b $auditDir${delim}auditruns${delim}tpcdbatch -d
$dbname -f $runDir${delim}qtextt$loopStream.sql -r on -b on -s $sf -u t1
-m $inlistmax -n $loopStream -l
$auditDir${delim}auditruns${delim}querytext${delim}stream$loopStream.lis
t");
    }
    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 $auditDir${delim}auditruns${delim}quft.sql -r on -b on -s $sf -u t -m
$inlistmax -n $numStream -l
$auditDir${delim}auditruns${delim}querytext${delim}streamuf.list");
  }
  else {
    $ret=system("$auditDir${delim}auditruns${delim}tpcdbatch -d $dbname
-f $auditDir${delim}auditruns${delim}quft.sql -r on -b on -s $sf -u t2
-m $inlistmax -n $numStream -l
$auditDir${delim}auditruns${delim}querytext${delim}streamuf.list");
  }
  print "update stream done\n";


  &getConfig("t");
}
else
{
  # we are running the query streams and update stream from different
nodes, use
  # files and rksh to control the update stream
  system("runthru.pe");
}

if ($platform eq "aix")
{
    # show the disks that are used/unused
    system("getdisks \"After the completion of the Throughput Test\"");
}
if ($gatherstats eq "on")
{
    # gather vm io and net stats
    if ($platform eq "aix")
    {
       # kill the stats that were being gathered
        $rc= `perl5 zap "-f" "vmstat"`;
        $rc= `perl5 zap "-f" "iostat"`;
        if ( $pn > 1 )
        {
            $rc= `perl5 zap "-f" "netstat"`;
        }
        $rc= `perl5 zap "-f" "getstats"`;
    }
}

#get log info
system("db2_all \']}db2 get db cfg for tpcd | grep -i log >>
$runDir${delim}endLog.Info\' ");
system("ls -ltra /node??vg.log/NODE00* >> $runDir${delim}endLog.Info");
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='TPCDBATC'\" >> $runDir${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 ( $platform eq "aix" )
{
   system("calcmetrics.pl $numStream");
}
elsif ( $platform eq "nt" )
{
    system("perl calcmetrics.pl $numStream");
}
else
{
    die "platform not supported, can't run calcmetrics ";
}

# concatenate all the throughput inter files that were used to
# generate these results into the calcmetrics output file
(mtinterX.metrics)
#cd $TPCD_RUN_DIR
&cat("$runDir${delim}mts*inter*","$runDir${delim}mtinter$runNum.metrics"
);

if ($runUF ne "no") {

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

if (&existfile("$runDir${delim}mp*")) {
  # generate the mplot stuff
  require 'gen_mplot';

  # generate the mlog information file
  require 'buildmlog';
}

if ($runUF eq "no") {
  &rm("$runDir${delim}mtuf*");
}

# deactivate the database
system("db2 deactivate database $dbname");

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("db2 get database configuration for $dbname >> $dbtunefile");
  system("db2 get database manager configuration >> $dbtunefile");
  system("db2set >> $dbtunefile");

}

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='TPCDBATC';\n";
  print VERTBL "connect reset;\n";
  print VERTBL "terminate;\n";
  close(VERTBL);
  system("db2 -vtf $file >> $logfile");
}
```

D.4 load_update_on_node

```
#!/bin/ksh

function get_envs {
   . /tmp/tpcd.envs
#  rm /tmp/tpcd.envs /tmp/tpcd.envs2
}

function runInsert_concat {
set -x;
     db2 connect to $dbname;
     str="db2 \"load from ";
    str2="db2 \"load from ";


str="$str$flatfilepath${delim}lineitem.tbl.u${updatepair}.${LN3}.new";

str2="$str2$flatfilepath${delim}order.tbl.u${updatepair}.${LN3}.new";
   str="$str of del modified by coldel| fastparse messages
${tempdir}${delim}line.msg.u${updatepair}.${LN3}.new remote file
/tmp/tpcd/remote.uf1_line replace into TPCDTEMP.LINEITEM_new
nonrecoverable cpu_parallelism 1 \" >
${tempdir}${delim}loaduf${function}.lineitem.u${updatepair}.${LN3}
2>&1";
   str2="$str2 of del modified by coldel| fastparse messages
${tempdir}${delim}ord.msg.u${updatepair}.${LN3}.new remote file
/tmp/tpcd/remote.uf1_ord replace into TPCDTEMP.ORDERS_new nonrecoverable
```

```
cpu_parallelism 1 \" >
${tempdir}${delim}loaduf${function}.order.u${updatepair}.${LN3} 2>&1";

#    print -- "$str";
#    print -- "$str2";
    eval "$str";
    eval "$str2";
  db2 connect reset; db2 terminate;
  return;
}
function runDelete_concat {
set -x;
     db2 connect to $dbname;
     str="db2 \"load from ";
     str="$str$flatfilepath${delim}delete.${updatepair}.${LN3}.new";
     str="$str of del modified by coldel| fastparse messages
${tempdir}${delim}del.msg.u${updatepair}.${LN3}.new remote file
/tmp/tpcd/remote.uf2_del replace into TPCDTEMP.ORDERS_DEL nonrecoverable
cpu_parallelism 1 \" >
${tempdir}${delim}loaduf${function}.delete.${updatepair}.${LN3} 2>&1";

#    print -- "$str";
#    print -- "$str2";
    eval "$str";
    eval "$str2";
  db2 connect reset; db2 terminate;
  return;
}

function runDelete {
  typeset -i chk=0;
  typeset -i ln;
  (( ln=LN3 ))

while [ chk -lt split_deletes ]; do

#     jj=0;
#     while [ jj -lt 48 ]; do
       $auditDir${delim}auditruns${delim}tpcdbatch -z -d $dbname -i
$updatepair -j 2 -k $ln -m $inlistmax -x $chk >
$tempdir${delim}uf2.$ln.$chk.out 2>&1 &
#      (( jj=jj+1 ));
      (( chk=chk+1 ))
#    done
#    wait;
  done;
  wait;
}

if [ $# -lt 2 ]; then
  echo "Usage: $0 pair function"
  exit -1;
fi;

cd $HOME/tpcd/tools;

updatepair=$1
function=$2
if [ $# -eq 3 ]; then
  inlistmax=$3;
fi;
#get_envs;
. /tmp/tpcd.envs;

export TPCD_PATH_DELIM="/";

hnm=$(hostname -s);
typeset -Z5 LN3;
LN3=$DB2NODE;
dbname=$TPCD_DBNAME;
auditDir=$TPCD_AUDIT_DIR;
uftemp=$TPCD_UFTEMP;
delim=$TPCD_PATH_DELIM;
split_updates=$TPCD_SPLIT_UPDATES;
concurrentload=$TPCD_CONCURRENT_INSERTS_LOAD;
split_deletes=$TPCD_SPLIT_DELETES;
uftemppath=$TPCD_UFTEMPPATH;
platform=$TPCD_PLATFORM;
flatfilepath=$TPCD_FLATFILES;
tempdir=$TPCD_TMP_DIR;
physnode=$TPCD_PHYS_NODE;
lnperpn=$TPCD_LN_PER_PN;

if [ function -eq 1 ]; then
#  runInsert;
   runInsert_concat;
else if [ function -eq 2 ]; then
#  runDelete;
  runDelete_concat;
else
  echo "Problem in function number"
  exit -1;
fi
fi
```

D.5 load_uf1

```
#!/usr/bin/perl
# usage load_uf1 updatepair function inlistmax
# where updatepair ($1) is the number of update pair to load
#       function ($2) is 1 = running INSERTs

($myName = $0) =~ s@.*/@@; $usage="
Usage: load_uf1 updatepair function inlistmax
    updatepair = number of update pair to load
    function   = UF to implement : 1 for INSERT\n";

die $usage if (@ARGV < 2);

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

# 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";

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"};

# 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;
}

# 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}order.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;
}

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_uf1\n");
}
1;
```

D6 load_uf2

```
#!/bin/ksh

if [ $# -lt 2 ]; then
  echo "Usage: $0 pair function <inlistmax>"
  exit -1;
fi;

updatepair=$1
function=$2
inlistmax="";
if [ $# -eq 3 ]; then
  inlistmax=$3;
fi
HOME=/u/tpcd;

function get_envs {
  envdir=$PWD;
  sed 's/\#.*$//g' $envdir/tpcd.setup > /tmp/tpcd.envs
  sed 's/\#.*$//g' $envdir/tpcd.runsetup >> /tmp/tpcd.envs
  sed '/^ *$/d' /tmp/tpcd.envs > /tmp/tpcd.envs2
  sed 's/^export /' /tmp/tpcd.envs2 > /tmp/tpcd.envs
  . /tmp/tpcd.envs
#  rm /tmp/tpcd.envs /tmp/tpcd.envs2
}

get_envs;
db2_all "||/u/tpcd/tpcd/tools/load_update_on_node $updatepair $function
$inlistmax"

exit 0;
```

Appendix E: ACID Transaction Source Code

E.1 acid.sqc

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

#include "acid.h"

#define DEADLOCK -911

#define TRUNC2(d) ((floor((d)*100))/100)

void sqlerror(char * , struct sqlca *);

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
char dbname[8];
EXEC SQL END DECLARE SECTION;

#ifdef SQLWINT

 /*
 ** redefine gettimeofday
 */
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;
   long    okey;
   long    lEprice;
   double  eprice;
   EXEC SQL END DECLARE SECTION;

   okey = acid->o_key;

   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 a 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 = 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): %0.2f\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;
   long      o_key, l_key, delta;
   long      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 = 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);
   }

   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("ototal= %0.3f\n",ototal);
   printf("new_ototal= %0.3f\n",new_ototal);
#endif

   EXEC SQL UPDATE tpcd.orders
      SET o_totalprice = :new_ototal
      WHERE CURRENT OF o_cursor;
   if (sqlca.sqlcode != 0) {
```

```
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: UPDATE o_cursor", &sqlca);
      goto Terror;
   }

   if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, after update of ORDER: (%us %06uu)
%s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
      fprintf(out, "updated o_totalprice: %0.3f\n", new_ototal) ;
   }

   if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, before insert into HISTORY: (%us
%06uu) %s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
   }

   EXEC SQL INSERT INTO tpcd.history values
      (:l_partkey, :l_suppkey, :o_key, :l_key, :delta, CURRENT
TIMESTAMP);
   if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: INSERT INTO history", &sqlca);
      goto Terror;
   }

   if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, after insert into HISTORY: (%us %06uu)
%s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
   }

   /* sleep for 1 second for 80% of the transactions */
#ifdef SQLWINT
   if ( ((rand() % (100)) + 1) < 80 ) sleep(1);
#else
   if ( ((random() % (100)) + 1) < 80 ) sleep(1);
#endif

   switch (acid->termination) {
    case 1:
      {
        if (acid->logging)
        {
          gettimeofday(&tv, &tz);
          time(&timeT);
          fprintf(out,"acidT tag: %d, wait before COMMIT: (%us %06uu)
%s",
                  mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
        }
      }
      sleep(60);
    case 0:
      if (acid->logging) {
         gettimeofday(&tv, &tz);
         time(&timeT);
         fprintf(out,"acidT tag: %d, immediately before COMMIT: (%us
%06uu) %s",
                 mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
      }
      EXEC SQL CLOSE L_CURSOR;
      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);
         } /* 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;
      EXEC SQL CLOSE O_CURSOR;
      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;
   long      availqty;
   long      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;
          } 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
}
```