IBM RISC System/6000 Workgroup Server F50-332

(with upgrade to 332MHz PowerPC 604e)

and

IBM RISC System/6000 Server Model H50

using

DB2 Universal Database Enterprise - Extended Edition for AIX, Version 5.0

TPC Benchmark™ D Full Disclosure Report

IBM System Performance and Evaluation Center

TPC Accepted
March 28, 1998
Amended
April 6, 1999

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 January 26, 1998
Second Edition May 7, 1998
Third Edition April 6, 1999

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

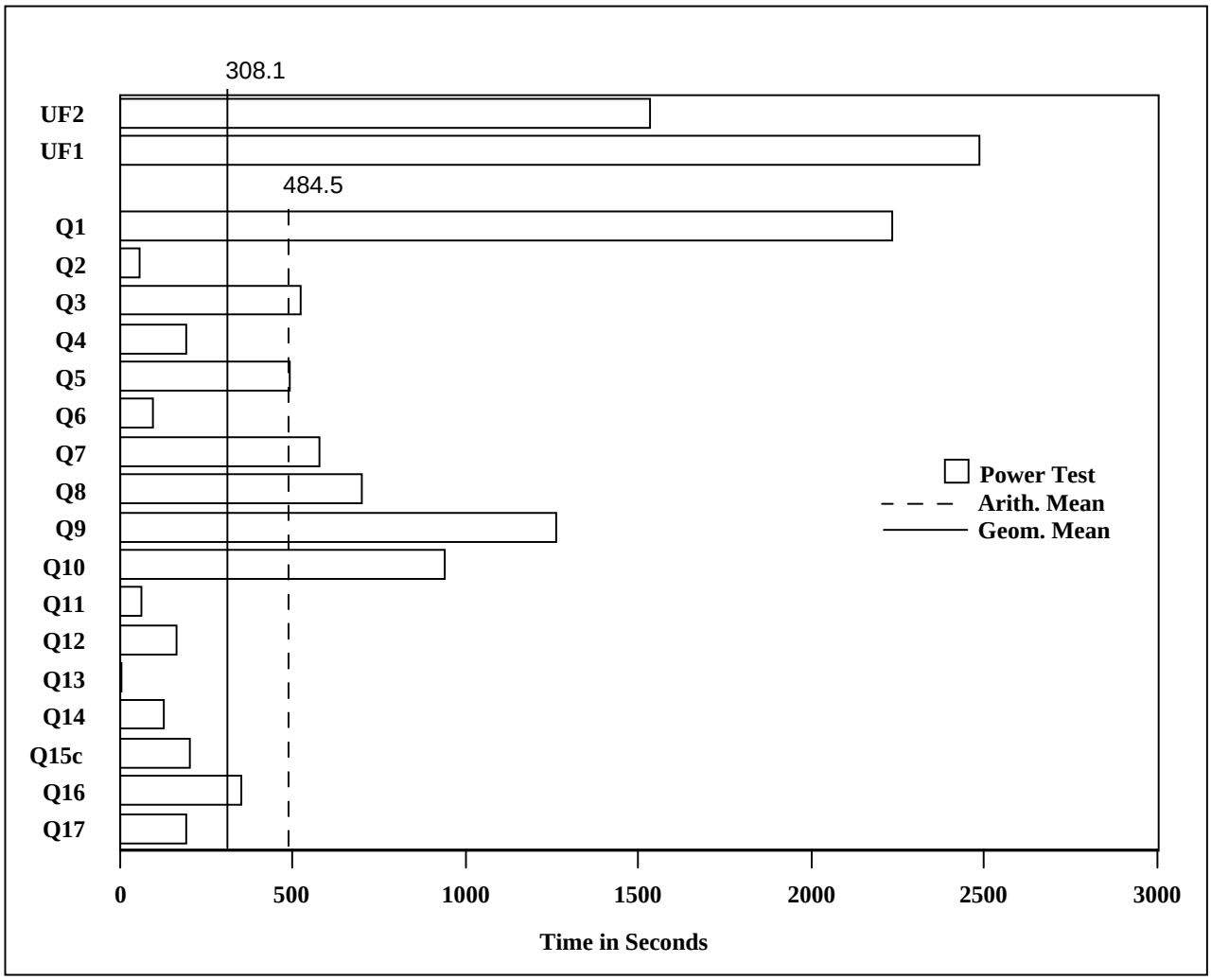
IBM RS/6000 F50-332 with DB2 UDB for AIX, V5

TPC-D Rev. 1.2.3

Report Date: January 26, 1998
Revised: April 6, 1999

Total System Cost	TPC-D Power	TPC-D Throughput	Price/Performance
$326,982	**1168.1** QppD@100GB	**499.1** QthD@100GB	**$428** $ / QphD@100GB

Database Size	Database Manager	Operating System	Other Software	Availability Date
100GB	DB2 UDB for AIX, V5	IBM AIX 4.2.1	None	March 31, 1998



Database Load Time: 28:38:33 hours	Total Data Storage/Database Size: 4.2	Raid: N

System Components

Processors	4 x 332MHz PowerPC 604e - 256K L2
Memory	3GB
Disk Controllers	1 PCI F/W SCSI 3 PCI SSA (Serial Storage Architecture)
Disk Drives	18 x 4.5GB F/W SCSI Hot Pluggable 80 x 4.5GB SSA Hot Pluggable
Total GB Storage	420.2 GB

IBM RS/6000 F50-332 with DB2 UDB for AIX, V5

TPC-D Rev. 1.2.3

Report Date: January 26, 1998
Revised: April 6, 1999

Server Model F50/332 4way TPC-D Pricing 1 Stream

Description	Part Number	Unit Price	Qty	Extended Price	5 yr. Maint Price
Server Hardware					
RS/6000 Server Model F50	7025-F50	**17,900**	1	**17,900**	11,952
1 604e, 128mb Memory, 4,5GB Disk,					
2 Intg SCSI2 F/W Adptr, CDROM					
332MHz 2way Proc Card Sel, 2-256kb L2	4357	**3,000**	1	**3,000**	2,640
332MHz 2way Proc Card, 2-256kb L2	4359	**10,000**	1	**10,000**	5,760
256MB DIMM Memory Select	4106	**1,280**	1	**1,280**	0
256MB DIMM Memory Modules	4110	**2,560**	11	**28,160**	0
Memory Expansion Feature 2nd Card	4093	1,038	1	1,038	0
16-bit Integrated SCSI Adapter Cable	2447	75	2	150	0
PCI SCSI-2 F/W Adapter	6208	360	1	360	0
PCI SSA 4port RAID Adapter	6215	3,000	3	9,000	0
Internal 4/8 GB 4mm Tape Drive	6142	2,695	1	2,695	0
ASCII Display Station & Cable	3153-BG3	622	1	622	660
		Subtotal		74,205	21,012
Server Software					
AIX 4.2.1 F50 + Support	5765-C34	**145**	1	**145**	0
C++ for AIX	5765-421	**1,578**	1	**1,578**	0
DB2 Enterprise Extended Edition for AIX	0795-115	34,999	1	34,999	0
DB2 EEE for AIX, up to 4 nodes	04L7-900	34,999	1	34,999	0
		Subtotal		71,721	0
Storage Devices					
4.5 GB SCSI-2 F/W Hot Swap Disk	2901	**1,400**	17	**23,800**	0
SCSI Hot Swap 6-Pack Kits	6519	375	2	750	0
System Rack Model R00	7015-R00	3,110	1	3,110	1,488
SSA Disk Subsystem (Rack-mount)	7133-020	19,350	5	96,750	48,000
4.5 GB Disk Drive Module	3401	**1,900**	60	**114,000**	0
SSA Cables	5006	40	10	400	0
		Subtotal		238,810	49,488
		Total		**384,736**	**70,500**
Discounts					
Midrange Service Agreement, Extended Maintenance		17%		(21,932.55)	
IBM Software Revenue Discount		17%		(11,899.66)	
Dollar Volume Discount		30%		(94,421.40)	
	Five Year Cost of Ownership:			**326,982.39**	
	QppD	1168.1			
	QthD	499.1			
	QphD	763.50			
	$/QphD@100GB			**428.27**	

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 based on assumptions about past or future purchases are not permitted. All discounts reflect standard pricing policies for the listed components. For complete details see the pricing sections of the TPC benchmark specifications. If you find that the stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org. Thank you.

Numerical Quantities Summary

Measurement Results:

Database Scaling (SF/Size)	=	100
Total Data Storage/Database Size	=	4.2
Database Load Time	=	28:38:33
Query Streams for Throughput Test	=	0
TPC-D Power Metric (QppD@100GB)	=	1,168.1
TPC-D Throughput Metric (QthD@100GB)	=	499.1
Composite Query-per-Hour Rating (QphD@100GB)	=	763.5
Total System Price over 5 years	=	$326,982
TPC-D Price/Performance Metric ($/QphD@100GB)	=	$428

Measurement Intervals

Measurement Interval in Throughput Test (Ts) = 12,263 seconds

Duration of Stream Execution

Stream ID	Seed Used	Start Date	Start Time	End Date	End Time
UF1		12/13/97	14:48:39	12/13/97	15:30:08
Stream 00	1056204987	12/13/97	14:48:39	12/13/97	18:13:02
UF2		12/13/97	17:47:24	12/13/97	18:13:02

Timing Intervals (in Seconds)

Stream ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
Stream 00	2237.6	60.6	527.3	193.6	494.7	101.5	579.4	701.2	1263.2

Stream ID	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	UF1	UF2
Stream 00	946.4	66.1	169.4	7.1	131.4	207.6	353.3	195.8	2489.1	1537.5

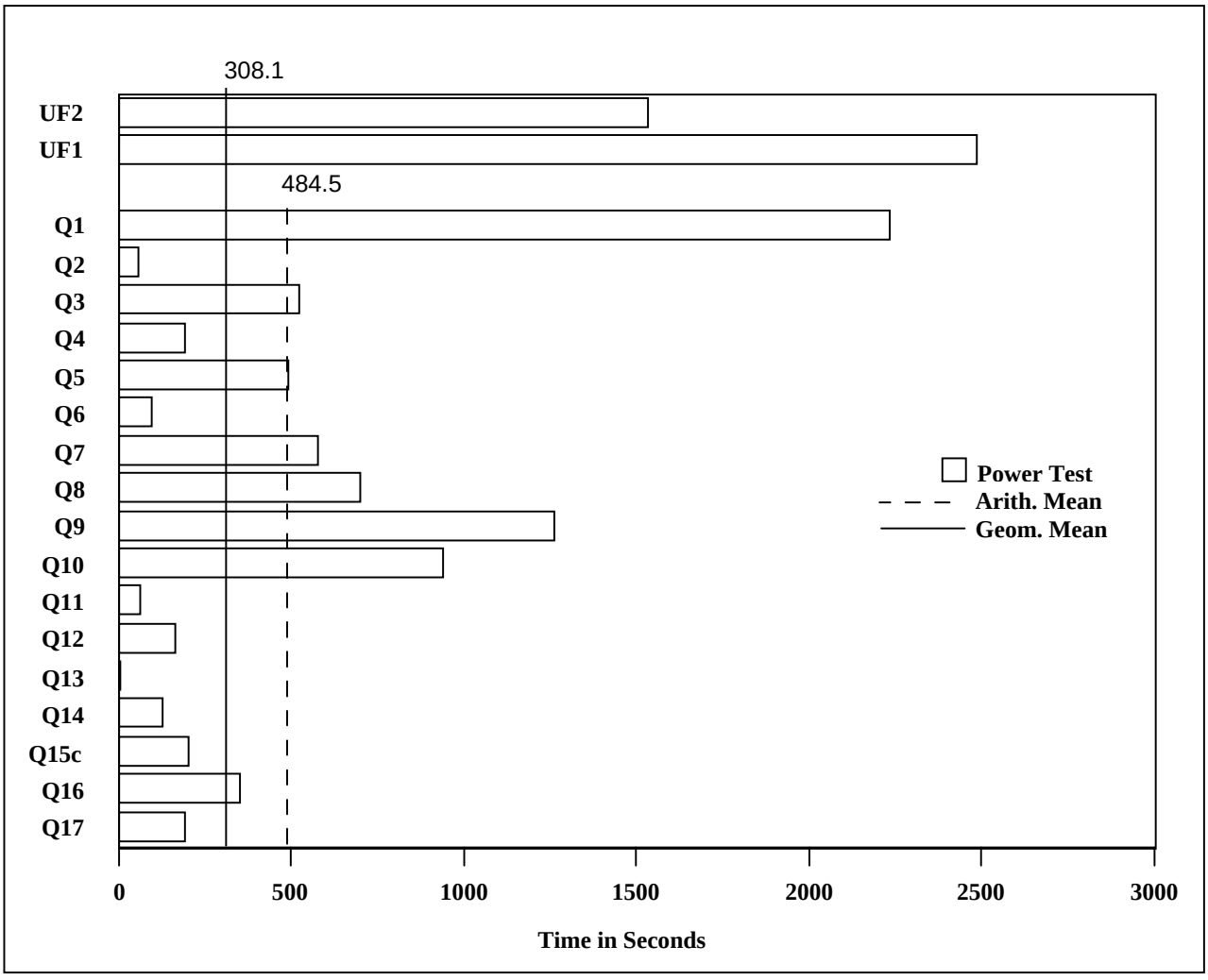
IBM RS/6000 H50
with DB2 UDB for AIX, V5

TPC-D Rev. 1.2.3

Report Date: January 26, 1998
Revised: April 6, 1999

Total System Cost	TPC-D Power	TPC-D Throughput	Price/Performance
$340,202	**1168.1** QppD@100GB	**499.1** QthD@100GB	**$446** $ / QphD@100GB

Database Size	Database Manager	Operating System	Other Software	Availability Date
100GB	DB2 UDB for AIX, V5	IBM AIX 4.2.1	None	March 31, 1998



Database Load Time: 28:38:33 hours	Total Data Storage/Database Size: 4.2	Raid: N

System Components

Processors	4 x 332MHz PowerPC 604e - 256K L2
Memory	3GB
Disk Controllers	1 PCI F/W SCSI 3 PCI SSA (Serial Storage Architecture)
Disk Drives	18 x 4.5GB F/W SCSI Hot Pluggable 80 x 4.5GB SSA Hot Pluggable
Total GB Storage	420.2 GB

IBM RS/6000 H50 with DB2 UDB for AIX, V5

TPC-D Rev. 1.2.3

Report Date: January 26, 1998
Revised: April 6, 1999

Server Model H50 4way TPC-D Pricing 1 Stream

Description	Part Number	Unit Price	Qty	Extended Price	5 yr. Maint Price
Server Hardware					
RS/6000 Server Model H50	7025-H50	**22,000**	1	**22,000**	10,992
128mb Memory, 4,5GB Disk,					
2 Intg SCSI2 F/W Adptr, CDROM					
332MHz 2way Proc Card Sel, 2-256kb L2	4338	**3,000**	1	**3,000**	1,920
332MHz 2way Proc Card, 2-256kb L2	4320	**10,000**	1	**10,000**	5,760
256MB DIMM Memory Select	4106	**1,280**	1	**1,280**	0
256MB DIMM Memory Modules	4110	**2,560**	11	**28,160**	0
Memory Expansion Feature 2nd Card	4093	1,038	1	1,038	0
16-bit Integrated SCSI Adapter Cable	2447	75	2	150	0
PCI SCSI-2 F/W Adapter	6208	360	1	360	0
PCI SSA 4port RAID Adapter	6215	3,000	3	9,000	0
Internal 4/8 GB 4mm Tape Drive	6142	2,695	1	2,695	0
ASCII Display Station & Cable	3153-BG3	622	1	622	660
		Subtotal		78,305	19,332
Server Software					
AIX 4.2.1 H50 + Support	5765-C34	**145**	1	**145**	0
C++ for AIX	5765-421	**1,578**	1	**1,578**	0
DB2 Enterprise Extended Edition for AIX	0795-115	34,999	1	34,999	0
DB2 EEE for AIX, up to 4 nodes	04L7-900	34,999	1	34,999	0
		Subtotal		71,721	0
Storage Devices					
4.5 GB SCSI-2 F/W Hot Swap Disk	2901	**1,400**	11	**15,400**	0
SCSI Hot Swap 6-Pack Kits	6519	375	2	750	0
Multi-Storage Tower w/ 2 x 4.5 GB Disks	7131-105	**6,150**	1	**6,150**	6,720
4.5 GB SCSI-2 1" Hot-swap Disk	3090	**1,900**	3	**5,700**	0
4.5 GB SCSI-2 1" non Hot-swap Disk	3036	**1,800**	1	**1,800**	0
System Rack Model R00	7015-R00	3,110	2	6,220	2,976
SSA Disk Subsystem w/ 4 - 4.5GB Disks	7133-020	19,350	5	96,750	48,000
4.5 GB Disk Drive Module	3401	**1,900**	60	**114,000**	0
SSA Cables	5006	40	10	400	0
		Subtotal		247,170	57,696
		Total		**397,196**	**77,028**
Discounts					
Midrange Service Agreement, Extended Maintenance		17%		(23,963.41)	
IBM Software Revenue Discount		17%		(11,899.66)	
Dollar Volume Discount		30%		(98,159.40)	
	Five Year Cost of Ownership:			**340,202**	
	QppD	1168.1			
	QthD	499.1			
	QphD	763.54			
	$/QphD@100GB			**446**	

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 based on assumptions about past or future purchases are not permitted. All discounts reflect standard pricing policies for the listed components. For complete details see the pricing sections of the TPC benchmark specifications. If you find that the stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org. Thank you.

IBM RS/6000 H50 with DB2 UDB for AIX, V5

TPC-D Rev. 1.2.3

Report Date: January 26, 1998
Revised: April 6, 1999

Numerical Quantities Summary

Measurement Results:

Database Scaling (SF/Size)	=	100
Total Data Storage/Database Size	=	4.2
Database Load Time	=	28:38:33
Query Streams for Throughput Test	=	0
TPC-D Power Metric (QppD@100GB)	=	1,168.1
TPC-D Throughput Metric (QthD@100GB)	=	499.1
Composite Query-per-Hour Rating (QphD@100GB)	=	763.5
Total System Price over 5 years	=	$340,202
TPC-D Price/Performance Metric ($/QphD@100GB)	=	$446

Measurement Intervals

Measurement Interval in Throughput Test (Ts) = 12,263 seconds

Duration of Stream Execution

Stream ID	Seed Used	Start Date	Start Time	End Date	End Time
UF1		12/13/97	14:48:39	12/13/97	15:30:08
Stream 00	1056204987	12/13/97	14:48:39	12/13/97	18:13:02
UF2		12/13/97	17:47:24	12/13/97	18:13:02

Timing Intervals (in Seconds)

Stream ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
Stream 00	2237.6	60.6	527.3	193.6	494.7	101.5	579.4	701.2	1263.2

Stream ID	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	UF1	UF2
Stream 00	946.4	66.1	169.4	7.1	131.4	207.6	353.3	195.8	2489.1	1537.5

Information Paradigm

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL

Certified Auditor

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

December 19, 1997

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

Platform: IBM RS/6000 F50-332
DataBase Manager: DB2 UDB for AIX, Version 5
Operating System: IBM AIX Version 4.2.1

The results were:

CPU's Speed	Memory	Disks	QppD@100GB	QthD@100GB
IBM RS/6000 F50-332				
4 x PowerPC (332 MHz)	256 KB L2 3 GB	80 x 4.5 GB SSA 18 x 4.5 GB SCSI	1168.1	499.1

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 "Reset" method
- The throughput metric was computed using the results from the power test
- The ratio between the longest and the shortest query was such that no adjustment was necessary
- A compliant implementation specific layer was used
- The query text was produced using compliant variants and minor modifications

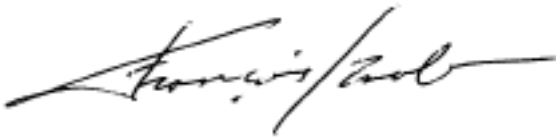
1373 North Franklin Street • Colorado Springs, CO 80903-2527 • Office : 719/473-7555 • Fax : 719/473-7554

- 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

RS/6000 F50-332 with DB2

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

Information Paradigm

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL

Certified Auditor

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

January 13, 1998

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

Platform: IBM RS/6000 H50-332
DataBase Manager: DB2 UDB for AIX, Version 5
Operating System: IBM AIX Version 4.2.1

The results were:

CPU's Speed	Memory	Disks	QppD@100GB	QthD@100GB
IBM RS/6000 H50-332				
4 x PowerPC (332 MHz)	256 KB L2 3 GB	80 x 4.5 GB SSA 18 x 4.5 GB SCSI	1168.1	499.1

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 "Reset" method
- The throughput metric was computed using the results from the power test
- The ratio between the longest and the shortest query was such that no adjustment was necessary
- A compliant implementation specific layer was used
- The query text was produced using compliant variants and minor modifications

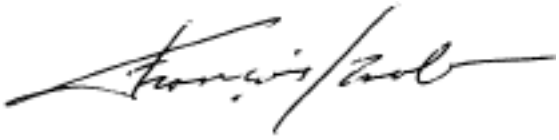
1373 North Franklin Street • Colorado Springs, CO 80903-2527 • Office : 719/473-7555 • Fax : 719/473-7554

- 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:

The performance measurements were done on a deskside Workgroup Server Model F50-332. The priced system is a rack-mounted Model H50-332. Both models have identical active components set in a different packaging.

Respectfully Yours,



François Raab
President

RS/6000 H50-332 with DB2

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 1.2.3 dated June 25, 1997, for measurements on the IBM RISC System/6000 Workgroup Server F50 with upgrade to 332MHz PowerPC 604e and the IBM RISC System/6000 Server Model H50. The phrase RS/6000 will be substituted for RISC System/6000 for the remainder of this document.

The software used includes AIX Version 4.2.1 and DB2 Universal Database Enterprise Extended Edition Version 5 for AIX.

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 1.2.3) on June 25, 1997. This is the full disclosure report for benchmark testing of the IBM RS/6000 Workgroup Server F50-332 and the IBM RS/6000 Server Model H50 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 and Runpower 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" .

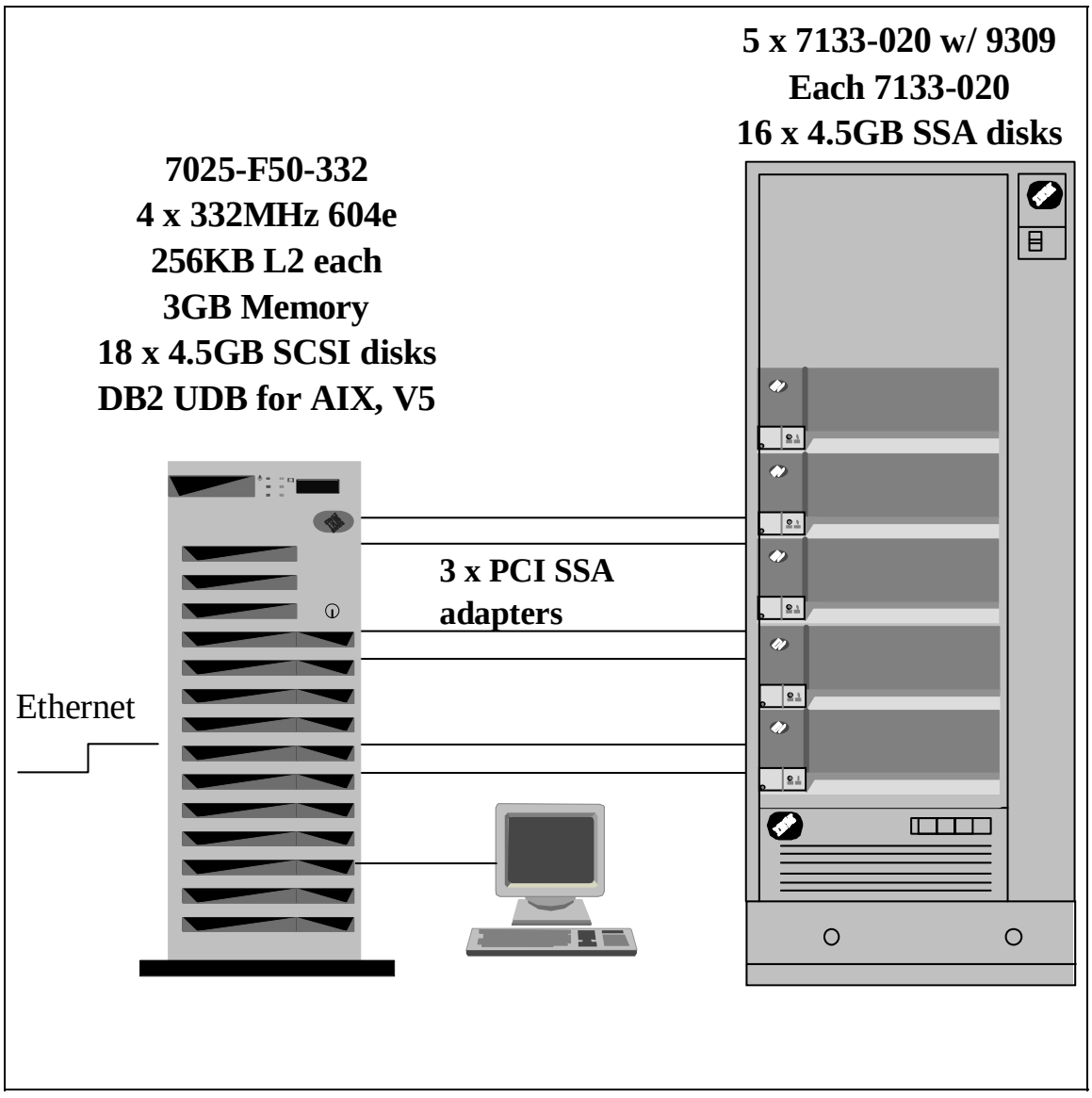
1.3 Configuration Diagrams

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*

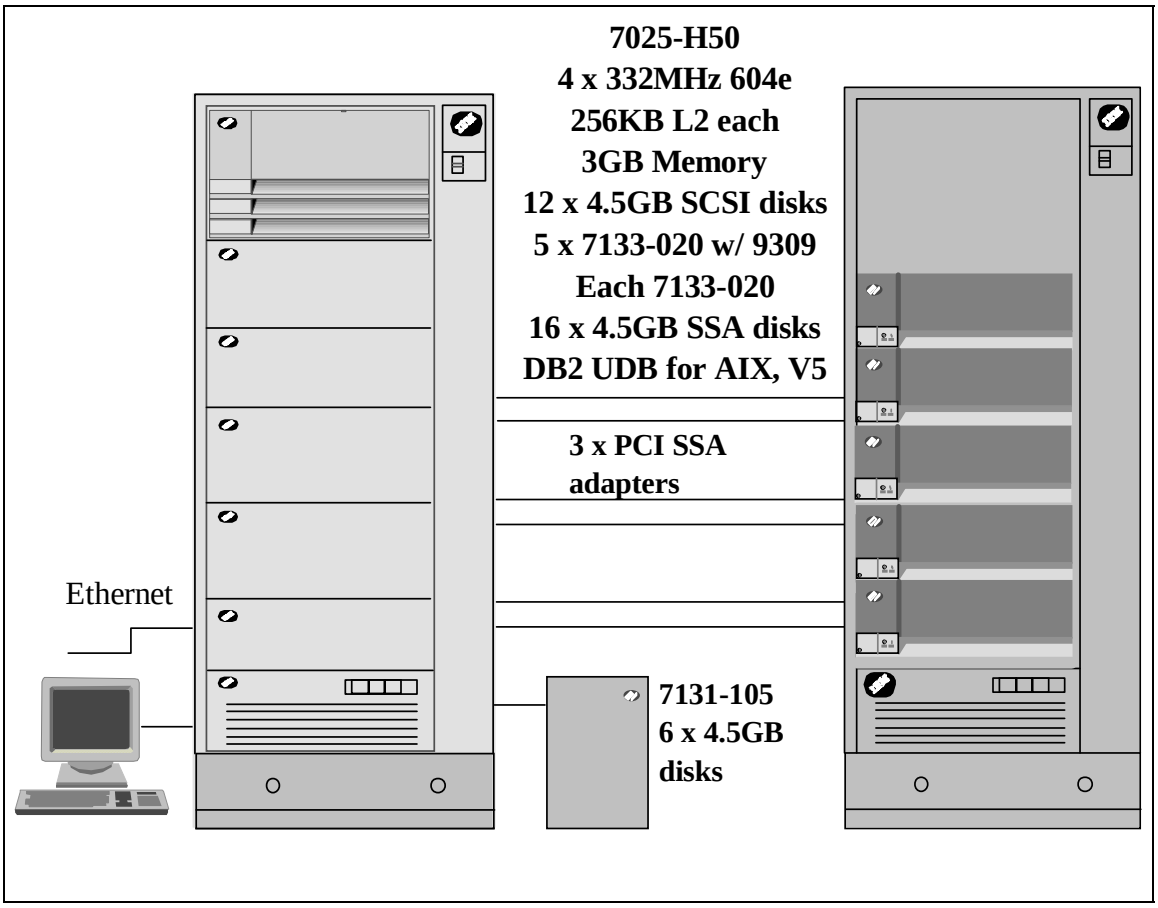
- *Number of LAN (e.g. Ethernet) connections, including routers, work stations, terminals, etc., that were physically used in the test or are incorporated into the pricing structure*
- *Type and run-time execution location of software components (e.g. DBMS, query processing tools/languages, middle-ware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

IBM RS/6000 Workgroup Server F50-332 Benchmark and Price Configuration



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

IBM RS/6000 Server Model H50 Price 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 and qualification databases. 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.

There are three phases for the loading of the database, the generation of the flat data files , the splitting and permuting of the data, and the building of the data from this data. The generation and splitting of the data use the gensplit100GB.ksh script, documented in Appendix B.2. The data is then permut ed using the sort.ksh script, documented in Appendix B.3. The buildtpcd script is then used to create the database, load the data into the tables, create indices, gather statistics, 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.4 through B.10

A brief description of the other files which are in Appendix B.11 thru B.17 follows:

- B.11 - DB2 nodes file (db2nodes.cfg) contains the names of the nodes where the database is to be build.
- B.12 - run config file (dss.dbconfig100.eastwoodmln) contains the database and 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.eastwood.mln).
- B.14 - DB2 commands to create the tables in the defined tablespaces (dss.ddl100GB.tbl.eastwood.mln).
- B.15 - DB2 commands to creates the indices for the tables (dss.index.prune.include)
- B.16 - DB2 commands to run statistics for the indices (dss.runstats)
- B.17 - 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)

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.15.

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 (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 1.2.0 and QGEN version 1.2.0 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 1.2.0 was used to generate the substitution parameters.

3.4 Query Text

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

Appendix C.1 , "Executable Query Text" contains the executable query text. Appendix C.2 , "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 and runpower 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 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: EXTENDEDPRICE = EXTENDEDPRICE + ((DELTA) * (EXTENDEDPRICE/QUANTITY)), TOTALPRICE = TOTALPRICE + (COST*(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 query was executed before and after a measurement to show that the database was always in a consistent state both initially and after a measurement.

```
SELECT DECIMAL(SUM(DECIMAL(INTEGER(INTEGER(DECIMAL
(INTEGER(100*DECIMAL(L_EXTENDEDPRICE,20,2)),20,2)*
(1-L_DISCOUNT)) * (1+L_TAX)),20,2)/100.0,20,2)
FROM TPCD.LINEITEM   WHERE L_ORDERKEY = O_ORDERKEY
```

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 query defined in 3.3.2 , "Consistency Condition" was run after initial database build and prior to executing the ACID transaction. The query showed that the database was in a consistent state.

After executing 100 ACID transactions the query defined in 3.3.2 , "Consistency Condition" was run again. The query 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 TPC-D query 1 (from TPC-D spec clause 2.3) 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 TPC-D query 1 with a randomly selected value of DELTA (not equal to 2000), was started.
4. T1 completed.
5. T2 completed.
6. T3 completed.
7. It was verified that the appropriate rows in the ORDERS, LINEITEM and HISTORY tables were changed.

4.4 Durability Requirements

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

4.4.1 Permanent Unrecoverable Failure of any Single Durable Medium

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

Failure of Durable Medium Containing Database Tables

1. The complete database is backed up to disk.
2. The consistency test 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 execution stream was started.
5. One of the disks containing database tables was powered off after at least 20 ACID transactions had completed.
6. The applications running the ACID transactions terminated after receiving an error from DB2.
7. The disk from step 4 was powered back on.
8. The DB2 database was restored from the backup copy in step 1.
9. DB2 was restarted and its transaction log was used to roll forward the transactions that had completed but weren't written to disk before the failure.
10. Step 3 was performed giving hist2. It was verified that hist2 - hist1 was equal or greater than the number of records in the success file.
11. Consistency condition was verified.

Failure of Recovery Log Data and System Crash

1. The consistency test was verified.
2. The current count of the total number of records in the HISTORY table was determined giving hist1.
3. A test to run 100 ACID transactions on each execution stream was started.
4. One of the disks containing DB2 transaction log data was powered off after at least 20 ACID transactions had completed.
5. The run was not affected because of log mirroring and ACID transactions continued to execute successfully.
6. The system was shutdown by switching the Emergency Power Off, after at least another 20 transactions had completed.
7. The system was powered back on and rebooted.
8. The mirrored partition on the disk was reestablished and re-syncronized.
9. Step 2 was performed giving hist2. It was verified that hist2 - hist1 was equal or greater than the number of records in the success file.
10. Consistency test 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

Adapter	Physical Volume	Logical Volume	Size (MB)	Description of contents
scsi 0	hdisk0	rootvg	4,200	OS, paging, user directories, applications, etc...
	hdisk1	rootvg	4,200	
	hdisk2	lv.2.A1	3,216	Temporary tablespace
	hdisk2	db2log	500	Mirrored filesystem for DB2 logs
	hdisk3	lv.3.A1	3,216	Temporary tablespace
	hdisk3	db2log	500	Mirrored filesystem for DB2 logs
	hdisk4	lv.4.A1	3,216	Temporary tablespace
	hdisk4	db2log	500	Mirrored filesystem for DB2 logs
	hdisk5	lv.5.A1	3,216	Temporary tablespace
	hdisk5	db2log	500	Mirrored filesystem for DB2 logs
	hdisk6	lv.6.A1	3,216	Temporary tablespace
	hdisk6	db2log	500	Mirrored filesystem for DB2 logs
	hdisk7	lv.7.A1	3,216	Temporary tablespace
	hdisk7	db2log	500	Mirrored filesystem for DB2 logs
	hdisk8	lv.8.A1	3,216	Temporary tablespace
	hdisk8	db2log	500	Mirrored filesystem for DB2 logs
	hdisk9	lv.9.A1	3,216	Temporary tablespace
	hdisk9	db2log	500	Mirrored filesystem for DB2 logs
	hdisk10	lv.10.A1	3,216	Temporary tablespace
	hdisk10	db2log	500	Mirrored filesystem for DB2 logs
	hdisk11	lv.11.A1	3,216	Temporary tablespace
	hdisk11	db2log	500	Mirrored filesystem for DB2 logs
	hdisk12	lv.12.A1	3,216	Temporary tablespace
	hdisk12	db2log	500	Mirrored filesystem for DB2 logs
	hdisk13	lv.13.A1	3,216	Temporary tablespace
	hdisk13	db2log	500	Mirrored filesystem for DB2 logs
	hdisk14	lv.14.A1	3,216	Temporary tablespace
	hdisk14	db2log	500	Mirrored filesystem for DB2 logs
	hdisk15	lv.15.A1	3,216	Temporary tablespace
	hdisk15	db2log	500	Mirrored filesystem for DB2 logs

	hdisk16	lv.16.A1	3,216	Temporary tablespace
	hdisk16	db2log	500	Mirrored filesystem for DB2 logs
	hdisk17	lv.17.A1	3,216	Temporary tablespace
	hdisk17	db2log	500	Mirrored filesystem for DB2 logs
ssa0	hdisk18	lv.hdisk18.B1	2,144	data tablespace (all tables)
	hdisk18	lv.hdisk18.C1	2,144	index tablespace (all indices)
	hdisk19	lv.hdisk19.B1	2,144	data tablespace (all tables)
	hdisk19	lv.hdisk19.C1	2,144	index tablespace (all indices)
	hdisk20	lv.hdisk20.B1	2,144	data tablespace (all tables)
	hdisk20	lv.hdisk20.C1	2,144	index tablespace (all indices)
	hdisk21	lv.hdisk21.B1	2,144	data tablespace (all tables)
	hdisk21	lv.hdisk21.C1	2,144	index tablespace (all indices)
	hdisk22	lv.hdisk22.B1	2,144	data tablespace (all tables)
	hdisk22	lv.hdisk22.C1	2,144	index tablespace (all indices)
	hdisk23	lv.hdisk23.B1	2,144	data tablespace (all tables)
	hdisk23	lv.hdisk23.C1	2,144	index tablespace (all indices)
	hdisk24	lv.hdisk24.B1	2,144	data tablespace (all tables)
	hdisk24	lv.hdisk24.C1	2,144	index tablespace (all indices)
	hdisk25	lv.hdisk25.B1	2,144	data tablespace (all tables)
	hdisk25	lv.hdisk25.C1	2,144	index tablespace (all indices)
	hdisk26	lv.hdisk26.B1	2,144	data tablespace (all tables)
	hdisk26	lv.hdisk26.C1	2,144	index tablespace (all indices)
	hdisk27	lv.hdisk27.B1	2,144	data tablespace (all tables)
	hdisk27	lv.hdisk27.C1	2,144	index tablespace (all indices)
	hdisk28	lv.hdisk28.B1	2,144	data tablespace (all tables)
	hdisk28	lv.hdisk28.C1	2,144	index tablespace (all indices)
	hdisk29	lv.hdisk29.B1	2,144	data tablespace (all tables)
	hdisk29	lv.hdisk29.C1	2,144	index tablespace (all indices)
	hdisk30	lv.hdisk30.B1	2,144	data tablespace (all tables)
	hdisk30	lv.hdisk30.C1	2,144	index tablespace (all indices)
	hdisk31	lv.hdisk31.B1	2,144	data tablespace (all tables)
	hdisk31	lv.hdisk31.C1	2,144	index tablespace (all indices)
	hdisk32	lv.hdisk32.B1	2,144	data tablespace (all tables)

	hdisk32	lv.hdisk32.C1	2,144	index tablespace (all indices)
	hdisk33	lv.hdisk33.B1	2,144	data tablespace (all tables)
	hdisk33	lv.hdisk33.C1	2,144	index tablespace (all indices)
	hdisk34	lv.hdisk34.B1	2,144	data tablespace (all tables)
	hdisk34	lv.hdisk34.C1	2,144	index tablespace (all indices)
	hdisk35	lv.hdisk35.B1	2,144	data tablespace (all tables)
	hdisk35	lv.hdisk35.C1	2,144	index tablespace (all indices)
	hdisk36	lv.hdisk36.B1	2,144	data tablespace (all tables)
	hdisk36	lv.hdisk36.C1	2,144	index tablespace (all indices)
	hdisk37	lv.hdisk37.B1	2,144	data tablespace (all tables)
	hdisk37	lv.hdisk37.C1	2,144	index tablespace (all indices)
	hdisk38	lv.hdisk38.B1	2,144	data tablespace (all tables)
	hdisk38	lv.hdisk38.C1	2,144	index tablespace (all indices)
	hdisk39	lv.hdisk39.B1	2,144	data tablespace (all tables)
	hdisk39	lv.hdisk39.C1	2,144	index tablespace (all indices)
	hdisk40	lv.hdisk40.B1	2,144	data tablespace (all tables)
	hdisk40	lv.hdisk40.C1	2,144	index tablespace (all indices)
	hdisk41	lv.hdisk41.B1	2,144	data tablespace (all tables)
	hdisk41	lv.hdisk41.C1	2,144	index tablespace (all indices)
	hdisk42	lv.hdisk42.B1	2,144	data tablespace (all tables)
	hdisk42	lv.hdisk42.C1	2,144	index tablespace (all indices)
	hdisk43	lv.hdisk43.B1	2,144	data tablespace (all tables)
	hdisk43	lv.hdisk43.C1	2,144	index tablespace (all indices)
	hdisk44	lv.hdisk44.B1	2,144	data tablespace (all tables)
	hdisk44	lv.hdisk44.C1	2,144	index tablespace (all indices)
	hdisk45	lv.hdisk45.B1	2,144	data tablespace (all tables)
	hdisk45	lv.hdisk45.C1	2,144	index tablespace (all indices)
	hdisk46	lv.hdisk46.B1	2,144	data tablespace (all tables)
	hdisk46	lv.hdisk46.C1	2,144	index tablespace (all indices)
	hdisk47	lv.hdisk47.B1	2,144	data tablespace (all tables)
	hdisk47	lv.hdisk47.C1	2,144	index tablespace (all indices)
	hdisk48	lv.hdisk48.B1	2,144	data tablespace (all tables)
	hdisk48	lv.hdisk48.C1	2,144	index tablespace (all indices)

	hdisk49	lv.hdisk49.B1	2,144	data tablespace (all tables)
	hdisk49	lv.hdisk49.C1	2,144	index tablespace (all indices)
ssa1	hdisk50	lv.hdisk50.B1	2,144	data tablespace (all tables)
	hdisk50	lv.hdisk50.C1	2,144	index tablespace (all indices)
	hdisk51	lv.hdisk51.B1	2,144	data tablespace (all tables)
	hdisk51	lv.hdisk51.C1	2,144	index tablespace (all indices)
	hdisk52	lv.hdisk52.B1	2,144	data tablespace (all tables)
	hdisk52	lv.hdisk52.C1	2,144	index tablespace (all indices)
	hdisk53	lv.hdisk53.B1	2,144	data tablespace (all tables)
	hdisk53	lv.hdisk53.C1	2,144	index tablespace (all indices)
	hdisk54	lv.hdisk54.B1	2,144	data tablespace (all tables)
	hdisk54	lv.hdisk54.C1	2,144	index tablespace (all indices)
	hdisk55	lv.hdisk55.B1	2,144	data tablespace (all tables)
	hdisk55	lv.hdisk55.C1	2,144	index tablespace (all indices)
	hdisk56	lv.hdisk56.B1	2,144	data tablespace (all tables)
	hdisk56	lv.hdisk56.C1	2,144	index tablespace (all indices)
	hdisk57	lv.hdisk57.B1	2,144	data tablespace (all tables)
	hdisk57	lv.hdisk57.C1	2,144	index tablespace (all indices)
	hdisk58	lv.hdisk58.B1	2,144	data tablespace (all tables)
	hdisk58	lv.hdisk58.C1	2,144	index tablespace (all indices)
	hdisk59	lv.hdisk59.B1	2,144	data tablespace (all tables)
	hdisk59	lv.hdisk59.C1	2,144	index tablespace (all indices)
	hdisk60	lv.hdisk60.B1	2,144	data tablespace (all tables)
	hdisk60	lv.hdisk60.C1	2,144	index tablespace (all indices)
	hdisk61	lv.hdisk61.B1	2,144	data tablespace (all tables)
	hdisk61	lv.hdisk61.C1	2,144	index tablespace (all indices)
	hdisk62	lv.hdisk62.B1	2,144	data tablespace (all tables)
	hdisk62	lv.hdisk62.C1	2,144	index tablespace (all indices)
	hdisk63	lv.hdisk63.B1	2,144	data tablespace (all tables)
	hdisk63	lv.hdisk63.C1	2,144	index tablespace (all indices)
	hdisk64	lv.hdisk64.B1	2,144	data tablespace (all tables)
	hdisk64	lv.hdisk64.C1	2,144	index tablespace (all indices)
	hdisk65	lv.hdisk65.B1	2,144	data tablespace (all tables)

	hdisk65	lv.hdisk65.C1	2,144	index tablespace (all indices)
	hdisk66	lv.hdisk66.B1	2,144	data tablespace (all tables)
	hdisk66	lv.hdisk66.C1	2,144	index tablespace (all indices)
	hdisk67	lv.hdisk67.B1	2,144	data tablespace (all tables)
	hdisk67	lv.hdisk67.C1	2,144	index tablespace (all indices)
	hdisk68	lv.hdisk68.B1	2,144	data tablespace (all tables)
	hdisk68	lv.hdisk68.C1	2,144	index tablespace (all indices)
	hdisk69	lv.hdisk69.B1	2,144	data tablespace (all tables)
	hdisk69	lv.hdisk69.C1	2,144	index tablespace (all indices)
	hdisk70	lv.hdisk70.B1	2,144	data tablespace (all tables)
	hdisk70	lv.hdisk70.C1	2,144	index tablespace (all indices)
	hdisk71	lv.hdisk71.B1	2,144	data tablespace (all tables)
	hdisk71	lv.hdisk71.C1	2,144	index tablespace (all indices)
	hdisk72	lv.hdisk72.B1	2,144	data tablespace (all tables)
	hdisk72	lv.hdisk72.C1	2,144	index tablespace (all indices)
	hdisk73	lv.hdisk73.B1	2,144	data tablespace (all tables)
	hdisk73	lv.hdisk73.C1	2,144	index tablespace (all indices)
	hdisk74	lv.hdisk74.B1	2,144	data tablespace (all tables)
	hdisk74	lv.hdisk74.C1	2,144	index tablespace (all indices)
	hdisk75	lv.hdisk75.B1	2,144	data tablespace (all tables)
	hdisk75	lv.hdisk75.C1	2,144	index tablespace (all indices)
	hdisk76	lv.hdisk76.B1	2,144	data tablespace (all tables)
	hdisk76	lv.hdisk76.C1	2,144	index tablespace (all indices)
	hdisk77	lv.hdisk77.B1	2,144	data tablespace (all tables)
	hdisk77	lv.hdisk77.C1	2,144	index tablespace (all indices)
	hdisk78	lv.hdisk78.B1	2,144	data tablespace (all tables)
	hdisk78	lv.hdisk78.C1	2,144	index tablespace (all indices)
	hdisk79	lv.hdisk79.B1	2,144	data tablespace (all tables)
	hdisk79	lv.hdisk79.C1	2,144	index tablespace (all indices)
	hdisk80	lv.hdisk80.B1	2,144	data tablespace (all tables)
	hdisk80	lv.hdisk80.C1	2,144	index tablespace (all indices)
	hdisk81	lv.hdisk81.B1	2,144	data tablespace (all tables)
	hdisk81	lv.hdisk81.C1	2,144	index tablespace (all indices)

ssa2	hdisk87	lv.hdisk87.B1	2,144	data tablespace (all tables)
	hdisk87	lv.hdisk87.C1	2,144	index tablespace (all indices)
	hdisk88	lv.hdisk88.B1	2,144	data tablespace (all tables)
	hdisk88	lv.hdisk88.C1	2,144	index tablespace (all indices)
	hdisk89	lv.hdisk89.B1	2,144	data tablespace (all tables)
	hdisk89	lv.hdisk89.C1	2,144	index tablespace (all indices)
	hdisk91	lv.hdisk91.B1	2,144	data tablespace (all tables)
	hdisk91	lv.hdisk91.C1	2,144	index tablespace (all indices)
	hdisk94	lv.hdisk94.B1	2,144	data tablespace (all tables)
	hdisk94	lv.hdisk94.C1	2,144	index tablespace (all indices)
	hdisk95	lv.hdisk95.B1	2,144	data tablespace (all tables)
	hdisk95	lv.hdisk95.C1	2,144	index tablespace (all indices)
	hdisk96	lv.hdisk96.B1	2,144	data tablespace (all tables)
	hdisk96	lv.hdisk96.C1	2,144	index tablespace (all indices)
	hdisk97	lv.hdisk97.B1	2,144	data tablespace (all tables)
	hdisk97	lv.hdisk97.C1	2,144	index tablespace (all indices)
	hdisk98	lv.hdisk98.B1	2,144	data tablespace (all tables)
	hdisk98	lv.hdisk98.C1	2,144	index tablespace (all indices)
	hdisk99	lv.hdisk99.B1	2,144	data tablespace (all tables)
	hdisk99	lv.hdisk99.C1	2,144	index tablespace (all indices)
	hdisk103	lv.hdisk103.B1	2,144	data tablespace (all tables)
	hdisk103	lv.hdisk103.C1	2,144	index tablespace (all indices)
	hdisk104	lv.hdisk104.B1	2,144	data tablespace (all tables)
	hdisk104	lv.hdisk104.C1	2,144	index tablespace (all indices)
	hdisk105	lv.hdisk105.B1	2,144	data tablespace (all tables)
	hdisk105	lv.hdisk105.C1	2,144	index tablespace (all indices)
	hdisk107	lv.hdisk107.B1	2,144	data tablespace (all tables)
	hdisk107	lv.hdisk107.C1	2,144	index tablespace (all indices)
	hdisk110	lv.hdisk110.B1	2,144	data tablespace (all tables)
	hdisk110	lv.hdisk110.C1	2,144	index tablespace (all indices)
	hdisk111	lv.hdisk111.B1	2,144	data tablespace (all tables)
	hdisk111	lv.hdisk111.C1	2,144	index tablespace (all indices)

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. indices) or temporary space the level of RAID used must be disclosed for each device.

RAID was not used.

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 1.2.0 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.3 , "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	18	4,288MB	77,184MB		
SSA	80	4,288MB	343,040MB		
Total			420.2GB	100GB	4.2

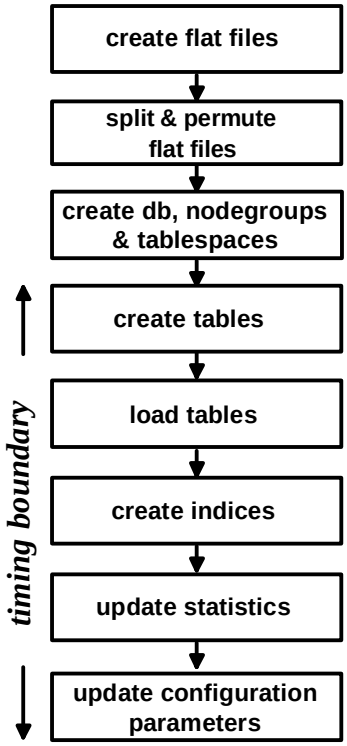
5.9 Details of Database Loading

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.

A single query stream throughput metric was calculated using the timings from the power test, as indicated in 5.3.1.4. A separate throughput test was not run.

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.

A throughput test was not run. Timings for the power test appear in the Numerical Quantities Summary.

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, with the higher QphD no more than 5% better than the lower QphD. The following table contains the reproducibility metrics for the system reported in this document.

	QppD@100GB	QthD@100GB	QphD@100GB
Run 1	1,168.1	499.1	763.5
Run 2	1,172.5	515	777
Difference	0.3%	3.2%	1.7%

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 and runpower 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.

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 starts m applications on each node, one for each of the m portions of delete data. Each application reads the keys for its portion on its node and performs the deletes using these key.

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 Revenue Discount: IBM Revenue Discount provides discounts for RS/6000 hardware based upon the total list price value. For the RS/6000 Workgroup Server F50 and Server Model H50 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 availability date for the hardware and system software used in this test is February 20, 1998. DB2 UDB for AIX, V5 is available March 31, 1998. The prices for all components are effective February 9, 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 and Database manager configuration taken at : Sat Dec 13
14:48:31 CST 1997
get database configuration for TPCD

       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) = 10
 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

*Database heap (4KB)                         (DBHEAP) = 2400
 Catalog cache size (4KB)             (CATALOGCACHE_SZ) = 64
*Log buffer size (4KB)                       (LOGBUFSZ) = 128
 Utilities heap size (4KB)              (UTIL_HEAP_SZ) = 5000
*Buffer pool size (4KB)                      (BUFFPAGE) = 125000
 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) = 4
*Number of I/O servers                     (NUM_IOSERVERS) = 21
 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) = 120
 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                                       =
/db2log/tpcdmln.log/NODE0000/
 Next active log file                                    = S0000043.LOG
 First active log file                                   = S0000043.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

```
get 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) = 1.377671e-06
*Communications bandwidth                (COMM_BANDWIDTH) = 2.000000e-01

*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) = 0
 Diagnostic data directory path               (DIAGPATH) =

 Default database monitor switches
   Buffer pool                          (DFT_MON_BUFPOOL) = OFF
   Lock                                    (DFT_MON_LOCK) = OFF
   Sort                                    (DFT_MON_SORT) = OFF
   Statement                               (DFT_MON_STMT) = OFF
   Table                                  (DFT_MON_TABLE) = OFF
   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
 Trust all clients                       (TRUST_ALLCLNTS) = YES
 Trusted client authentication           (TRUST_CLNTAUTH) = CLIENT

 Default database path                        (DFTDBPATH) = /home/haider

 Database monitor heap size (4KB)           (MON_HEAP_SZ) = 48
 UDF shared memory set size (4KB)            (UDF_MEM_SZ) = 256

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

*Sort heap threshold (4KB)                  (SHEAPTHRES) = 37000

 Directory cache support                      (DIR_CACHE) = YES

 Java Virtual Machine heap size (4KB)     (JAVA_HEAP_SZ) = 512

 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) = 0
 Initial number of agents in pool       (NUM_INITAGENTS) = 0
 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

 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) = 4096
*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 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=haider
export DB2INSTANCE

INSTHOME=/home/haider

#------------------------------------------------------------------------
# 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

export DB2_FORCE_FCM_BP=Y;
export DB2_VECTOR=Y;
export DB2VECTORSPACE=16000
export DB2_TABLEAPPENDMODE=yes
export DB2_RR_TO_RS=yes
```

A.4 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    4000    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     3137536 Amount of usable physical memory in Kbytes
False
conslogin   enable  System Console Login
False
fwversion   IBM,L97211 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
rtasversion 1       Open Firmware RTAS version
False
modelname   IBM,7025-F50  Machine name
   False
systemid    IBM,012605068 Hardware system identifier
   False
boottype    disk          N/A
   False
```

A.5 Compiler Options - Makefile.pe

```
#LOCAL=tpcd

BASE=$(HOME)/sqllib
COMPILE_FLAGS= -c -DSQLAIX -I$(BASE)/include -O2
#COMPILE_FLAGS= -c -DSQLAIX -I$(BASE)/include -g
LINK_FLAGS= -o $@ -L$(BASE)/lib -L/usr/lpp/db2_05_00/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 (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)
{
    $ENV{"TPCD_RUNSTATS"} = "dss.runstats";
}
if ( ($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_LOG_DIR"}) <= 0)
{
    $ENV{"TPCD_LOG_DIR"} = "NULL";
}
if (length($ENV{"TPCD_CONFIGFILE"}) <= 0)
{
    $ENV{"TPCD_CONFIGFILE"} = "dss.dbconfig";
}
if (length($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";
}



#set up local variables
$product=$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"};
$tbspddl=$ENV{"TPCD_TBSP_DDL"};
$indexddl=$ENV{"TPCD_INDEXDDL"};
$extraindex=$ENV{"TPCD_EXTRAINDEX"};
$runstats=$ENV{"TPCD_RUNSTATS"};
$dbgen=$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"};
if ( $fparse eq "yes" )
{
    $fastparse="FASTPARSE";
}
else
{
    $fastparse=" ";
}
$backupdir=$ENV{"TPCD_BACKUP_DIR"};
$log=$ENV{"TPCD_LOG"};
$logDir=$ENV{"TPCD_LOG_DIR"};
$machine=$ENV{"TPCD_MACHINE"};
$configfile=$ENV{"TPCD_CONFIGFILE"};
$smpdegree=$ENV{"TPCD_SMPDEGREE"};
$agentpri=$ENV{"TPCD_AGENTPRI"};
$activate=$ENV{"TPCD_ACTIVATE"};
$RealAudit=$ENV{"TPCD_AUDIT"};
if ( $RealAudit eq "yes" )
{
    # need some extra parameters for some of the setup
    if (length($ENV{"TPCD_AUDIT_DIR"}) <= 0)
    {
        die "TPCD_AUDIT_DIR environment variable not set\n";
    }
    $auditDir=$ENV{"TPCD_AUDIT_DIR"};
```

```
    $user=$ENV{"USER"};
}

# set up override of some parameters to override for qualification
database
if ( $qual eq "QUAL" )
{
    if ( length($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_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, indices and stats
if (( $machine ne "small" ) && ( $machine ne "medium" ) && ( $machine ne
"big" ))
{
    $hostname=$ENV{"HOSTNAME"};    #have to get the almaden.ibm.com
format?
    if ( $hostname eq "mistral" )
    {
        $machine="big";
    }
    print "set up for different hostnames...complete this\n";
}

$ioclnrs=1;
$chngpgs=60;

if ( $machine eq "small" )
{
    $buffpage = 5000;
    $sortheap = 3000;
    $sheapthres = 8000;
    $ioservers = 6;
}
elsif ( $machine eq "medium" )
{
    $buffpage = 10000;
    $sortheap =  8000;
    $sheapthres = 20000;
    $ioservers = 10;
}
elsif ( $machine eq "big" )
{
    $buffpage = 30000;
    $sortheap = 20000;
    $sheapthres = 50000;
    $ioservers = 20;
}
elsif ( $machine eq "sunsmp" )
{
    $buffpage = 60000;
    $sortheap = 20000;
    $sheapthres = 80000;
    $ioservers = 80;
}
elsif ( $machine eq "eastwood" )
{
    $buffpage = 80000;
    $sortheap = 50000;
    $sheapthres = 81000;
    $ioclnrs = 4;
    $chngpgs = 30;
    $ioservers = 21;
}
# echo parameter settings to acknowledge what is being built
print "Building a TPC-D $sf GB database on $dbpath with: \n";
print "   Mode = $mode \n";
print "   Tablespace ddl in $ddlpath${delim}$tbspddl \n";
print "   Table ddl in $ddlpath${delim}$ddl \n";
print "   Index ddl in                $ddlpath${delim}$indexddl\n";
if ( $extraindex ne "no" )
{
  print "             and               $ddlpath${delim}$extraindex\n";
}
print "   Runstats in                 $ddlpath${delim}$runstats \n";
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 "   Indices created before loading\n";
}
else
{
  print "   Indices created after loading\n";
}
if ( $loadstats eq "yes" )
{
  if ( $earlyindex eq "yes" )
  {
    print "   Statistics for tables and indices gathered during load\n";
  }
  else
  {
    print "   Statistics for tables and indices gathered after load\n";
  }
}
else
{
  print "   Statistics for tables and indices gathered after load\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 ( $configfile ne "NULL" )
{
  print "   Configuration parameters taken from $configfile\n";
}
else
{
  print "   Configuration paramters taken from
$ddlpath${delim}dss.dbconfig${sfReal}GB\n";
  $configfile="dss.dbconfig${sfReal}GB";
}
#print "   Copy image for load command created in $copydir\n";
if ( $log eq "yes" )
{
print "   Backup files placed in $backupdir\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";
}
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 "   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
$rc=system("db2set DB2OPTIONS=\"-t -v +c\" -i ");
# 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
&outtime("*** Starting to create the database");

&dodb_noconn("db2 \"create database $dbname on $dbpath collate using
identity with 'TPC-D $sf GB'\"",$once);

# reset the db and dbm configuration before we start
&dodb_conn($dbname,"db2 reset database configuration for $dbname;  \
              db2 alter bufferpool ibmdefaultbp size -1;           \
              db2 grant connect on database to public;             \
              db2 grant dbadm on database to $dbname;              \
              db2 commit",$all_ln);
&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);
}
&dodb_noconn("db2 update db cfg for $dbname using logprimary 100; \
```

```
             db2 update db cfg for $dbname using logsecond 5;    \
             db2 update db cfg for $dbname using logfilsiz 3500; \
             db2 update db cfg for $dbname using logbufsz
128",$all_ln);

# 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";
  &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");
}

# create the tables
&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 file
    if ( $product eq "pe" )
    {
       print "update files for pe v1.2 style are build through bhatta's
scripts\n";
    }
    else
    {
        $rc = system("perl buildupdatefiles");
        if ( $rc != 0 )
        {
            die "buildupdatefiles failed rc=$rc\n";
        }
    }
}
&outtime("*** Start Load Timing now - starting to create tables");
&dodb2file($dbname,"$ddlpath${delim}$ddl",$once);

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


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

  if ( $extraindex ne "no" )
  {
      # use this additional file for indices
     &dodb2file($dbname,"$ddlpath${delim}$extraindex",$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
  $rc = system("perl genloaduni $qual");
  if ( $rc != 0 )
  {
      die "genloaduni failed rc = $rc\n";
  }
}
elsif ( $mode eq "mln" )
{
  &outtime("*** Starting the dbgen, split, sort and load");
    # call the appropriate dbgen/split/(sort)/load for mln
#  $rc = system("perl genloadmln $qual");
#  if ( $rc <= 0 )
#  {
#      die "genloadmln failed rc = $rc\n";
#  }
  if ( $qual eq "QUAL" )
  {
    $rc = system("perl custom${delim}doloadQ.ksh");
  }
  else
  {
    $rc = system("perl custom${delim}100GB${delim}doload.ksh");
  }
  if ( $rc != 0 )
  {
      die "doload for $dbname failed rc = $rc\n";
  }
}
elsif ( $mode eq "mpp" )
{
  &outtime("*** Starting the dbgen, split, sort and load");
    # call the appropriate dbgen/split/(sort)/load for mpp
  print "mpp load not well tested yet!\n";
  if ( $qual eq "QUAL" )
  {
    print "must tailor for loading qualification database...to be
completed\n";
  }
  else
  {
    $rc = system("perl custom${delim}MPP${delim}doload.ksh");
  }
  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";
}

# if indices haven't been created, do so now
if ( $earlyindex ne "yes" )
{
  &dodb_noconn("db2 update db cfg for $dbname using buffpage $buffpage;
\
                db2 update db cfg for $dbname using sortheap $sortheap;
\
                db2 update db cfg for $dbname using num_iocleaners
$ioclnrs; \
                db2 update db cfg for $dbname using num_ioservers
$ioservers; \
                db2 update db cfg for $dbname using chngpgs_thresh
$chngpgs",
               $all_ln);
  &dodb_noconn("db2 update dbm cfg using sheapthres $sheapthres",$once);
  $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("*** Create index started");
  &dodb2file($dbname,"$ddlpath${delim}$indexddl",$once);

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

# if statistics not gathered on the load, run runstats  (we have to run
the
# stats at the same time whether it be both during load, or after load)
if (( $loadstats eq "no" ) || ( $earlyindex eq "no"))
{
    # if loadstats not gathered, then index stats not gathered either.
  &outtime("*** Runstats started");
  &dodb2file($dbname,"$ddlpath${delim}$runstats",$once);
  &outtime("*** Runstats 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);

if ( ($mode eq "uni") || ($mode eq "smp"))
{
    &dodb2file(${dbname},"${ddlpath}${delim}${configfile}",$once);
}
elsif ( $mode eq "mln" )
{
    &dodb2file($dbname,"$ddlpath${delim}$configfile",$all_ln);
}


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

if ( $RealAudit eq "yes" )
{
    # stop and restart the database to get config parameters 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";
    }
```

```
    # 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");
    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 checkdb $verifyType");
    system("perl tablesdb $verifyType");
}
# 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="/home/tpcd/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
(( i=0 ));
while [[ $i -le $numchunk ]];
do {
  # create named pipe for this node's chunk of dbgen
    npipe=$dsspath"/partsupp.$i.tbl"
    rm $npipe
    mkfifo $npipe
    DSS_OPTIONS="-F -s "$TPCD_SF" -T S -S $i"
    #print "options=$DSS_OPTIONS";
    $DSS_DBGEN/dbgen $DSS_OPTIONS &

    # run the autosplitter on the file
    db2split -c partsupp.cfg -i $npipe -o $dsspath/partsupp.$i
    (( i=$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=0 ));
while [[ $i -le $numchunk ]];
do {
# create named pipe for this node's chunk of dbgen
    npipe=$dsspath"/order.tbl"
    rm $npipe
    mkfifo $npipe
    DSS_OPTIONS="-F -s "$TPCD_SF" -T O -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=0 ));
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 sort.ksh

```
#! /bin/ksh
# input is the tablename

tbl=$1;
typeset -i ln=0;
typeset -Z5 lnZ5;

if [[ "$tbl" = lineitem ]];
#then colpos="+10";
then colpos="-k11,11r -k6,6nr";
elif [[ "$tbl" = orders ]];
#then colpos="+4";
then colpos="-k5,5 -k1,1";
else { print "I don't recognise $tbl!!"; exit 3; };
fi;


while (( $ln <= 3 ))
do
  lnZ5=$ln;
  dstFile="/dbgen_data/${tbl}.sort.${lnZ5}";
  srtErrDir="/tpcdjfs/sorterr";
  srcFile="/sort_data/${tbl}.*.${lnZ5}";
  sortout="/sort_data/${tbl}.${lnZ5}.sorterr";
  rm $dstFile
  cat ~/tools/custom/${tbl}.HEADER.${lnZ5} > $dstFile
```

```
  smrtsort -y1500000 -r -A -t\| $colpos -T $srtErrDir $srcFile | split
-l 15000000 - $dstFile;
#  sort -y1500000 -r -A -t\| $colpos -T $srtErrDir $srcFile >> $dstFile
2>$srtErrDir/${tbl}.3gb.${lnZ5}.sorterr;
#  smrtsort -y1500000 -r -A -t\| $colpos -T /tpcdjfs/sorterr
/tpcddata/${tbl}.?.${lnZ5} | split -l 15000000 -
/tpcdjfs/lineitem.${lnZ5}


  (( ln=$ln + 1 ));
done
```

B.4 doload.ksh

```
#! /bin/ksh
#    loadeverything
echo "loading customer at "`date`
custom/100GB/loadcust.ksh
echo "loading supplier at "`date`
custom/100GB/loadsupp.ksh
echo "loading part at "`date`
custom/100GB/loadpart.ksh
echo "loading partsupp at "`date`
custom/100GB/loadps.ksh
echo "loading orders at "`date`
custom/100GB/loadord.ksh
echo "loading lineitem at "`date`
custom/100GB/loadline.ksh

echo "loading nation and region at "`date`

db2 connect to tpcd;
db2 "load from /home/haider/HEADERS/nation.tbl of del modified by
coldel| fastparse noheader replace into TPCD.nation nonrecoverable";
db2 commit work;
db2 "load from /home/haider/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.5 loadline.ksh

```
#! /bin/ksh
#    my load lineitem
RAHOSTFILE=/home/haider/sqllib/db2nodes.cfg db2_all '||"]}typeset -i
ln=##;typeset -Z5 LN5=$ln; path="/dbgen_data"; db2 connect to
tpcd;str="db2 \"load from $path/lineitem.SORT.${LN5}aa,
$path/lineitem.SORT.${LN5}ab, $path/lineitem.SORT.${LN5}ac,
$path/lineitem.SORT.${LN5}ad, $path/lineitem.SORT.${LN5}ae,
$path/lineitem.SORT.${LN5}af, $path/lineitem.SORT.${LN5}ag,
$path/lineitem.SORT.${LN5}ah, $path/lineitem.SORT.${LN5}ai,
$path/lineitem.SORT.${LN5}aj, $path/lineitem.SORT.${LN5}ak of del
modified by coldel| fastparse dumpfile /tmp/haider/dmpline${ln}
messages /tmp/haider/msgline${ln} remote file /tmp/haider/rmtline${ln}
replace into TPCD.lineitem nonrecoverable\" ";print -- "$str";eval
"$str";db2 connect reset;db2 terminate'
```

B.6 loadpart.ksh

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

B.7 loadord.ksh

```
#! /bin/ksh
#    my load orders
RAHOSTFILE=/home/haider/sqllib/db2nodes.cfg db2_all '||"]}typeset -i
ln=##;typeset -Z5 LN5=$ln;db2 connect to tpcd;str="db2 \"load from
/dbgen_data/orders.SORT.${LN5}aa, /dbgen_data/orders.SORT.${LN5}ab,
/dbgen_data/orders.SORT.${LN5}ac of del modified by coldel| fastparse
dumpfile /tmp/haider/dmpord${ln} messages /tmp/haider/msgord${ln} remote
file /tmp/haider/rmtord${ln} replace into TPCD.orders nonrecoverable\"
";print -- "$str";eval "$str";db2 connect reset;db2 terminate'
```

B.8 loadps.ksh

```
#! /bin/ksh
#    my load partsupp
RAHOSTFILE=/home/haider/sqllib/db2nodes.cfg db2_all '||"]}typeset -i
ln=##;typeset -Z5 LN5=$ln;db2 connect to tpcd;str="db2 \"load from
/dbgen_data/partsupp.1.${LN5},/dbgen_data/partsupp.2.${LN5},/dbgen_data/
partsupp.3.${LN5},/dbgen_data/partsupp.4.${LN5},/dbgen_data/partsupp.5.$
{LN5},/dbgen_data/partsupp.6.${LN5},/dbgen_data/partsupp.7.${LN5},/dbgen
_data/partsupp.8.${LN5},/dbgen_data/partsupp.9.${LN5},/dbgen_data/partsu
pp.10.${LN5} of del modified by coldel| fastparse replace into
TPCD.partsupp nonrecoverable\" </dev/null >/tmp/haider/loadps.$LN5
2>&1";print -- "$str";eval "$str";db2 connect reset;db2 terminate'
```

B.9 loadcust.ksh

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

B.10 loadsupp.ksh

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

B.11 db2nodes.cfg

```
0 eastwood 0
1 eastwood 1
2 eastwood 2
3 eastwood 3
```

B.12 dss.dbconfig100.eastwoodmln

```
update db cfg for tpcd using buffpage 125000;
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 21;
update db cfg for tpcd using num_iocleaners 4;
--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 100;
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         120;
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 dbm cfg using sheapthres 37000;
update dbm cfg using numdb 1;
update database  manager configuration using fcm_num_buffers 4096;
update database manager configuration using fcm_num_rqb      2048;
update database manager configuration using cpuspeed -1;
update database manager configuration using comm_bandwidth 0.2;
update database manager configuration using num_poolagents 0;
update database manager configuration using max_querydegree any;
update database manager configuration using intra_parallel no;
update dbm cfg using diaglevel 0;
```

B.13 dss.ddl100GB.tbsp.eastwood.mln

```
--CONNECT TO TPCD;

DROP TABLESPACE TPCDTEMP;

-- create the dms managed tablespace
-- 16 lvs of 3072MB each on 16 disks
-- each lv has 201 PPs of size 16MB ==> 823296 pages
-- Total pages in temp tablespace: 13172736 ==> ~48 GB
CREATE TEMPORARY TABLESPACE TPCDTEMP1 MANAGED BY database
  USING (
        device '/dev/rlv.2.A1' 823296,
        device '/dev/rlv.6.A1' 823296,
        device '/dev/rlv.10.A1' 823296,
        device '/dev/rlv.14.A1' 823296) on NODE (0)
  USING (
        device '/dev/rlv.3.A1' 823296,
        device '/dev/rlv.7.A1' 823296,
        device '/dev/rlv.11.A1' 823296,
        device '/dev/rlv.15.A1' 823296) on NODE (1)
  USING (
        device '/dev/rlv.4.A1' 823296,
        device '/dev/rlv.8.A1' 823296,
        device '/dev/rlv.12.A1' 823296,
        device '/dev/rlv.16.A1' 823296) on NODE (2)
  USING (
        device '/dev/rlv.5.A1' 823296,
        device '/dev/rlv.9.A1' 823296,
        device '/dev/rlv.13.A1' 823296,
        device '/dev/rlv.17.A1' 823296) on NODE (3)
         EXTENTSIZE 32 PREFETCHSIZE 128;

COMMIT WORK;


DROP TABLESPACE TEMPSPACE1;

-- create the dms managed tablespace
-- 80 lvs of 2144MB each on 80 disks
-- each lv has 134 PPs of size 16MB ==> 548864 pages
-- Total pages in data tablespace: 43909120 ==> ~160 GB
```

```
CREATE TABLESPACE TPCDDATA MANAGED BY DATABASE
  USING (
       device '/dev/rlv.18.B1' 548864,
       device '/dev/rlv.22.B1' 548864,
       device '/dev/rlv.26.B1' 548864,
       device '/dev/rlv.30.B1' 548864,
       device '/dev/rlv.34.B1' 548864,
       device '/dev/rlv.38.B1' 548864,
       device '/dev/rlv.42.B1' 548864,
       device '/dev/rlv.46.B1' 548864,
       device '/dev/rlv.50.B1' 548864,
       device '/dev/rlv.54.B1' 548864,
       device '/dev/rlv.58.B1' 548864,
       device '/dev/rlv.62.B1' 548864,
       device '/dev/rlv.66.B1' 548864,
       device '/dev/rlv.70.B1' 548864,
       device '/dev/rlv.74.B1' 548864,
       device '/dev/rlv.78.B1' 548864,
       device '/dev/rlv.hdisk87.B1' 548864,
       device '/dev/rlv.hdisk94.B1' 548864,
       device '/dev/rlv.hdisk98.B1' 548864,
       device '/dev/rlv.hdisk105.B1' 548864  ) on NODE (0)
  USING (
       device '/dev/rlv.19.B1' 548864,
       device '/dev/rlv.23.B1' 548864,
       device '/dev/rlv.27.B1' 548864,
       device '/dev/rlv.31.B1' 548864,
       device '/dev/rlv.35.B1' 548864,
       device '/dev/rlv.39.B1' 548864,
       device '/dev/rlv.43.B1' 548864,
       device '/dev/rlv.47.B1' 548864,
       device '/dev/rlv.51.B1' 548864,
       device '/dev/rlv.55.B1' 548864,
       device '/dev/rlv.59.B1' 548864,
       device '/dev/rlv.63.B1' 548864,
       device '/dev/rlv.67.B1' 548864,
       device '/dev/rlv.71.B1' 548864,
       device '/dev/rlv.75.B1' 548864,
       device '/dev/rlv.79.B1' 548864,
       device '/dev/rlv.hdisk88.B1' 548864,
       device '/dev/rlv.hdisk95.B1' 548864,
       device '/dev/rlv.hdisk99.B1' 548864,
       device '/dev/rlv.hdisk107.B1' 548864 ) on NODE (1)
  USING (
       device '/dev/rlv.20.B1' 548864,
       device '/dev/rlv.24.B1' 548864,
       device '/dev/rlv.28.B1' 548864,
       device '/dev/rlv.32.B1' 548864,
       device '/dev/rlv.36.B1' 548864,
       device '/dev/rlv.40.B1' 548864,
       device '/dev/rlv.44.B1' 548864,
       device '/dev/rlv.48.B1' 548864,
       device '/dev/rlv.52.B1' 548864,
       device '/dev/rlv.56.B1' 548864,
       device '/dev/rlv.60.B1' 548864,
       device '/dev/rlv.64.B1' 548864,
       device '/dev/rlv.68.B1' 548864,
       device '/dev/rlv.72.B1' 548864,
       device '/dev/rlv.76.B1' 548864,
       device '/dev/rlv.80.B1' 548864,
       device '/dev/rlv.hdisk89.B1' 548864,
       device '/dev/rlv.hdisk96.B1' 548864,
       device '/dev/rlv.hdisk103.B1' 548864,
       device '/dev/rlv.hdisk110.B1' 548864) on NODE (2)
  USING (
       device '/dev/rlv.21.B1' 548864,
       device '/dev/rlv.25.B1' 548864,
       device '/dev/rlv.29.B1' 548864,
       device '/dev/rlv.33.B1' 548864,
       device '/dev/rlv.37.B1' 548864,
       device '/dev/rlv.41.B1' 548864,
       device '/dev/rlv.45.B1' 548864,
       device '/dev/rlv.49.B1' 548864,
       device '/dev/rlv.53.B1' 548864,
       device '/dev/rlv.57.B1' 548864,
       device '/dev/rlv.61.B1' 548864,
       device '/dev/rlv.65.B1' 548864,
       device '/dev/rlv.69.B1' 548864,
       device '/dev/rlv.73.B1' 548864,
       device '/dev/rlv.77.B1' 548864,
       device '/dev/rlv.81.B1' 548864,
       device '/dev/rlv.hdisk91.B1' 548864,
       device '/dev/rlv.hdisk97.B1' 548864,
       device '/dev/rlv.hdisk104.B1' 548864,
       device '/dev/rlv.hdisk111.B1' 548864)  on NODE (3)
        EXTENTSIZE 32 PREFETCHSIZE 640;

COMMIT WORK;

-- create the dms managed tablespace for indices
-- 80 lvs of 2144MB each on 80 disks
-- each lv has 134 PPs of size 16MB ==> 548864 pages
-- Total pages in index tablespace: 43909120 ==> ~160 GB
CREATE TABLESPACE TPCDINDEX MANAGED BY DATABASE
  USING (
       device '/dev/rlv.18.C1' 548864,
       device '/dev/rlv.22.C1' 548864,
       device '/dev/rlv.26.C1' 548864,
       device '/dev/rlv.30.C1' 548864,
       device '/dev/rlv.34.C1' 548864,
       device '/dev/rlv.38.C1' 548864,
       device '/dev/rlv.42.C1' 548864,
       device '/dev/rlv.46.C1' 548864,
       device '/dev/rlv.50.C1' 548864,
       device '/dev/rlv.54.C1' 548864,
       device '/dev/rlv.58.C1' 548864,
       device '/dev/rlv.62.C1' 548864,
       device '/dev/rlv.66.C1' 548864,
       device '/dev/rlv.70.C1' 548864,
       device '/dev/rlv.74.C1' 548864,
       device '/dev/rlv.78.C1' 548864,
       device '/dev/rlv.hdisk87.C1' 548864,
       device '/dev/rlv.hdisk94.C1' 548864,
       device '/dev/rlv.hdisk98.C1' 548864,
       device '/dev/rlv.hdisk105.C1' 548864  ) on NODE (0)
  USING (
       device '/dev/rlv.19.C1' 548864,
       device '/dev/rlv.23.C1' 548864,
       device '/dev/rlv.27.C1' 548864,
       device '/dev/rlv.31.C1' 548864,
       device '/dev/rlv.35.C1' 548864,
       device '/dev/rlv.39.C1' 548864,
       device '/dev/rlv.43.C1' 548864,
       device '/dev/rlv.47.C1' 548864,
       device '/dev/rlv.51.C1' 548864,
       device '/dev/rlv.55.C1' 548864,
       device '/dev/rlv.59.C1' 548864,
       device '/dev/rlv.63.C1' 548864,
       device '/dev/rlv.67.C1' 548864,
       device '/dev/rlv.71.C1' 548864,
       device '/dev/rlv.75.C1' 548864,
       device '/dev/rlv.79.C1' 548864,
       device '/dev/rlv.hdisk88.C1' 548864,
       device '/dev/rlv.hdisk95.C1' 548864,
       device '/dev/rlv.hdisk99.C1' 548864,
       device '/dev/rlv.hdisk107.C1' 548864 ) on NODE (1)
  USING (
       device '/dev/rlv.20.C1' 548864,
       device '/dev/rlv.24.C1' 548864,
       device '/dev/rlv.28.C1' 548864,
       device '/dev/rlv.32.C1' 548864,
       device '/dev/rlv.36.C1' 548864,
       device '/dev/rlv.40.C1' 548864,
       device '/dev/rlv.44.C1' 548864,
       device '/dev/rlv.48.C1' 548864,
       device '/dev/rlv.52.C1' 548864,
       device '/dev/rlv.56.C1' 548864,
       device '/dev/rlv.60.C1' 548864,
       device '/dev/rlv.64.C1' 548864,
       device '/dev/rlv.68.C1' 548864,
       device '/dev/rlv.72.C1' 548864,
       device '/dev/rlv.76.C1' 548864,
       device '/dev/rlv.80.C1' 548864,
       device '/dev/rlv.hdisk89.C1' 548864,
       device '/dev/rlv.hdisk96.C1' 548864,
       device '/dev/rlv.hdisk103.C1' 548864,
       device '/dev/rlv.hdisk110.C1' 548864) on NODE (2)
  USING (
       device '/dev/rlv.21.C1' 548864,
       device '/dev/rlv.25.C1' 548864,
       device '/dev/rlv.29.C1' 548864,
       device '/dev/rlv.33.C1' 548864,
       device '/dev/rlv.37.C1' 548864,
       device '/dev/rlv.41.C1' 548864,
       device '/dev/rlv.45.C1' 548864,
       device '/dev/rlv.49.C1' 548864,
       device '/dev/rlv.53.C1' 548864,
       device '/dev/rlv.57.C1' 548864,
       device '/dev/rlv.61.C1' 548864,
       device '/dev/rlv.65.C1' 548864,
       device '/dev/rlv.69.C1' 548864,
       device '/dev/rlv.73.C1' 548864,
       device '/dev/rlv.77.C1' 548864,
       device '/dev/rlv.81.C1' 548864,
       device '/dev/rlv.hdisk91.C1' 548864,
       device '/dev/rlv.hdisk97.C1' 548864,
       device '/dev/rlv.hdisk104.C1' 548864,
       device '/dev/rlv.hdisk111.C1' 548864)  on NODE (3)
        EXTENTSIZE 32 PREFETCHSIZE 640;

COMMIT WORK;

create nodegroup NODE0 on NODE (0);

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

commit work;

alter bufferpool ibmdefaultbp size -1;
commit work;

create tablespace UFTEMP1 managed by system
     using ('/db2log/UF100GB/uf1_0') on node (0)
     using ('/db2log/UF100GB/uf1_1') on node (1)
     using ('/db2log/UF100GB/uf1_2') on node (2)
     using ('/db2log/UF100GB/uf1_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP2 managed by system
     using ('/db2log/UF100GB/uf2_0') on node (0)
     using ('/db2log/UF100GB/uf2_1') on node (1)
     using ('/db2log/UF100GB/uf2_2') on node (2)
     using ('/db2log/UF100GB/uf2_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP3 managed by system
     using ('/db2log/UF100GB/uf3_0') on node (0)
     using ('/db2log/UF100GB/uf3_1') on node (1)
     using ('/db2log/UF100GB/uf3_2') on node (2)
     using ('/db2log/UF100GB/uf3_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP4 managed by system
     using ('/db2log/UF100GB/uf4_0') on node (0)
     using ('/db2log/UF100GB/uf4_1') on node (1)
     using ('/db2log/UF100GB/uf4_2') on node (2)
     using ('/db2log/UF100GB/uf4_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;

create tablespace UFTEMP5 managed by system
     using ('/db2log/UF100GB/uf5_0') on node (0)
     using ('/db2log/UF100GB/uf5_1') on node (1)
     using ('/db2log/UF100GB/uf5_2') on node (2)
     using ('/db2log/UF100GB/uf5_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP6 managed by system
     using ('/db2log/UF100GB/uf6_0') on node (0)
     using ('/db2log/UF100GB/uf6_1') on node (1)
     using ('/db2log/UF100GB/uf6_2') on node (2)
     using ('/db2log/UF100GB/uf6_3') on node (3)
```

```
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP7 managed by system
     using ('/db2log/UF100GB/uf7_0') on node (0)
     using ('/db2log/UF100GB/uf7_1') on node (1)
     using ('/db2log/UF100GB/uf7_2') on node (2)
     using ('/db2log/UF100GB/uf7_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP8 managed by system
     using ('/db2log/UF100GB/uf8_0') on node (0)
     using ('/db2log/UF100GB/uf8_1') on node (1)
     using ('/db2log/UF100GB/uf8_2') on node (2)
     using ('/db2log/UF100GB/uf8_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;

create tablespace UFTEMP9 managed by system
     using ('/db2log/UF100GB/uf9_0') on node (0)
     using ('/db2log/UF100GB/uf9_1') on node (1)
     using ('/db2log/UF100GB/uf9_2') on node (2)
     using ('/db2log/UF100GB/uf9_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP10 managed by system
     using ('/db2log/UF100GB/uf10_0') on node (0)
     using ('/db2log/UF100GB/uf10_1') on node (1)
     using ('/db2log/UF100GB/uf10_2') on node (2)
     using ('/db2log/UF100GB/uf10_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP11 managed by system
     using ('/db2log/UF100GB/uf11_0') on node (0)
     using ('/db2log/UF100GB/uf11_1') on node (1)
     using ('/db2log/UF100GB/uf11_2') on node (2)
     using ('/db2log/UF100GB/uf11_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP12 managed by system
     using ('/db2log/UF100GB/uf12_0') on node (0)
     using ('/db2log/UF100GB/uf12_1') on node (1)
     using ('/db2log/UF100GB/uf12_2') on node (2)
     using ('/db2log/UF100GB/uf12_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP13 managed by system
     using ('/db2log/UF100GB/uf13_0') on node (0)
     using ('/db2log/UF100GB/uf13_1') on node (1)
     using ('/db2log/UF100GB/uf13_2') on node (2)
     using ('/db2log/UF100GB/uf13_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP14 managed by system
     using ('/db2log/UF100GB/uf14_0') on node (0)
     using ('/db2log/UF100GB/uf14_1') on node (1)
     using ('/db2log/UF100GB/uf14_2') on node (2)
     using ('/db2log/UF100GB/uf14_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP15 managed by system
     using ('/db2log/UF100GB/uf15_0') on node (0)
     using ('/db2log/UF100GB/uf15_1') on node (1)
     using ('/db2log/UF100GB/uf15_2') on node (2)
     using ('/db2log/UF100GB/uf15_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP16 managed by system
     using ('/db2log/UF100GB/uf16_0') on node (0)
     using ('/db2log/UF100GB/uf16_1') on node (1)
     using ('/db2log/UF100GB/uf16_2') on node (2)
     using ('/db2log/UF100GB/uf16_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP17 managed by system
     using ('/db2log/UF100GB/uf17_0') on node (0)
     using ('/db2log/UF100GB/uf17_1') on node (1)
     using ('/db2log/UF100GB/uf17_2') on node (2)
     using ('/db2log/UF100GB/uf17_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP18 managed by system
     using ('/db2log/UF100GB/uf18_0') on node (0)
     using ('/db2log/UF100GB/uf18_1') on node (1)
     using ('/db2log/UF100GB/uf18_2') on node (2)
     using ('/db2log/UF100GB/uf18_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP19 managed by system
     using ('/db2log/UF100GB/uf19_0') on node (0)
     using ('/db2log/UF100GB/uf19_1') on node (1)
     using ('/db2log/UF100GB/uf19_2') on node (2)
     using ('/db2log/UF100GB/uf19_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP20 managed by system
     using ('/db2log/UF100GB/uf20_0') on node (0)
     using ('/db2log/UF100GB/uf20_1') on node (1)
     using ('/db2log/UF100GB/uf20_2') on node (2)
     using ('/db2log/UF100GB/uf20_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;

create tablespace UFTEMP21 managed by system
     using ('/db2log/UF100GB/uf21_0') on node (0)
     using ('/db2log/UF100GB/uf21_1') on node (1)
     using ('/db2log/UF100GB/uf21_2') on node (2)
     using ('/db2log/UF100GB/uf21_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP22 managed by system
     using ('/db2log/UF100GB/uf22_0') on node (0)
     using ('/db2log/UF100GB/uf22_1') on node (1)
     using ('/db2log/UF100GB/uf22_2') on node (2)
     using ('/db2log/UF100GB/uf22_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP23 managed by system
     using ('/db2log/UF100GB/uf23_0') on node (0)
     using ('/db2log/UF100GB/uf23_1') on node (1)
     using ('/db2log/UF100GB/uf23_2') on node (2)
     using ('/db2log/UF100GB/uf23_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;
create tablespace UFTEMP24 managed by system
     using ('/db2log/UF100GB/uf24_0') on node (0)
     using ('/db2log/UF100GB/uf24_1') on node (1)
     using ('/db2log/UF100GB/uf24_2') on node (2)
     using ('/db2log/UF100GB/uf24_3') on node (3)
    extentsize 32 prefetchsize 32;
commit work;

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

B.14 dss.ddl100GB.tbl.eastwood.mln

```
CREATE TABLE TPCD.NATION  ( N_NATIONKEY  INTEGER NOT NULL,
                            N_NAME       CHAR(25) NOT NULL,
                            N_REGIONKEY  INTEGER NOT NULL,
                            N_COMMENT    VARCHAR(152))
                            IN TSNODE_1;

CREATE TABLE TPCD.REGION  ( R_REGIONKEY  INTEGER NOT NULL,
                            R_NAME       CHAR(25) NOT NULL,
                            R_COMMENT    VARCHAR(152))
                            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 )
        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)
         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 )
        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)
        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)
        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)
        IN TPCDDATA INDEX IN TPCDINDEX
                 PARTITIONING KEY(L_ORDERKEY) USING HASHING;

COMMIT WORK;
```

B.15 dss.index.prune.include

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

```
!date;
CREATE UNIQUE INDEX "TPCD    "."C_NAT_CKEY_REG" ON "TPCD    "."CUSTOMER"
                ("C_NATIONKEY" ASC,
                 "C_CUSTKEY" ASC)
                PCTFREE 0 ;
commit work;

!date;
create unique index tpcd.c_ck on tpcd.customer (c_custkey) pctfree 0;
commit;

!date;
CREATE INDEX "TPCD    "."L_OKCDRD" ON "TPCD    "."LINEITEM"
                ("L_ORDERKEY" ASC,
                 "L_COMMITDATE" ASC,
                 "L_RECEIPTDATE" ASC)
                PCTFREE 1 ;
commit work;

!date;
CREATE INDEX TPCD.L_RDCDSDOKSM ON TPCD.LINEITEM
                (L_RECEIPTDATE ASC,
                 L_COMMITDATE ASC,
                 L_SHIPDATE ASC,
                 L_ORDERKEY ASC,
                 L_SHIPMODE ASC )
               PCTFREE 2;
commit work;

!date;
CREATE INDEX "TPCD    "."L_SDDSLQEPSKPK" ON "TPCD    "."LINEITEM"
                ("L_SHIPDATE" ASC,
                 "L_DISCOUNT" ASC,
                 "L_QUANTITY" DESC,
                 "L_EXTENDEDPRICE" ASC,
                 "L_SUPPKEY" ASC,
                 "L_PARTKEY" ASC)
                PCTFREE 3 ;
commit work;

!date;
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 2 ;
commit work;

!date;
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 3 ;
commit work;


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

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

!date;
CREATE UNIQUE INDEX "TPCD    "."N_REGKEYNATKEYNAM" ON "TPCD
"."NATION"
                ("N_REGIONKEY" ASC,
                 "N_NATIONKEY" ASC,
                 "N_NAME" ASC)
                PCTFREE 0 ;
commit work;

!date;
CREATE UNIQUE INDEX "TPCD  "."O_CK_OD_OK_SP" ON "TPCD    "."ORDERS"
                ("O_CUSTKEY" ASC,
                 "O_ORDERDATE" ASC,
                 "O_ORDERKEY" ASC,
                 "O_SHIPPRIORITY" ASC)
                PCTFREE 2 ;
commit work;

!date;
CREATE UNIQUE INDEX "TPCD  "."O_OK_OD_OP" ON "TPCD    "."ORDERS"
                ("O_ORDERKEY" ASC) include(
                 "O_ORDERDATE" ASC,
                 "O_ORDERPRIORITY" ASC)
                PCTFREE 2 ;
commit work;

!date;
CREATE INDEX "TPCD    "."O_CLERK" ON "TPCD    "."ORDERS"
                ("O_CLERK" ASC)
                PCTFREE 2 ;
commit work;

!date;
CREATE UNIQUE INDEX "TPCD    "."O_OD_OK_OP_CK" ON "TPCD    "."ORDERS"
                ("O_ORDERDATE" ASC,
                 "O_ORDERKEY" ASC,
                 "O_ORDERPRIORITY" ASC,
                 "O_CUSTKEY" ASC)
                PCTFREE 2 ;
commit work;
```

```
!date;
CREATE UNIQUE INDEX "TPCD  "."P_TP_PK" ON "TPCD    "."PART"
                ("P_TYPE" ASC,
                 "P_PARTKEY" ASC)
                PCTFREE 0 ;
commit work;

!date;
CREATE UNIQUE INDEX "TPCD    "."PK_P_PARTKEY" ON "TPCD    "."PART"
                ("P_PARTKEY" ASC)
                PCTFREE 0 ;
commit work;

!date;
CREATE INDEX "TPCD    "."P_SIZE" ON "TPCD    "."PART"
                ("P_SIZE" ASC)
                PCTFREE 0 ;
commit work;

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

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

!date;
CREATE UNIQUE INDEX "TPCD    "."UXP_SZTYPKMF" ON "TPCD    "."PART"
                ("P_SIZE" ASC,
                 "P_TYPE" ASC,
                 "P_PARTKEY" ASC,
                 "P_MFGR" ASC)
                PCTFREE 0 ;
commit work;

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

!date;
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;

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

!date;
CREATE UNIQUE INDEX "TPCD    "."PK_S_SUPPKEY" ON "TPCD    "."SUPPLIER"
                ("S_SUPPKEY" ASC)
                PCTFREE 0 ;
commit work;

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

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

!date;
```

B.16 dss.runstats

```
--CONNECT TO TPCD;

RUNSTATS ON TABLE TPCD.NATION    AND DETAILED INDICES ALL;
RUNSTATS ON TABLE TPCD.REGION    AND DETAILED INDICES ALL;
RUNSTATS ON TABLE TPCD.SUPPLIER  AND DETAILED INDICES ALL;
RUNSTATS ON TABLE TPCD.PART      AND DETAILED INDICES ALL;
RUNSTATS ON TABLE TPCD.PARTSUPP  AND DETAILED INDICES ALL;
RUNSTATS ON TABLE TPCD.CUSTOMER  AND DETAILED INDICES ALL;
RUNSTATS ON TABLE TPCD.ORDERS    AND DETAILED INDICES ALL;
RUNSTATS ON TABLE TPCD.LINEITEM  AND DETAILED INDICES ALL;

COMMIT WORK;

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

B.17 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_AUDIT_DIR=/home/haider/tpcd        # top level directory of tar file
for
                                        # all the tpcd scripts
TPCD_AUDITDIR=/home/haider/tpcd         # oopss...need 2 for the time
being for
                                        # acid tests
                                        # 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=4                        # number of logical nodes per
                                         physical node
TPCD_SF=100                             # size of the database
                                         (1=1GB,...)
TPCD_BUILD_DATABASE=NULL                # whether to build a new database
                                        # NOT CURRENTLY USED
TPCD_DBPATH=/home/haider/tpcd100     # path for database (defaults to
                                         home)
TPCD_DDLPATH=/home/haider/tpcd/ddl/vanilla  # path for all ddl files
TPCD_TBSP_DDL=dss.ddl100GB.tbsp.eastwood.mln     # ddl file for
                                                  tablespaces
TPCD_DDL=dss.ddl100GB.tbl.eastwood.mln              # ddl file for
                                                    tables
TPCD_QUAL_TBSP_DDL=                     # ddl file for qual tablespaces
TPCD_QUAL_DDL=                          # ddl file for qual database
                                        # tablespace and table should be
                                         identical
                                        # identical to regular ddl except
                                        # container names
TPCD_INDEXDDL=dss.index.prune.include   # ddl file for indices
TPCD_EXTRAINDEX=                        # use this additional file when
                                        # creating indices
TPCD_RUNSTATS=dss.runstats              # ddl file for runstats
TPCD_DBGEN=/home/haider/tpcd/appendix/dbgen # path name to data
generation code
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/haider/tpcd/tailor   # path name for the directory
                                        # that contains any tailoring
                                        # specific files
                                        # or code. eg split config files
TPCD_EARLYINDEX=no                     # create indices before the load
                                       # LOAD specific parameters follow:
TPCD_TEMP=/tmp/haider                   # path for LOAD temp files
                                        # defaults to
                                        # /u/<instance>/sqllib/tmp
TPCD_SORTBUF=4096                       # sortbuf size for LOAD
TPCD_LOAD_PARALLELISM=0                 # degree of parallelism to use on
                                        # load
                                        # 0 = use the "intelligent
                                        # default" that
                                        # the load will chose at run time
TPCD_LOADSTATS=no                       # gather statistics during load
                                        # ignored if EARLYINDEX is not
                                        # set due to runstats limitation
TPCD_COPY_DIR=/dev/null                 # directory where copy image is
                                        # created on load command
                                        # CURRENTLY UNUSED
TPCD_FASTPARSE=yes                      # use fastparse on load

                                     # Backup specific parameters follow:
TPCD_BACKUP_DIR=/sort_data/backupdir  # directory where backup files are
                                      # placed
TPCD_LOG_DIR=/db2log/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 - whether to turn
                                        # LOG_RETAIN on
                                        # i.e. are backups needed to be
                                        # taken

                                        # CONFIG specific parameters
TPCD_CONFIGFILE=dss.dbconfig100.eastwoodmln    # name of configuration
                                        # file in ddl path
TPCD_DBM_CONFIG=NULL                    # not used yet  Will be used when
                                        # dbm cfg set up separately from
                                        # db cfg
TPCD_QUALCONFIGFILE=NULL                # name of configuration file in
                                        # ddl path for qualification
                                        # database
TPCD_DBM_QUALCONFIG=NULL                # not used yet  Will be used when
                                        # dbm cfg set up separately from
                                        # db cfg
TPCD_MACHINE=eastwood                   # big/medium/small  size of
                                        # machine used to
                                        # determine buffpage,
                                        # sortheap,sheapthres
                                        # and ioservers parms for load,
                                        # create, index and runstats
TPCD_SMPDEGREE=1                        # 1...# of degrees of parallelism
                                        # to run with
TPCD_AGENTPRI=NULL                      # set agentpri to this value
                                        # (default is SYSTEM)
TPCD_ACTIVATE=yes                       # activate the database upon
                                        # build completion

                                        # run specific parameters
TPCD_AUDIT=yes                          # no/yes
                              # no - don't set up qualification db stuff
                              # yes - set up qualification db queries
                              #     - build the update function tables
                              #       and data before we get into the
                              #       timing of the creation of the
                                    #       tables and the load.
TPCD_TMP_DIR=/tmp/haider                 # place to put temp working files
TPCD_QUERY_TEMPLATE_DIR=mln          # 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=TPCDQUAL            # name of qualification database
TPCD_NUMSTREAM=1                     # number of streams for the
                                     # throughput test
TPCD_FLATFILES=/db2_store/100GBUFffiles # where to generate flat files
                                          # for update functions
TPCD_SPLIT_UPDATES=24                # number of portions to split the
                                     # update function into.
TPCD_CONCURRENT_INSERTS=12           # number of concurrent portions to run
TPCD_CONCURRENT_INSERTS_LOAD=8       # number of concurrent loads to run
                                     # for UF1 inserts
TPCD_SPLIT_DELETES=30                # number of portions to split the
                                     # delete function into.
                                     # this variable is only valid in UNI/SMP
                                     # mode.  It is ignored in MLN/MPP mode
TPCD_GEN_UPDATEPAIRS=6               # 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
                                     # yes - generate a seed file base on
                                     #       year/month/day (for audited runs)
                                     # no  - use dbgen's default seeds
TPCD_RUN_ON_MULTIPLE_NODES=NULL # pe 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 during power
                                     # run

TPCD_UFTEMP=UFTEMP                   # base name of tablespace(s) where the
                                     # staging tables for the update
                                     # functions are created
                                     # this name will be used as the
                                     # basename for the tablespaces...eg
                                     # UFTEMP1 UFTEMP2 ....
TPCD_HAVECOMPILER=no                 # rebuild tpcdbatch executable
                                     # yes/no
TPCD_SLEEP=5                         # ?
TPCD_INLISTMAX=400                   # max num of keys to delete at a time
                                     # for UF2, use "default" for default.
```

Appendix C: Query Text and Output

C.1 Query Executable Text for Test Database

Query 1

```
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 1056204987 as a seed to the RNG
--#SET ROWS_OUT -1 ROWS_FETCH -1
-- 'TPCD Pricing Summary Report Query (Q1)';
--#TAG Q1

SELECT
  L_RETURNFLAG,
  L_LINESTATUS,
  SUM(L_QUANTITY) AS SUM_QTY,
  SUM(L_EXTENDEDPRICE) AS SUM_BASE_PRICE,
  SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)) 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') - 93 DAYS
GROUP BY L_RETURNFLAG, L_LINESTATUS
ORDER BY L_RETURNFLAG, L_LINESTATUS;

--#EOBLK
```

Query 2

```
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 1056204987 as a seed to the RNG
--#SET ROWS_OUT -1 ROWS_FETCH -1
--  'TPCD Minimum Cost Supplier (Q2)';
--#TAG Q2

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 = 28
  AND P_TYPE LIKE '%STEEL'
  AND S_NATIONKEY = N_NATIONKEY
  AND N_REGIONKEY = R_REGIONKEY
  AND R_NAME = 'ASIA'
  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 = 'ASIA')
ORDER BY S_ACCTBAL DESC, N_NAME,S_NAME,P_PARTKEY
FETCH FIRST 100 ROWS ONLY;

--#EOBLK
```

Query 3

```
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 1056204987 as a seed to the RNG
--#SET ROWS_OUT -1 ROWS_FETCH -1
--  'TPCD Shipping Priority Query (Q3)';
--#TAG Q3

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 = 'FURNITURE'
  AND C_CUSTKEY = O_CUSTKEY
  AND L_ORDERKEY = O_ORDERKEY
  AND O_ORDERDATE < DATE('1995-03-08')
  AND L_SHIPDATE > DATE('1995-03-08')
GROUP BY L_ORDERKEY, O_ORDERDATE, O_SHIPPRIORITY
ORDER BY REVENUE DESC, O_ORDERDATE
FETCH FIRST 10 ROWS ONLY;

--#EOBLK
```

Query 4

```
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 1056204987 as a seed to the RNG
--#SET ROWS_OUT -1 ROWS_FETCH -1
-- 'TPCD Order Priority Checking Query (Q4)';
--#TAG Q4

SELECT
  O_ORDERPRIORITY,
  COUNT(*) AS ORDER_COUNT
FROM TPCD.ORDERS
WHERE O_ORDERDATE >= DATE('1995-09-01')
  AND O_ORDERDATE < DATE('1995-09-01') + 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;

--#EOBLK
```

Query 5

```
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 1056204987 as a seed to the RNG
--#SET ROWS_OUT -1 ROWS_FETCH -1
-- 'TPCD Local Supplier Volume Query (Q5)';
--#TAG Q5

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;

--#EOBLK
```

Query 6

```
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 1056204987 as a seed to the RNG
--#SET ROWS_OUT -1 ROWS_FETCH -1
-- 'TPCD Forecasting Revenue Change Query (Q6)';
--#TAG Q6

SELECT
  SUM(L_EXTENDEDPRICE*L_DISCOUNT) AS REVENUE
FROM TPCD.LINEITEM
WHERE L_SHIPDATE >= DATE('1995-01-01')
  AND L_SHIPDATE < DATE('1995-01-01') + 1 YEAR
  AND L_DISCOUNT BETWEEN 0.04 - 0.01 AND 0.04 + 0.01
  AND L_QUANTITY < 24;

--#EOBLK
```

Query 7

```
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 1056204987 as a seed to the RNG
--#SET ROWS_OUT -1 ROWS_FETCH -1
-- 'TPCD Volume Shipping Query (Q7)';
--#TAG Q7

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 = 'JORDAN' AND N2.N_NAME = 'FRANCE')
  OR (N1.N_NAME = 'FRANCE' AND N2.N_NAME = 'JORDAN'))
  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;

--#EOBLK
```

Query 8

```
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 1056204987 as a seed to the RNG
--#SET ROWS_OUT -1 ROWS_FETCH -1
```

```
-- 'TPCD Market Share Query (Q08)
--#TAG Q8

SELECT
  YEAR AS YEAR,
   SUM(CASE WHEN NATION = 'JORDAN'
  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 = 'MIDDLE EAST'
  AND S_NATIONKEY = N2.N_NATIONKEY
  AND O_ORDERDATE BETWEEN DATE('1995-01-01')
  AND DATE('1996-12-31')
  AND P_TYPE = 'SMALL PLATED STEEL' ) as ALL_NATIONS
GROUP BY YEAR
ORDER BY YEAR;

--#EOBLK
```

Query 9

```
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 1056204987 as a seed to the RNG
--#SET ROWS_OUT -1 ROWS_FETCH -1
-- 'TPC Product Type Profit Measure Query (Q9)';
--#TAG Q9

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 '%metalic%') AS PROFIT
GROUP BY NATION, YEAR
ORDER BY NATION, YEAR DESC;

--#EOBLK
```

Query 10

```
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 1056204987 as a seed to the RNG
--#SET ROWS_OUT -1 ROWS_FETCH -1
-- 'TPCD Returned Item Reporting Query (Q10)';
--#TAG Q10

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('1994-03-01')
  AND O_ORDERDATE <  DATE('1994-03-01') + 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;
--#EOBLK
```

Query 11

```
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 1056204987 as a seed to the RNG
--#SET ROWS_OUT -1 ROWS_FETCH -1
-- 'TPCD Important Stock Identification QUERY (Q11)';
--#TAG Q11

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 = 'JORDAN'
GROUP BY PS_PARTKEY HAVING SUM(PS_SUPPLYCOST*PS_AVAILQTY) >
(SELECT SUM(PS_SUPPLYCOST*PS_AVAILQTY) * 0.0000010000
FROM TPCD.PARTSUPP, TPCD.SUPPLIER, TPCD.NATION
WHERE PS_SUPPKEY = S_SUPPKEY
  AND S_NATIONKEY = N_NATIONKEY
  AND N_NAME = 'JORDAN')
ORDER BY VALUE DESC;

--#EOBLK
```

Query 12

```
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 1056204987 as a seed to the RNG
--#SET ROWS_OUT -1 ROWS_FETCH -1
-- 'TPCD Shipping Modes and Order Priority Query (Q12)';
--#TAG Q12

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 ('TRUCK','AIR')
  AND L_COMMITDATE < L_RECEIPTDATE
  AND L_SHIPDATE < L_COMMITDATE
  AND L_RECEIPTDATE >= DATE('1995-01-01')
  AND L_RECEIPTDATE < DATE('1995-01-01') + 1 YEAR
GROUP BY L_SHIPMODE
ORDER BY L_SHIPMODE;

--#EOBLK
```

Query 13

```
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 1056204987 as a seed to the RNG
--#SET ROWS_OUT -1 ROWS_FETCH -1
-- 'TPCD Sales Clerk Performance (Q13)';
--#TAG Q13

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#000000550'
  AND L_RETURNFLAG = 'R') AS PERFORMANCE
GROUP BY YEAR
ORDER BY YEAR;

--#EOBLK
```

Query 14

```
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 1056204987 as a seed to the RNG
--#SET ROWS_OUT -1 ROWS_FETCH -1
-- 'TPCD Promotion Effect Query (Q14)';
--#TAG Q14

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('1995-10-01')
  AND L_SHIPDATE <  DATE('1995-10-01') + 1 MONTH;

--#EOBLK
```

Query 15

```
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 1056204987 as a seed to the RNG
--#SET ROWS_OUT -1 ROWS_FETCH -1
-- 'TPCD Top Supplier Query (Q15) Variant C';
--#TAG Q15c

WITH
  REVENUE ( SUPPLIER_NO, TOTAL_REVENUE ) AS
(SELECT
  L_SUPPKEY,
  SUM(L_EXTENDEDPRICE * (1-L_DISCOUNT))
FROM TPCD.LINEITEM
WHERE L_SHIPDATE >= DATE('1995-09-01')
  AND L_SHIPDATE <  DATE('1995-09-01') + 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;



--#EOBLK
```

Query 16

```
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 1056204987 as a seed to the RNG
--#SET ROWS_OUT -1 ROWS_FETCH -1
-- 'TPCD Parts/Supplier Relationship Query (Q16)';
--#TAG Q16

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#32'
  AND P_TYPE NOT LIKE 'MEDIUM POLISHED%'
  AND P_SIZE IN (25,1,39,26,30,28,41,34)
  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;

--#EOBLK
```

Query 17

```
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 1056204987 as a seed to the RNG
--#SET ROWS_OUT -1 ROWS_FETCH -1
-- 'TPCD Small-Quantity-Order Revenue Query (Q17)';
--#TAG Q17

SELECT
  SUM(L_EXTENDEDPRICE)/7.0 AS AVG_YEARLY
FROM TPCD.LINEITEM, TPCD.PART
WHERE P_PARTKEY = L_PARTKEY
  AND P_BRAND = 'Brand#32'
  AND P_CONTAINER = 'MED BAG'
  AND L_QUANTITY <
(SELECT 0.2* AVG(L_QUANTITY)
FROM TPCD.LINEITEM
WHERE L_PARTKEY = P_PARTKEY);
--#EOBLK
```

C.2 Qualification Query Output

Qualification Query 1

```
Start timestamp  12/15/97 17:44:54

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

Tag: Q1       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)) 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                      3773527.000        5320034523.170
     5054630090.592        5257019393.989               25.513
   35968.781               0.050       147907
N            F                       100245.000         141459686.100
      134380852.769         139710306.872               25.625
   36160.451               0.050         3912
N            O                      7467526.000       10522099310.120
     9995443611.633       10395924590.678               25.551
   36002.283               0.050       292262
R            F                      3780003.000        5329934636.320
     5063377096.125        5266493172.501               25.554
   36032.549               0.050       147920


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


Stop timestamp  12/15/97 17:45:00
```

Qualification Query 2

```
Start timestamp  12/15/97 17:45:06

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

Tag: Q2       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
-----------------------------------------------
```

```
Stop timestamp  12/15/97 17:45:07
```

Qualification Query 3

```
Start timestamp  12/15/97 17:45:14

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

Tag: Q3       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
-----------------------------------------------

Stop timestamp  12/15/97 17:45:16
```

Qualification Query 4

```
Start timestamp  12/15/97 17:45:00

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

Tag: Q4       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
-----------------------------------------------

Stop timestamp  12/15/97 17:45:01
```

Qualification Query 5

```
Start timestamp  12/15/97 17:45:16

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

Tag: Q5       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
-----------------------------------------------

Stop timestamp  12/15/97 17:45:21
```

Qualification Query 6

```
Start timestamp  12/15/97 17:45:05

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

Tag: Q6       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
-----------------------------------------------

Stop timestamp  12/15/97 17:45:06
```

Qualification Query 7

```
Start timestamp  12/15/97 17:45:21

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

Tag: Q7       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
-----------------------------------------------

Stop timestamp  12/15/97 17:45:25
```

Qualification Query 8

```
Start timestamp  12/15/97 17:45:09

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

Tag: Q8       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
-----------------------------------------------


Stop timestamp  12/15/97 17:45:11
```

Qualification Query 9

```
Start timestamp  12/15/97 17:45:25

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

Tag: Q9       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
-----------------------------------------------


Stop timestamp  12/15/97 17:45:32
```

Qualification Query 10

```
Start timestamp  12/15/97 17:45:02

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

Tag: Q10      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
RmCM120L24Pkg7PS1zwC1lBCnz4L6i15PkixP26l66
      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
6ij7M5PBMx2kwwyz62Oj4SL5S0mRCwl3mlRmw     24-525-332-7244
7ih7yRz214zO67AiNPx64nO5l5k yj6i3jLA5PCL15Q4QlA3ll60iM1P iBxCixg6
1hCh2RCnjOzk5R OnO 10hhC3m4631m5
       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
-----------------------------------------------


Stop timestamp  12/15/97 17:45:04
```

Qualification Query 11

```
Start timestamp  12/15/97 17:45:04

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

Tag: Q11      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

Truncated at 10 rows

Number of rows retrieved is:  22
```

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

Stop timestamp  12/15/97 17:45:05
```

Qualification Query 12

```
Start timestamp  12/15/97 17:45:12

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

Tag: Q12      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
---------------------------------------------

Stop timestamp  12/15/97 17:45:13
```

Qualification Query 13

```
Start timestamp  12/15/97 17:45:21

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

Tag: Q13      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
---------------------------------------------

Stop timestamp  12/15/97 17:45:21
```

Qualification Query 14

```
Start timestamp  12/15/97 17:45:08

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

Tag: Q14      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
---------------------------------------------

Stop timestamp  12/15/97 17:45:09
```

Qualification Query 15

```
Start timestamp  12/15/97 17:45:01

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

Tag: Q15c     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
---------------------------------------------

Stop timestamp  12/15/97 17:45:02
```

Qualification Query 16

```
Start timestamp  12/15/97 17:45:07

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

Tag: Q16      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
---------------------------------------------

Stop timestamp  12/15/97 17:45:08
```

Qualification Query 17

```
Start timestamp  12/15/97 17:45:13

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

Tag: Q17      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
-----------------------------------------------


Stop timestamp  12/15/97 17:45:14
```

C.3 First 10 rows of Test Database Tables

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

R_REGIONKEY R_NAME                    R_COMMENT

----------- -------------------------
-----------------------------------------
-------------------------------------------------------------------------------
--------
------------------------------
          0 AFRICA
xSx31zz31Cl1z4OAnmm05AjiOxC3AMMNOgC0kACgwn
gg3glP7LLLywlQy7R

          1 AMERICA
kgyh3LSnC72k6zlAz0LP3k2L4QB1QL1O673OjO1SPj
0ngQ7CO100SBgmgRQ4lgPCMk21A425iklyAR4yBRAwR4Cm5miNw 4jl13mMnxw17B


          2 ASIA                      NSg6xlMlA1lzm6mOR0Ajx
nhRA77NgRxBwL1M6Py R
jySB3RLwkyPkwMM2R1BQ xAzkOgkjml10gAghinP5inmNmR76MlijMS3S2zxONR15


          3 EUROPE
zlSL7Qwg12hMBL5lhlz0M45QkjShwSyiO04MLOh7wn
1ARLQPyPAyAiil576l1Li7AlnR1S 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
Lliwk6hMh300izngN32CPwC
ikyLk6khMzSRA


          5 ETHIOPIA                               0 NS7n LSOP
Oz5n1AlB2S02nN0lMh4S
BxP iRhBO 047R26 2BlM


         14 KENYA                                  0
yA0jNSxmS25NRiCy1B6yhSL4NLMROQ
3kLyAPx43wil27kN1QmPLmPw7ywh2kyj2iSBwSjx26zClx

         15 MOROCCO                                0
7mC2hnARnj21nwllAOhR2nwRmgz22A
kP6xj3iQiPNB0PjxBMzS3g
4y0L4Alx1gNCAP2kjQMBR2xwiiSNLgCMQx74RmhMy4OONhPCNQQ 3y

         16 MOZAMBIQUE                             0
LB6SkSOmkznRLS4z5P2BOmC23RnA6h
7mn0gPAAN7nkxy6jO0k3w3R32R6A1ASk0LMyngO36jCL1gl5wQw4AMPC


          1 ARGENTINA                              1
zQn3Okwz1wLn7PLS3OhCgn56kP5PyR
ikgi1B7lL


          2 BRAZIL                                 1
gLmS0nACAmnBCj2klki7RCPNgPxnCO
jNg4k OiAg57COSOm1NwCnOyLx40R SC
y20gPPAkNk5hxRhR5OmgS1iPQQzNAxPL30n67OgyC l617S
h4LS
          3 CANADA                                 1 4yMO AhnQ5Lh
wzQAM662Aw1ByCl7C
xmzRwNR5nAlO4 x


         17 PERU                                   1 35h63
yS7gjRh22iyBlhy5wjLMQnln
74k0j5O ygx1miQgA66hyhP3740PQOgRkwhy1i3LSwzjB 3yjh0QnyMMN A7Qn
6NOC2jg65QRLAgS

         24 UNITED STATES                          1
QQ41AxzBSQACRA1wCLyONCi5yzCylj
QR144lnNLm2 Q7PRyk BRzP



  10 record(s) selected.


SELECT * FROM TPCD.PART P_PARTKEY ORDER BY P_PARTKEY FETCH FIRST 10 ROWS
ONLY

P_PARTKEY   P_NAME
P_MFGR
            P_BRAND    P_TYPE                      P_SIZE
P_CONTAINER P_RET
AILPRICE            P_COMMENT
----------- -------------------------------------------------------------
------------
------------- ---------- ------------------------- -----------
----------- ------------------------ -------------------------
          1 goldenrod lace spring chartreuse ivory
Manufacturer
#1            Brand#13   PROMO BURNISHED COPPER               7 JUMBO PKG
    +9.
01000000000000E+002 zMg1PACmQ 7RCCC7
          2 snow ghost azure burnished lemon
Manufacturer
#1            Brand#13   LARGE BRUSHED BRASS                  1 LG CASE
    +9.
02000000000000E+002 Bxg4RlO6051n7NjN zn
          3 cornflower navajo salmon lemon orchid
Manufacturer
#4            Brand#42   STANDARD POLISHED BRASS             21 WRAP CASE
    +9.
03000000000000E+002 4241RR3By
          4 olive dim lemon light khaki
Manufacturer
#3            Brand#34   SMALL PLATED BRASS                  14 MED DRUM
    +9.
04000000000000E+002 z1n7znz6
          5 lavender cornsilk linen seashell lemon
Manufacturer
#3            Brand#32   STANDARD POLISHED TIN               15 SM PKG
    +9.
05000000000000E+002 gj4Lg5BhBk12iS
          6 cornsilk beige chartreuse medium blue
Manufacturer
#2            Brand#24   PROMO PLATED STEEL                   4 MED BAG
    +9.
06000000000000E+002 yNjzS Njyh4mgLx Om
          7 honeydew purple cream mint coral
Manufacturer
#1            Brand#11   SMALL PLATED COPPER                 45 SM BAG
    +9.
07000000000000E+002 PSNg0L
          8 puff blush tomato papaya navy
Manufacturer
#4            Brand#44   PROMO BURNISHED TIN                 41 LG DRUM
    +9.
08000000000000E+002 k042AL4y21N1yNPC77
          9 burnished violet pink rose drab
Manufacturer
#4            Brand#43   SMALL BURNISHED STEEL               12 WRAP CASE
    +9.
09000000000000E+002 37PLkwhgiAP0xCkxO
         10 slate dark white lavender purple
Manufacturer
#5            Brand#54   LARGE BURNISHED STEEL               44 LG CAN
    +9.
10010000000000E+002 wPP74M1Lwj1

  10 record(s) selected.


SELECT * FROM TPCD.SUPPLIER ORDER BY S_SUPPKEY FETCH FIRST 10 ROWS ONLY

S_SUPPKEY   S_NAME                    S_ADDRESS
          S
_NATIONKEY S_PHONE         S_ACCTBAL                S_COMMENT



------------ --------------------------
---------------------------------------- -
---------- --------------- ------------------------
-----------------------------
-------------------------------------------------------------------------------
-
          1 Supplier#000000001        N kw4gn1OM Ahw3Sg70BBgQw57lgjzj55R

         17 27-918-335-1736   +5.75594000000000E+003 lLniMi51QPmO1
C2hy27wkN21mmg
53 BhQBB1O2x4OmiR4kO5kN1BS 4PwMhk Pk2nRnA2 k

          2 Supplier#000000002        ji3yh016B5

         15 15-679-861-2259   +4.03268000000000E+003
B32z0yzh2lPyOwQkAjA704yM2R7l
lRlk2 xCliy41QNmQn0RNl0OQ4jgMy3kSRBLzyw25CB5 lk0A 54

          3 Supplier#000000003        mxBQBnxO3CSwl7

         11 11-383-516-1199   +4.19239999999999E+003
BS0OzjiO1yM6Rgl4mxLNhjSMPB37
Sw7ym3R7ll2n4SSCilz6nlL5SBOig

          4 Supplier#000000004        7zR323R73NMB77wi1

         15 25-843-787-7479   +4.64108000000000E+003 w
lQn6QyOSSxhw10C6gz2BngiLRA
MmgnRxiLiO3
```

```
          5 Supplier#000000005        AmMQ7Mg
10ByLCP52M13xN31jh5hzOgnm00B
        11 21-151-690-3663   -2.83840000000000E+002
PAziBQQixjwS7P4lQhn10i74050M
 AzkxACnOAyjnSm3CQ26SOx5kynSR0nlLSzi3y3nzPPNikNl3P3 kLwwOP7AM30CO0ymAh

          6 Supplier#000000006        QQL6hxmnMkkgMwgm7CB5B 30Llz

        14 24-696-997-4969   +1.36579000000000E+003 giSki24 gRNAmB
1yOzPR6Q2kiNC
Q0h3LLyxmROA5O7OOi5zlzy

          7 Supplier#000000007        z45m2jBRzl5ilLNz4

        23 33-990-965-2201   +6.82035000000000E+003
1PhngjmiSQl0RzRACP0l4S70xSL
QPSBM16072SkMLCgm4OOMjARLNQk3g1P3BB32AgBMl462B0CP7Rh24

          8 Supplier#000000008        xz5m4C  A4AAj0kANQ

        17 27-498-742-3860   +7.62785000000000E+003
lz57Mw6RNwCSCzmAShwN7S45w20C5zS6zi 5Al1ORMwnQmjS5SgBnRhQ1lCkyBlhN 6MP7
kAzNw3gSjyyLMiNzhCmPn0 g5x23Q
          9 Supplier#000000009        m7k7CnC3wiP

        10 20-403-398-8662   +5.30237000000000E+003 xPLzNgk5nzA
jm3PLmySlm PS  z
ANRjSgh2njAg

         10 Supplier#000000010        wN1S4mQ0g7Px5Lj34xw6kS4Li4NzB4mO

        24 34-852-489-8585   +3.89191000000000E+003
5xwg6AOz0NzhONL6kC43zR3AhzO6
njCiwPg7k6MxwP1mN2 Rg 5Q426


  10 record(s) selected.


SELECT * FROM TPCD.PARTSUPP ORDER BY PS_PARTKEY FETCH FIRST 10 ROWS ONLY

PS_PARTKEY  PS_SUPPKEY  PS_AVAILQTY PS_SUPPLYCOST            PS_COMMENT




----------- ----------- ----------- -------------------------
--------------------
------------------------------------------------------------------------------
--------
------------------------------------------------------------------------------
--------
---------------------
          1           2        3325   +7.71640000000000E+002
00PL56QkQRSkg2z7MAN
Nj4i1h2zLQQLiQnAlML1S6 k4hg3hP5hk3ywMLwy 7gjR3 4Q7S1Qmzx2jOS37Mk61n
yCg4Q7k522P0
055wg23B0Mw3BOwSy6z5Q6xljABx3LAj6R6CmM
l4jjMzQ02LkiiyCCwBk7w465kLBz7QlCk26ARLOxk
7z2hC0jw7
          1      250002        8076   +9.93490000000000E+002
nSO7Mln4N7LlxgAyM2M
zNnO7k0NlhjyShgCy30A 27QML0SQ77CPPgkCQAQCwz5M3MmSSAQ
LxMLMC0Bj4CNN276SmQRSljPxz5
z3 L2mLMQSBghjLnCOR4Nl 156OMP C76QL
xiyw0kSQy1w6ygAxOA4hx7Ngihi5NAPNl2LQ4SRnNhn7
mygOB0z
          1      500002        3956   +3.37090000000000E+002 6215k
jLCizNlOBl62n
P4lLQy431kOzyzn2M6L3h73lCljhlx3x5ghjlOyL76A0hzPk2CS2jkxN gAN3gnk652
Cj4k4



          1      750002        4069   +3.57840000000000E+002
234OCA5ghw0P0gS3n2j
CS35yAm 3L5C7iB k7 w1 R52LLOACQ6i606OB
2MP1x0wC23ik2Omk4NnxzmS6z5z5il661l2l2g0P3
OLk66jzQxjSSAnwSnQ3xz QBQR2j10hNmmyQ14hl45l4x5C B5Qz Lk26yhQNmS54A2O7wlC
P0C57Cy
xSL3
          2           3        8895   +3.78490000000000E+002
MMNOM3BnMM6NBzjB 2m
g i jALB
nQhBM5ROi5N7A5w4B4S2k1506OzMgh6SRB7nP1hQCjgjR17SBA77g6niCwi0L6Pghh1S0O4
mlSL1OShRkyxQS7NNQj


          2      250003        4969   +9.15270000000000E+002
6S66zNlykhii26wwAxz
1PRMxggAy446yylPBS5wP Rl6ggNkyikkyhxymMShNgQmBim1N60
00NSjwPw02lhPPSmn3yRSCn105l
lnPBk2MlRlxxQAmR mO2kxiRh5Pk4x2OnS40nnQRm
16L6NC2RSAkL36g6w64L5w4w74Lnz5wROSOQNi
ngx4imSnPwz5N hSxNg
          2      500003        8539   +4.38370000000000E+002 BPOgj3k
MgQR2 x6kn3
BR6lkmhMzjQk6S343LmN6lz2Qh512MSi4nQ5Bghxlh401yyn463mxOh7Bxgx1PR60jwSMQ66
1OCA6mPO
33R05R2S6N330in0Qx0AC4QhBnOkz4NlQkMR2gh3 k1xk SjN40C77hm2Qlh01jAkSP


          2      750003        3025   +3.06390000000000E+002
y5BNy3Aw02nxyMxgzP5
BS14gg7MCnLlSkih56gOMOy4QyNj5P3iM jOmkRQhRR3hylCn4jN LlgSxyPigjRLgBygM
RR4CL3Pjx
6CRMNlliA7w2ALwkzn06khOOzyQNB63wP4BLSk wPNk4My


          3           4        4651   +9.20920000000000E+002 P7
437MmnM0Pik lAwB
j0gSnml2lzAMA6417zgS154nLCC0Q6BC1lgxyB6BkOj6QCC6n4mw2w7jgCNP zz5AMw37 z

          3      250004        4093   +4.98130000000000E+002
PyRmlwO76kO3igxhS64
h5x6PBLM2PxkO0j3NMRgzP6S1ghhw4Nnn04my03lzCyQQ4M3gS2kO2iOmxPzlCM330zN1yPn
zml1ixgB



  10 record(s) selected.


SELECT * FROM TPCD.CUSTOMER ORDER BY C_CUSTKEY FETCH FIRST 10 ROWS ONLY

C_CUSTKEY   C_NAME                    C_ADDRESS
      C
_NATIONKEY C_PHONE         C_ACCTBAL                 C_MKTSEGMENT
C_COMMENT



----------- -------------------------
---------------------------------------- -
---------- --------------- ------------------------- ------------
---------------
------------------------------------------------------------------------------
--------
----------------------
          1 Customer#000000001        ANhzAAh6R3 glS4Sx

         15 25-989-741-2988   +7.11560000000000E+002 BUILDING
j5S37kk6zkOzkM5
NOz6jwwimkN66CmOhM5ySy w6PAj2xjOAmhkw6ChSR 21BMRkL0kLM5zxg654CRlB3
1Lxm3S

          2 Customer#000000002        MN0L3OzNgylx2

         13 23-768-687-3665   +1.21650000000000E+002 AUTOMOBILE
M4QB23ixkg0yk6m
3gwim6zi32PS7lj2


          3 Customer#000000003        PSL74SNCwwN2ON66lxgnw7mR4hLP2k

          1 11-719-748-3364   +7.49812000000000E+003 AUTOMOBILE
mSC13MBj4n0P6Mg
h0ml02zOBlyjw3NzB1


          4 Customer#000000004        mknn1Sh0NPMz1k5Lw2OB mO

         4 14-128-190-5944   +2.86683000000000E+003 MACHINERY
MN6ChhSMwPwzOky
ww7C5ROlhMS0C4iR2nC6kQmywx3yim62QNySOmQRQnwizihMOg


          5 Customer#000000005        yOww5znhPNi5OlQNPChkLx2BLPxNSB

         3 13-750-942-6364   +7.94470000000000E+002 HOUSEHOLD    24BOSzg
03m710w
ll   iNxnwQ0OmzzgO7A3ykBj2 g755hhCyMO7QnARx5Pg3kyAQA35
i0CS1MSLg0xN2iyg0liwnMwnO
x52nj5iQkNQPP
          6 Customer#000000006        nS70ykL4n k51ik3R5wlNzjnjBL2N51ki

         20 30-114-968-4951   +7.63857000000000E+003 AUTOMOBILE
hPMLmxPw05R1mz1
26jjRAj1kOP7xLC6
yS3ALCRBR5B3im650BLm4O3SwBP7xlwOklmPRS31RNN0gMkkPm4COigCRMlniz2
7jwg63yz
          7 Customer#000000007        ChljB04OgAizN6kQhRi7LjjNiCM0A AS

         18 28-190-982-9759   +9.56195000000000E+003 AUTOMOBILE
QM63L2miSw3hy34
iQ11235 011mkgk0SkCRC73L1CgiLROzNwjO4PQSBx2n2iQg5h


          8 Customer#000000008        kCRz0CknMw7mh4P50QjBnxSLRxQCMOAh
yNn
         17 27-147-574-9335   +6.81974000000000E+003 BUILDING
xlRh1P5M73Lix x
yM Lmng0RO4MBQyLll7wzwgyOLCxi2yCLgL1zO4yOiAPj


          9 Customer#000000009        L4z65g2RNgN6PxM5kRjnPB7k2kwL62

         8 18-338-906-3675   +8.32407000000000E+003 FURNITURE
7zRiSzmj4Ak7L6N
7RljhM5437B6CPmP54RC1x1x7C6hziN6l


         10 Customer#000000010        L3jg3xAwi6A0B103B0Aymm

         5 15-741-346-9870   +2.75354000000000E+003 HOUSEHOLD    7Lm
LiCwwxQMykg
NOR6kzCyP1B21QyA57hBlSOPnx6m53iSOP6w44M3CP MnP7Alky4OwkOwSh20341



  10 record(s) selected.


SELECT * FROM TPCD.ORDERS ORDER BY O_ORDERKEY FETCH FIRST 10 ROWS ONLY

O_ORDERKEY  O_CUSTKEY   O_ORDERSTATUS O_TOTALPRICE
O_ORDERDATE O_ORD
ERPRIORITY O_CLERK         O_SHIPPRIORITY O_COMMENT


----------- ----------- ------------- -------------------------
----------- -----
---------- ---------------- ---------------
----------------------------------------
----------------------------------------
          1     3689999 O                +2.24560830000000E+005
01/02/1996  5-LOW
```

```
          Clerk#000095055               0 A0xCm5ARNL
mxjChn2kC64xA4L6zBg2O5jhg M
42izyPO  QlymN1ky5kmSiSgBAQA
         2     7800163 O                +3.99888500000000E+004
12/01/1996  1-URG
ENT       Clerk#000087916               0 5PRxL1nM7xhQNzP2hnjhy1zz
ykhg4P2A MMg5Px3OCN 0B0iyCRgiC2
         3    12331391 F                +3.06426430000000E+005
10/14/1993  5-LOW
          Clerk#000095426               0 nm0kygQBnw7RS3AAA4k


         4    13677602 O                +7.88405000000000E+003
10/11/1995  5-LOW
          Clerk#000012340               0 CP42CySQlz64n3mCyjm17 4B0CL
L5772m4k2A
i4h1nPySwSmNyCl4jOAOx5y4Rjx36nhO1x2x4Qw
         5     5562202 F                +1.84997170000000E+005
07/30/1994  5-LOW
          Clerk#000092480               0
3PNC7zMP534MSizgy34Bxj62l0C7n6PBk7

         6     3913430 F                +4.19510999999999E+003
02/21/1992  4-NOT
 SPECIFIED Clerk#000005798              0 1CN00NA0z75SwwCxMNB0MLNL


         7    13005694 O                +2.05492900000000E+005
01/10/1996  2-HIG
H         Clerk#000046961               0 gmiC6hj5L4
0ixCAQkmB6giC16l4L16g

         32     6695788 O               +2.14045340000000E+005
07/16/1995  2-HIG
H         Clerk#000061561               0 7ihNSz00NCxA31PPx6RM4ih
BPPlz417SLk3SR
A1zx0nlikRgjkx
         33     6100004 F               +1.07296910000000E+005
10/27/1993  3-MED
IUM       Clerk#000040860               0 jkACLh 0igMiy72n
Sky0h0B6NB70j7Q

         34     8611441 O               +1.28003890000000E+005
07/21/1998  3-MED
IUM       Clerk#000022278               0 05k 2x242klm jyA
wB0CBzzQnz5P11nAml5AL
5jC lg5

  10 record(s) selected.


SELECT * FROM TPCD.LINEITEM WHERE timestamp(generate_unique()) < current
timesta
mp - current timezone + 30 seconds ORDER BY L_ORDERKEY optimize for 1
rows FETCH
 FIRST 11 ROWS ONLY

L_ORDERKEY  L_PARTKEY   L_SUPPKEY   L_LINENUMBER L_QUANTITY
 L_EXTE
NDEDPRICE          L_DISCOUNT               L_TAX
L_RETURNFLA
G L_LINESTATUS L_SHIPDATE L_COMMITDATE L_RECEIPTDATE L_SHIPINSTRUCT
      L
_SHIPMODE L_COMMENT
----------- ----------- ----------- -------------
------------------------- ------
----------------- ------------------------- -------------------------
-----------
- ------------- ----------- ------------- --------------
------------------------- -
--------- ---------------------------------------------
         1     1563445       63448            6
+3.20000000000000E+001   +4.8
2678399999999E+004   +7.00000000000000E-002   +2.00000000000000E-002 N

  O           01/30/1996 02/07/1996   02/03/1996    DELIVER IN PERSON
      M
AIL       CB0SnyOL PQ32B70wB75k 6Aw10m0wh
         1    15518935      768951            1
+1.70000000000000E+001   +3.3
2037200000000E+004   +4.00000000000000E-002   +2.00000000000000E-002 N

  O           03/13/1996 02/12/1996   03/22/1996    DELIVER IN PERSON
      T
RUCK      iPBw4mMm7w7kQ zNPL i261OPP
         1     6730908      730909            2
+3.60000000000000E+001   +6.9
7885200000000E+004   +9.00000000000000E-002   +6.00000000000000E-002 N

  O           04/12/1996 02/28/1996   04/20/1996    TAKE BACK RETURN
      M
AIL       5wM04SNyl0AnghCP2nx lAi
         1     6369978      369979            3
+8.00000000000000E+000   +1.6
3812800000000E+004   +1.00000000000000E-001   +2.00000000000000E-002 N

  O           01/29/1996 03/05/1996   01/31/1996    TAKE BACK RETURN
      R
EG AIR    SQC2C 5PNCy4mM
         1     2402664      152671            5
+2.40000000000000E+001   +3.7
5969600000000E+004   +1.00000000000000E-001   +4.00000000000000E-002 N

  O           03/30/1996 03/14/1996   04/01/1996    NONE
      F
OB        C2gOQj OB6RLk1BS15 igN
         1      213150      463151            4
+2.80000000000000E+001   +2.9
7679200000000E+004   +9.00000000000000E-002   +6.00000000000000E-002 N

  O           04/21/1996 03/30/1996   05/16/1996    NONE
      A
IR        Om0L65CSAwSj5k6k
         2    16818625      818626            1
+2.40000000000000E+001   +3.70267200000000E+004   +0.00000000000000E+000
  +8.00000000000000E-002 N

  O           03/05/1997 02/09/1997   03/11/1997    COLLECT COD
      A
IR        O52M70MRgRNnmm476mNm
         3    10616973      116994            1
+3.80000000000000E+001   +7.1
7987200000000E+004   +0.00000000000000E+000   +5.00000000000000E-002 A

  F           12/11/1993 11/27/1993   12/16/1993    TAKE BACK RETURN
      R
AIL       3AR yMS77lQ12kR
         3     443837      193838            5
+6.00000000000000E+000   +1.0
6848600000000E+004   +4.00000000000000E-002   +1.00000000000000E-002 R

  F           01/28/1994 12/15/1993   02/14/1994    TAKE BACK RETURN
      R
EG AIR    6nmBmjQkgiCyzCQBkxPPOx5j4hB 0lRywgniP7
         3    10016338       16339            3
+4.40000000000000E+001   +5.5
1685199999999E+004   +7.00000000000000E-002   +3.00000000000000E-002 R

  F           01/19/1994 12/23/1993   02/15/1994    TAKE BACK RETURN
      S
HIP       LhiA7wygz0k4g4zRhMLBAM
         3     4580261       80270            4
+3.70000000000000E+001   +4.9
6184800000000E+004   +9.00000000000000E-002   +5.00000000000000E-002 A

  F           12/22/1993 12/28/1993   01/04/1994    DELIVER IN PERSON
      A
IR        jykM7l5S7xl6nin26jhn75C60kj2iQAgg
```

C.4 Query Substitution Parameters

```
Power stream    Seed = 634771394
-- TPC-D Parameter Substitution (Version 1.2.0 )
-- using 634771394 as a seed to the RNG
Q1  DELTA        117
Q2  SIZE         47
    TYPE         BRASS
    REGION       EUROPE
Q3  SEGMENT      MACHINERY
    DATE         1995-03-20
Q4  DATE         1997-08-01
Q5  REGION       MIDDLE EAST
    DATE         1996-01-01
Q6  DATE         1997-01-01
    DISCOUNT     0.06
    QUANTITY     25
Q7  NATION1      UNITED KINGDOM
    NATION2      MOROCCO
Q8  NATION       UNITED KINGDOM
    REGION       EUROPE
    TYPE         LARGE POLISHED BRASS
Q9  COLOR        thistle
Q10 DATE         1994-12-01
Q11 NATION       UNITED KINGDOM
    FRACTION     0.0000010000
Q12 SHIPMODE1    SHIP
    SHIPMODE2    MAIL
    DATE         1996-01-01
Q13 CLERK        Clerk#000000935
Q14 DATE         1997-10-01
Q15 DATE         1997-08-01
Q16 BRAND        Brand#54
    TYPE         LARGE BRUSHED
    SIZE1        24
    SIZE2        26
    SIZE3        17
    SIZE4        29
    SIZE5        6
    SIZE6        40
    SIZE7        31
    SIZE8        7
Q17 BRAND        Brand#54
    CONTAINER    JUMBO BAG
```

Appendix D: Implementation Specific Layer & Driver Code

D.1 tpcdbatch.sqc

```
/************************************************************************
******
*
*   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 "2.3"

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

#define TPCDBATCH_MAX_COLS    100                               /*
@d30369 tjg */

#define TPCDBATCH_CHAR char

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

/* #define TPCD_PREPARETIME 1 */      /* for separate prep/exec on uf
jen 1106 */

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

#define PARALLEL_UPDATES 1

#ifdef PARALLEL_UPDATES
#define UF1OUTSTREAMPATTERN "%s%cuf1.%02d.%d.out"
#ifdef TPCD_V2
#define UF2OUTSTREAMPATTERN "%s%cuf2.%02d.%d.out"
#else
#define UF2OUTSTREAMPATTERN "%s%cuf2.%02d.%d.%d.out"        /*DELjen add
delchunk*/
#endif
#define BUFSIZE 1024
#endif


#define T_STAMP_FORM_1 1
#define T_STAMP_FORM_2 2


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

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



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

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


/** Timer-necessary defines for portability **/
#if (defined (SQLOS2) || defined (SQLWINT))
typedef struct timeb Timer_struct;
#elif (defined (SQLUNIX) || defined (SQLAIX))
typedef struct timestruc_t 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 */

/************************************************************************
****/
/* 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    */
   time_t               stream_start_time;     /* start time for stream
     */
   time_t               stream_end_time;       /* end time for stream
     */
   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;                        /* @d30369
tjg */
   unsigned int         stmt_num;                           /* @d24993
tjg */
   double               elapse_time;                        /* @d24993
tjg */
   double               adjusted_time;
   char                 start_stamp[50];       /* start time stamp for
block  */
   char                 end_stamp[50];         /* end time stamp for
block    */
```

```
   char                tag[50];              /* block tag
    */
   struct stmt_info    *next;                                  /* @d24993
tjg */

};


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

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


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


/*********************************************************************/
/* Structure containing control flags passed between functions       */
/*********************************************************************/
struct ctrl_flags
{                                                              /* @d25594
tjg */
   int eo_infile;
   int time_stamp;
   int eo_block;                                               /* @d30369
tjg */
   int select_status;
};


/*********************************************************************
/
/* Function Prototypes
*/
/*********************************************************************
/
int Get_SQL_stmt(struct global_struct *g_struct);

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

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

long error_check(void);                                        /* @d28763
tjg */

void 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 );

#ifdef PARALLEL_UPDATES
void runUF1_fn( int updatePair, int i, int copyOnOrOff );
void runUF2_fn( int updatePair, int i, int deleteChunk, int copyOnOrOff
);
#endif

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

char *get_time_stamp(int form, time_t time_pointer);      /* @d25594 tjg
*/
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_str[4000] = "\0";      /* Assume max SQL statment
                                       of 4000 char  */
char    dbname[9] = "\0";
char    userid[9] = "\0";
char    passwd[9] = "\0";

EXEC SQL END DECLARE SECTION;
```

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


/** 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 */
char               tempdir[256];           /* to hold TPCD_TMP_DIR
                                              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 */


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

   struct global_struct g_struct =
                { NULL, NULL, NULL, NULL, 0, 0, "\0", 0.1, "\0", 1,
FALSE, 0,
                      NULL, "\0", "\0", NULL };


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

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

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

/*********************************************************************
*******
 *
     *
 *   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_1,(time_t)NULL) );

      /* 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 %18.18s \n",
               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));     /*
@d30369 tjg */


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


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


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

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

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

   return(0);

}  /* end of main  */



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

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

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

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

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


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

         eostmt=0;
      }

      /* Need this hack here to check if there's any more empty lines
left
         in the input file.  Continue only if there are aren't any */
      else if (strcmp(input_ln, "\0")) /* HACK */ {   /* A regular SQL
statement */
         if (stmt_num_flag) {  /* print this out only if it's the first
line
                                  of the SQL statement.  We only want
this
                                  line to appear once per statement */
            if (verbose)
               fprintf(stderr, "\nTag: %-5.5s    Sequence number: %d\n",
\

g_struct->s_info_ptr->tag,g_struct->s_info_ptr->stmt_num);
            fprintf(outstream, "\nTag: %-5.5s    Sequence number: %d\n",
\

g_struct->s_info_ptr->tag,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;                                         /* @d23684
tjg */
         test_semi = strstr (input_ln,";");
         if (test_semi == NULL) {   /* if there's no semi-colon keep on
going */
            strcat (stmt_str,input_ln);
            strcat (stmt_str," ");
            eostmt = 0;
         }

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

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

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

   }  while (!eostmt);

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

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

   /** Determine whether statement is a SELECT or other SQL **/
   if (good_sql) {
      strcpy(temp_str,stmt_str);
      uppercase(temp_str);  /* Make sure that select is made to SELECT
*/
      select_status=strtok(temp_str," ");
      if ( (stmt_str[0] == '(') || (!strcmp(select_status,"SELECT")) ||
           (!strcmp(select_status,"VALUES")) ||
           (!strcmp(select_status,"WITH")) )
         return TPCDBATCH_SELECT;
      else
         return TPCDBATCH_NONSELECT;
   }

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

   else                                                      /* @d23684
tjg */
      return TPCDBATCH_NONSQL;

}  /*  Get_SQL_stmt */



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

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

   for (loopvar=0; loopvar<sqlda->sqld; loopvar++)
   {
      switch (sqlda->sqlvar[loopvar].sqltype)
      {
       case SQL_TYP_INTEGER:                             /* INTEGER */
       case SQL_TYP_NINTEGER:
         if ((sqlda->sqlvar[loopvar].sqldata=
                  (TPCDBATCH_CHAR *)malloc(sizeof(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;                    /* @d22817
tjg */

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

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

          case SQL_TYP_SMALL:
          case SQL_TYP_NSMALL:

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

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

               temp_ptr = temp_string;
```

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

               else
                  strcpy(decimal_string, " ");

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

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

            break;

         case SQL_TYP_CSTR:
         case SQL_TYP_NCSTR:
         case SQL_TYP_DATE:
         case SQL_TYP_NDATE:
         case SQL_TYP_TIME:
         case SQL_TYP_NTIME:
         case SQL_TYP_STAMP:
         case SQL_TYP_NSTAMP:

sqlda->sqlvar[col].sqldata[sqlda->sqlvar[col].sqllen+1]='\0';
            strcpy(temp_string,(char *)sqlda->sqlvar[col].sqldata);
            fprintf(outstream, "%-*s  ",(col_lengths[col]),temp_string);
            break;

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

   fprintf(outstream, "\n");

   return;
}


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

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

#if (defined (SQLOS2) || defined (SQLWINT))
   ftime (start_time);
#elif defined (SQLAIX)
   rc = gettimer(TIMEOFDAY, start_time);
#else
#error Unknown operating system
#endif

   if (rc != 0) {
      fprintf(stderr,"Timer call failed, aborting test\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_nsec;
#endif


#if (defined (SQLAIX) || defined (SQLUNIX))
   status = gettimer(TIMEOFDAY, &end_time);
#elif (defined (SQLOS2) || defined (SQLWINT))
   ftime(&end_time);
#else                           /** If another operating system **/
#error Unknown operating system
#endif

   if (status != 0)
      fprintf(stderr,"Bad return from gettimer, don't trust timer
results...\n");

   else
   {
#if defined (SQLAIX)
      result_sec = end_time.tv_sec - start_time->tv_sec;
      result = (double) result_sec;

      if ((start_time->tv_nsec > 0) && \
          (start_time->tv_nsec < 1000000000) && \
          (end_time.tv_nsec > 0) && \
          (end_time.tv_nsec < 1000000000))
      {
         result_nsec = end_time.tv_nsec - start_time->tv_nsec;
         result = (double) result_sec + ((double)
result_nsec/1000000000);
      }
#elif (defined (SQLOS2) || defined(SQLWINT))
      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);
}



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

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

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

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

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

      EXEC SQL CONNECT ;                                      /* @d28763
tjg */

      if (sqlca.sqlcode == SQLE_RC_NOSUDB ) { /* no connection exists */

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

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

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

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

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

      }
   }
   return (temp_sqlca.sqlcode);

}  /* error_check */



/*************************************************/
/* Displays a help screen                        */
/*************************************************/
void display_usage()
{
   printf("\ntpcdbatch  --  version %s",TPCDBATCH_VERSION);
   printf("\n\nSyntax is:\n");
   printf("tpcdbatch [-d dbname] [-f file_name] [-l file_name] [-r
on/off]");
   printf("\n         [-v on/off] [-b on/off] [-u p/t/t1/t2]");
   printf("\n         [-s scale_factor] [-n stream_num] [-m inlistmax]
[-h]\n");
   printf("\n  where: -d  Database name");
   printf("\n                  Default - dbname set in $DB2DBDFT");
   printf("\n         -f  Input file containing SQL statements");
   printf("\n                  Default - stdin ");
   printf("\n         -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' :                                              /* @d26350
tjg */
          case 'F' :

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

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

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

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

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

          case 'u' :                                              /* @d26350
tjg */
          case 'U' :
            g_struct->c_l_opt->update=-1; /* init to invalid number */
            if (!strcmp(uppercase(argv[++loopvar]),"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' :                                              /* @d26350
tjg */
          case 'B' :
            if (!strcmp(uppercase(argv[++loopvar]),"ON"))
               g_struct->s_info_ptr->query_block=1;
            else
               g_struct->s_info_ptr->query_block=0;
            break;

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

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

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

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

#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 jen -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);
   }

   if (getenv("TPCD_TMP_DIR") != NULL)
      strcpy(tempdir, getenv("TPCD_TMP_DIR"));
   else {
      fprintf(stderr,"\nThe TPCD_TMP_DIR environment variable\n");
      fprintf(stderr,"is not set....exiting\n");
      exit(-1);
   }

#ifdef PARALLEL_UPDATES
   /* hijack the plane and take it to Timbuktu */
   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,
copyOnOrOff);
         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(                                          /*kmw*/
                                                       /* C++ */char
*decptr,
                                                       /* C++ */char
*asciiptr,
                                                       short prec,
                                                       short scal)
{/* */
   int allzero = TRUE;
   /* C++ */char *srcptr;
   unsigned char sign;
   /* C++ */char *targptr, decimal_point = '.';
   int rc = 0;                                        /*kmw*/
   int tmpint, src_nibble;
   int count, j, limit[3];

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


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


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

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

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

   }

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


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

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

   /** 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 (getenv("TPCD_RUN_ON_MULTIPLE_NODES") != NULL)
   {
      if
(!strcmp(uppercase(getenv("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 */
   }
   else
   {
      fprintf(stderr,"\nThe TPCD_RUN_ON_MULTIPLE_NODES environment
variable\n");
      fprintf(stderr,"is not set....exiting\n");
      exit(-1);
   }
   /* check to make sure the TPCD_COPY_DIR environment variable is set
*/
   if (getenv("TPCD_COPY_DIR") == NULL)
   {
      fprintf(stderr,"\nThe TPCD_COPY_DIR environment variable is not
set");
      fprintf(stderr,"....exiting\n");
      exit(-1);
   }

   /* 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);
```

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

/*********************************************************************
***
       * SEED jen
       * we want to print the seed used in the output files.  For the
seed usage
       * we can now reuse the seeds from run to run, therefore all the
power runs
       * will use the 1st seed in the file, and the throughput streams
will use
       * the 2nd to #streams+1 seeds.
       * determine the seed to use...e.g. given 3 streams will have the
following:
       *                                        Entry in seed file
       *      TEST        Stream Number     Run 1    Run 2
       *      power        0                1        1
       *      throughput   1                2        2
       *                   2                3        3
       *                   3                4        4

*********************************************************************
*/
      seedEntry = g_struct->c_l_opt->intStreamNum + 1;
      /* end SEED jen */
      /* open the generated seed file...if not there, try the default */

      sprintf(file_name, "%s%sauditruns%smseedme",
              getenv("TPCD_AUDIT_DIR"),
              getenv("TPCD_PATH_DELIM"),
              getenv("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.
  *
 *********************************************************************
****/
   /* jenn add code to handle vldb */
   if ( g_struct->scale_factor >= 1000 )
   {
      sprintf(stmt_str, "SET CURRENT FUCNTION PATH VLDB, SYSTEM PATH");
      EXEC SQL EXECUTE IMMEDIATE :stmt_str;
      if (sqlca.sqlcode < 0)
      {
         sqlcode = error_check();            /* Don't bother printing
the SQL0100W
                                                error from going through
the loop
                                                above one extra time */
      }
   }
```

```
   /** Get start timestamp for stream **/
   g_struct->stream_start_time = time(NULL);
   strcpy(g_struct->file_time_stamp,
          get_time_stamp(T_STAMP_FORM_2,g_struct->stream_start_time));

   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,
                 getenv("TPCD_PATH_DELIM"),
                 g_struct->file_time_stamp);
      }
      else
      {
         /* query stream */
         sprintf(file_name, "%s%sstrcnt%d.%s",g_struct->run_dir,
                 getenv("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 %18.18s\n",

get_time_stamp(T_STAMP_FORM_1,g_struct->stream_start_time));
      }
      else
      {
         /* query stream */
         fprintf(g_struct->stream_report_file,
                 "Stream number %d starting at %18.18s\n",
                 g_struct->c_l_opt->intStreamNum,

get_time_stamp(T_STAMP_FORM_1,g_struct->stream_start_time));
      }
   }

   /* set up the update_num_file name so that if we do use semaphores,
*/
   /* we will have a filename to generate the semkey */
   if ( (getenv("TPCD_DBNAME") != NULL) &&
        (getenv("USER") != NULL)        &&
        (getenv("TPCD_AUDIT_DIR") != NULL) )
   {
      sprintf(g_struct->update_num_file, "%s%s%s.%s.update.pair.num",
              getenv("TPCD_AUDIT_DIR"),
              getenv("TPCD_PATH_DELIM"),
              uppercase(getenv("TPCD_DBNAME")),
              lowercase(getenv("USER")));
      sprintf(g_struct->sem_file, "%s.%s.semfile",
              getenv("TPCD_DBNAME"),
              getenv("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);
      }
}
   else
   {
      fprintf(stderr,"\nThe environment is not set up correctly, ensure
that\n");
      fprintf(stderr,"$TPCD_DBNAME, $USER and $TPCD_AUDIT_DIR are
assigned..\n");
      exit(-1);
   }
   /* 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:                                  /* @d30369
tjg */
         col_lengths[col] = TPCDBATCH_MAX (col_name_length,6);
         break;
       case SQL_TYP_INTEGER:
       case SQL_TYP_NINTEGER:
         col_lengths[col] = TPCDBATCH_MAX (col_name_length,11);
         break;
       case SQL_TYP_CSTR:
       case SQL_TYP_NCSTR:
       case SQL_TYP_DATE:
       case SQL_TYP_NDATE:
       case SQL_TYP_TIME:
       case SQL_TYP_NTIME:
       case SQL_TYP_STAMP:
       case SQL_TYP_NSTAMP:
       case SQL_TYP_CHAR:
       case SQL_TYP_NCHAR:
       case SQL_TYP_VARCHAR:
       case SQL_TYP_NVARCHAR:
       case SQL_TYP_LONG:
       case SQL_TYP_NLONG:
         col_lengths[col] = TPCDBATCH_MAX (col_name_length,col_width);
         break;

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

       case SQL_TYP_DECIMAL:
       case SQL_TYP_NDECIMAL:

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

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

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

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

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

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

}
```

```
/*****************************************************************/
/* Gets the current system time and prints it out                */
/*****************************************************************/
char *get_time_stamp(int form, time_t time_pointer)
{
   time_t  temp_stamp = 0;
   struct tm *tp;

   if (time_pointer == (time_t)NULL)
      temp_stamp = time(NULL);
   else
      temp_stamp = time_pointer;

   tp = localtime(&temp_stamp);
   if (form == T_STAMP_FORM_1)
      strftime(newtime,50,"%x %X",tp);
   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;

   char db_size[8] = "\0";

   int scale_factor = 0;

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

   /* multiply the scale factor by 1000 to be able to handle scale
factors
      less than 1 */
   scale_factor = (int)(g_struct->scale_factor * 1000);
   switch(scale_factor)
   {
    case 10:
      strcpy(db_size,"10MB");
      break;
    case 12:
      strcpy(db_size,"12MB");
      break;
    case 100:
      strcpy(db_size,"100MB");
      break;
    case 1000:
      strcpy(db_size,"1GB");
      break;
    case 3000:
      strcpy(db_size,"3GB");
      break;
    case 8000:
      strcpy (db_size, "8GB");
      break;
    case 10000:
      strcpy(db_size,"10GB");
      break;
    case 30000:
      strcpy(db_size,"30GB");
      break;
    case 100000:
      strcpy(db_size,"100GB");
      break;
    case 1000000:
      strcpy(db_size,"1000GB");
      break;
    default:
      fprintf(stderr,"\nUndefined scale factor\n");
      break;
   }


   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;
   min = s_info_head_ptr;

   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 %18.18s\n",
            get_time_stamp(T_STAMP_FORM_1,g_struct->stream_start_time));
   /* print the stream stop timestamp to the inter file */
   fprintf (outstream, "Stream stop time stamp %18.18s\n",
            get_time_stamp(T_STAMP_FORM_1,g_struct->stream_end_time));

   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%18.18s%18.18s\n",
                  s_info_ptr->stmt_num,s_info_ptr->tag,
                  s_info_ptr->elapse_time,s_info_ptr->adjusted_time,
                  s_info_ptr->start_stamp,s_info_ptr->end_stamp);

         /* 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%18.18s%18.18s\n",

max->tag,max->elapse_time,max->adjusted_time,max->start_stamp,
               max->end_stamp);
      fprintf (outstream,
               "Min Qry %-3.3s %15.1f %15.1f%18.18s%18.18s\n",

min->tag,min->elapse_time,min->adjusted_time,min->start_stamp,
               min->end_stamp);
   }

   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) */
      Ts =
(difftime(g_struct->stream_end_time,g_struct->stream_start_time)) + 1;
      QppD = (3600 * g_struct->scale_factor) / adjusted_g_mean;
      QthD = (num_stmt * 3600 * g_struct->scale_factor) / Ts;
      QphD = sqrt(QppD*QthD);

      fprintf (outstream,
               "QppD@%-8.8s = %10.3f\n\nTs %11 = %10.0f\nQthD@%-8.8s =
%10.3f\n\nQphD@%-8.8s = %10.3f\n",
               db_size,QppD, Ts,db_size, QthD,db_size, QphD);
   }

}

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

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

   free(sqlda);
   sqlda_allocated = 0; /* fix free() problem on NT
                           wlc 090597 */
}

/* ************ PARALLEL_UPDATES version of runUF1 and runUF2 only */

/*********************************************/
/* processing to run the insert update function */
/*********************************************/
void runUF1 ( int updatePair, int copyOnOrOff )
{
#ifndef TPCD_V2
   char statement[3000];
   char sourcedir[256];
#endif

   int split_updates = 2;       /* no. of ways update records are split
*/
   int concurrent_inserts = 2;  /* jenCI no of concurrent updates to be
*/
                                /* jenCI run at once*/
   int loop_updates = 1;        /* jenCI no of updates to be run in one
*/
                                /* jenCI "concurrent" invocation.
should*/
                                /* jenCI be split_updates /
concurrent_inserts*/
   int i, 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];
   char sourcefile[256];
   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);

#ifndef TPCD_V2
   /* for PE we want to call load_update 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_update %d 1",sourcedir,updatePair);
   if (system(statement))
   {
      fprintf (outstream,
               "load_update failed for UF1, examine UF1.log for cause.
Exiting.\n");
      if (verbose)
         fprintf (stderr,
               "load_update failed for UF1, examine UF1.log for cause.
Exiting.\n");
      exit (-1);
   }
   fprintf (outstream, "load_update finished for UF1.\n");

#endif  /* not TPCD_V2 */

   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_V2
      /* 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",
            getenv("TPCD_DBNAME"), getenv("USER"));
   fprintf(stderr,"UF1 semfile = %s\n",UF1_semfile);
   su_hSem = CreateSemaphore(NULL, 0,
                             concurrent_inserts,
/*jenCI*/
                             (LPCTSTR)(UF1_semfile));
   if (su_hSem == NULL)
   {
      fprintf(stderr,
               "CreateSemaphore (ready semaphore) failed, GetLastError:
%d, quitting\n",
               GetLastError());
      exit(-1);
   }
#endif /* SQLWINT */
   if (verbose) fprintf(stderr,"Semaphore created successfully!\n");

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

   for (i=0; i < concurrent_inserts; i++)
/*jenCI*/
   {
#ifndef SQLWINT
      if ((childpid[i] = fork()) == 0)
      {
         runUF1_fn (updatePair, i, 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",
               getenv("TPCD_AUDIT_DIR"), dbname, updatePair, i ); /* aph
082797 */

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

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

   if( (outstream = fopen(outstreamfilename, APPENDMODE)) == NULL )
   {
      fprintf(stderr,"\nThe output file could not be opened.  ");
      fprintf(stderr,"Make sure that the filename is correct.\n");
      fprintf(stderr,"filename = %s\n",outstreamfilename);
      exit(-1);
   }
   /* 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++)
/*jenCI*/
   {
      sprintf (myoutstreamfile, UF1OUTSTREAMPATTERN, tempdir,
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;  /* jenCI no of concurrent updates to be
*/
                                /* jenCI run at once*/
   int loop_updates = 1;        /* jenCI no of updates to be run in one
*/
                                /* jenCI "concurrent" invocation.
should*/
                                /* jenCI be split_updates /
concurrent_inserts*/
   int startChunk = 0;          /* jenCI number of first chunk to insert
for */
                                /* jenCI this child */
   int stopChunk = 0;           /* jenCI number of last chunk to insert
for */
                                /* jenCI this child */
   int chunk = 0;               /* jenCI counter for within the set of
chunks*/

   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_V2
   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;
#endif

   if (getenv ("TPCD_SPLIT_UPDATES") != NULL)
      split_updates = atoi (getenv ("TPCD_SPLIT_UPDATES"));
   if (getenv ("TPCD_CONCURRENT_INSERTS") != NULL)
/*jenCI*/
      concurrent_inserts = atoi (getenv ("TPCD_CONCURRENT_INSERTS"));
/*jenCI*/
   loop_updates = split_updates / concurrent_inserts;
/*jenCI*/
   /* determine the starting and stopping point of the chunks that this
jenCI*/
   /* invocation will apply. i is starting chunk number with range 0
jenCI*/
   /* through (concurrent_inserts -1)
jenCI*/
   startChunk = i * loop_updates;
/*jenCI*/
   stopChunk = startChunk + (loop_updates - 1);
/*jenCI*/

   sprintf (myoutstreamfile, UF1OUTSTREAMPATTERN, tempdir, 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
%18.18s\n",
            updatePair, i, get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
   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_V2
   fprintf( myoutstream,
            "Staging table load for update pair %d set %d starting at
%18.18s\n",
            updatePair, i, get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
   /* jenCI  we must loop around for the sequential chunks that we are
*/
   /* jenCI  applying to load all data chunks
*/
   for ( chunk = startChunk; chunk <= stopChunk; chunk++ )
/*jenCI*/
      {
/*jenCI*/
      /* jenCI  substitions done through this loop of code from i to
chunk */
      /* jenCI  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",
               tempdir, PATH_DELIM, chunk);
      sprintf (localMsgFile2, "%s%ctpcdloadline.msg.%d",
               tempdir, PATH_DELIM, chunk);
      sprintf (remoteMsgFile1, "%s%ctpcdloadorder.rmt.%d",
               tempdir, PATH_DELIM, chunk);
      sprintf (remoteMsgFile2, "%s%ctpcdloadline.rmt.%d",
               tempdir, PATH_DELIM, chunk);

      /* 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;

      /* 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_%d",updatePair, chunk);
      pActionString->length = strlen(pActionString->data);

      /* file modifcation */
      pFileTypeMod = (struct sqlchar *)malloc(256);
      strcpy(pFileTypeMod->data, "COLDEL|");
      pFileTypeMod->length = strlen(pFileTypeMod->data);

      /* 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_%d\n",
                    updatePair, chunk);
         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 inserted into
TPCDTEMP.ORDERS%d_%d at %18.18s\n",

LoadInfoOut.rowsLoaded,dataFileList.target.location->location_entry,
              updatePair, chunk,
get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));

      /* 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);


      /* set up the datafile structure */
```

```
      strcpy(dataFileList.target.location->location_entry,sourcefile);

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

      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_%d\n",
                updatePair, chunk);
         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 inserted into
TPCDTEMP.LINEITEM%d_%d at %18.18s\n",

LoadInfoOut.rowsLoaded,dataFileList.target.location->location_entry,
              updatePair, chunk,
get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
   } /* jenCI end loop processing for all chunks applied by this
invocation */
   fprintf( myoutstream,
            "Staging table load for update pair %d set %d completed at
%18.18s\n\n",
              updatePair, i, get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
   /* 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 /* TPCD_V2 */

   for ( chunk = startChunk; chunk <= stopChunk; chunk++ )
/*jenCI*/
   {
/*jenCI*/
      /* first lock the staging tables into share mode so we use fewer
locks */
      /* jenCI  substitions done through this loop of code from i to
chunk */
      /* jenCI  no s&d codes for this change...to messy to follow */
      sprintf(stmt_str, "LOCK  TABLE TPCDTEMP.ORDERS%d_%d IN SHARE
MODE",
              updatePair, chunk);
      EXEC SQL EXECUTE IMMEDIATE  :stmt_str;
      if (sqlca.sqlcode < 0)
      {
         fprintf(stderr,"Lock table tpcdtemp.orders%d_%d failed
sqlcode=%d\n",
                 updatePair,chunk,sqlca.sqlcode);
         sqlcode = error_check();         /* Don't bother printing the
SQL0100W
                                             error from going through
the loop
                                             above one extra time */
      }
      /* first lock the staging tables into share mode so we use fewer
locks */
      sprintf(stmt_str, "LOCK  TABLE TPCDTEMP.LINEITEM%d_%d IN SHARE
MODE",
              updatePair, chunk);
      EXEC SQL EXECUTE IMMEDIATE  :stmt_str;
      if (sqlca.sqlcode < 0) {
         fprintf(stderr,"Lock table tpcdtemp.lineitem%d_%d failed
sqlcode=%d\n",
                 updatePair,chunk,sqlca.sqlcode);
         sqlcode = error_check();         /* Don't bother printing the
SQL0100W
                                             error from going through
the loop
                                             above one extra time */
      }

      sprintf(stmt_str,
              "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_%d",
              updatePair, chunk);

      /* wlc  062797 */
      sqlcode = SQL_RC_E911;
      maxwait = 1;

      while (sqlcode == SQL_RC_E911 && maxwait <= MAXWAIT)
      {
         sqlcode = 0;
#ifdef TPCD_PREPARETIME
         /* temp code to test timing START*/
         fprintf( myoutstream,
                  "\nUF1 orders update pair %d chunk %d prep at
%18.18s\n",

updatePair,chunk,get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
         EXEC SQL PREPARE INSO FROM :stmt_str;
         fprintf( myoutstream,
                  "\nUF1 orders pair %d chunk %d prep finished at
%18.18s\n",

updatePair,chunk,get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
         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
%18.18s\n",

updatePair,chunk,get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
         /* temp code to test timing end*/
#else
#if 0
            sprintf(statement,"db2 connect to tpcd; db2 list tablespaces
show detail; db2 connect reset; db2 terminate");   /*jen temp*/
            system(statement);  /*jen temp*/
#endif
         EXEC SQL EXECUTE IMMEDIATE :stmt_str;
#endif
         if (sqlca.sqlcode < 0)
         {
            fprintf(myoutstream,
                    "Insert into orders pair %d chunk %d failed
sqlcode=%d\n",
                    updatePair,chunk,sqlca.sqlcode);
            sprintf(statement,"db2 connect to tpcd; db2 list tablespaces
show detail; db2 connect reset; db2 terminate");   /*jen temp*/
            system(statement);  /*jen temp*/

            sqlcode = error_check();       /* Don't bother printing the
SQL0100W
                                              error from going through
the loop
                                              above one extra time */
         }
         if (sqlcode == SQL_RC_E911)
         {          /* we've hit a deadlock */
            fprintf (myoutstream,
                     "\nDeadlock detected inserting from
tpcdtemp.orders%d_%d...Retrying...\n",
                     updatePair, chunk);
#ifndef SQLWINT
            sleep (UF_DEADLOCK_SLEEP);
#else
            Sleep (UF_DEADLOCK_SLEEP*1000);           /* 10 Seconds Des
*/
#endif
            maxwait++;                                /* jen DEADLOCK */
         }
      }
      if (sqlcode < 0)
      {
         fprintf (myoutstream,"\nAn error occurred inserting into
TPCD.ORDERS\n");
         fprintf (myoutstream,
                  "for update pair number %d chunk %d sqlcode=%ld
...Exiting\n",
                  updatePair, chunk,sqlcode);
         if (sqlcode == -911)
         {
            fprintf (myoutstream,"# of deadlocks exceeds %i\n",
MAXWAIT);
         }
         rc=-1;
         goto UF1_exit;
      }
      /* report the number of row inserted */
      fprintf(myoutstream, "%ld rows inserted into TPCD.ORDERS at
%18.18s\n",
              sqlca.sqlerrd[2],
              get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));

      sprintf(stmt_str,
              "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_%d",
              updatePair, chunk);

      /* wlc  062797 */
      sqlcode = SQL_RC_E911;
      maxwait = 1;

      while (sqlcode == SQL_RC_E911 && maxwait <= MAXWAIT)
```

```
      {
         sqlcode = 0;
#ifdef TPCD_PREPARETIME
         /* temp code to test timing START*/
         fprintf( myoutstream,
              "\nUF1 lineitem update pair %d chunk %d prep starting at
%18.18s\n",

updatePair,chunk,get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
         EXEC SQL PREPARE INSL FROM :stmt_str;
         fprintf( myoutstream,
              "\nUF1 lineitem update pair %d chunk %d prep starting at
%18.18s\n",

updatePair,chunk,get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
         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 INSL;
         fprintf( myoutstream,
              "\nUF1 lineitem update pair %d chunk %d prep starting at
%18.18s\n",

updatePair,chunk,get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
         /* 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 the
SQL0100W
                                            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_%d...Retrying...\n",
                     updatePair, chunk);
#ifndef SQLWINT
            sleep (UF_DEADLOCK_SLEEP);
#else
            Sleep (UF_DEADLOCK_SLEEP*1000);           /* 10 Seconds Des
*/
#endif
            maxwait++;                                /* jen DEADLOCK */
         }
      }

      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);
         if (sqlcode == -911)
         {
            fprintf (myoutstream,"# of deadlocks exceeds %i\n",
MAXWAIT);
         }
         rc=-1;
         goto UF1_exit;
      }
      /* report the number of row inserted */
      fprintf(myoutstream, "%ld rows inserted into TPCD.LINEITEM at
%18.18s\n",
              sqlca.sqlerrd[2],
              (char *)get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
      fprintf( myoutstream,
               "UF1 for update pair %d chunk %d complete at
%18.18s\n\n",
               updatePair, chunk,
get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));

      rc=0;

      EXEC SQL COMMIT WORK;
      error_check();
   }  /* end looping through each sequential chunk of data within this
jenCI */
    /* invocation
jenCI */

   fprintf( myoutstream,
            "UF1 for update pair %d set %d complete at %18.18s\n\n",
            updatePair, i, get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
    goto connect_reset;       /*successful, skip rollback */    /*
971021jen */

UF1_exit:                                                        /*
971021jen */
   EXEC SQL ROLLBACK WORK;                                       /*
971021jen */

connect_reset:                                                   /*
971021jen */
   EXEC SQL CONNECT RESET;
   error_check();                                               /* @d22275
tjg */

   /* close myoutstream file */
   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_V2
      /* 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);
      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 )
{
#ifndef TPCD_V2
   char statement[3000];
   char sourcedir[256];
#endif
   int split_deletes = 1;    /*   no. of ways update records are split
@dxxxxxhar */
   int numChunks = 1;    /* number of database partitions DELjen */
   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);

   /* v2 : just read the variable TPCD_SPLIT_DELETES */
   if (getenv ("TPCD_SPLIT_DELETES") != NULL)
      split_deletes = atoi (getenv ("TPCD_SPLIT_DELETES"));
#ifndef TPCD_V2
   /* DELjen in mln/mpp, the TPCD_SPLIT_DELETES really is the */
   /* number of chunks per data partitions whereas split_deletes in the
code*/
   /* later on is used to mean the number of partitons....so set
numChunks */
   /* to be TPCD_SPLIT_DELETES and split_deletes = number of partitions
*/
   /* mln or mpp: deletes are done by calling tpcdbatch on each logical
node */
   if ((getenv ("TPCD_PHYS_NODE") != NULL) &&
       (getenv ("TPCD_LN_PER_PN") != NULL))
   {
      numChunks = split_deletes;
      split_deletes = atoi (getenv ("TPCD_PHYS_NODE")) *
         atoi (getenv ("TPCD_LN_PER_PN"));
   }
#endif
```

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

#ifdef TPCD_V2
#ifndef SQLWINT
   /*    we will use the first flat file to generate the semaphore key
*/
   if (getenv("TPCD_FLATFILES") != NULL)
      sprintf(sourcefile, "%s%cdelete.%d.0",
              getenv("TPCD_FLATFILES"), PATH_DELIM, updatePair);
   else
      sprintf(sourcefile,".%cdelete.%d.0", 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",
            getenv("TPCD_DBNAME"), getenv("USER"));
   fprintf(stderr,"UF2 semfile = %s\n",UF2_semfile);
   su_hSem = CreateSemaphore(NULL, 0,
                             split_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 < split_deletes; i++)
   {
#ifndef SQLWINT
      if ((childpid[i] = fork()) == 0)
      {
         runUF2_fn (updatePair, i, 0, copyOnOrOff);   /* DELjen use
deleteChunk */
         /* parm = 0 when running smp/uni only used for mln/mpp since
this */
         /* call handles the chunking in smp/uni */
      }
      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; */

#if 0
         sprintf (commandline,
                  "\\tpcdaudit\\auditruns\\tpcdbatch -z -d %s -i %d -j 2
-k %d",
                  dbname, updatePair, i );
         if (CreateProcess("tpcdbatch", commandline, NULL, NULL, FALSE,
                           DETACHED_PROCESS, NULL, NULL, NULL,
&childprocess[i])
             /*
               if (CreateProcess(NULL, commandline, NULL, NULL, FALSE,
               0, NULL, NULL, NULL, &childprocess[i]) */
             == FALSE)

         {
            fprintf (stderr, "CreateProcess failed: GetLastError: %d\n",
                     GetLastError());
            exit (-1);
         }
         else
         {
            if (verbose)
               fprintf (stderr, "Stream #%d started with pid %d\n",
                        childprocess[i].dwProcessId);
         }
#endif
         sprintf (commandline,
            "start /b %s\\auditruns\\tpcdbatch.exe -z -d %s -i %d -j 2
-k %d -x 0",
            getenv("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, split_deletes * -1) != 0)
/*jenSEM*/
   {
/*jenSEM*/
      fprintf(stderr,
              "Failure to update wait on delete semaphone with %d
children\n",
              split_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

#else  /* non TPCD_V2 */
   if (getenv("TPCD_PELOAD_DIR") != NULL)
      strcpy(sourcedir,getenv("TPCD_PELOAD_DIR"));
   else
      strcpy(sourcedir,".");

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

#endif /* TPCD_V2 */

   if( (outstream = fopen(outstreamfilename, APPENDMODE)) == NULL ) {
      fprintf(stderr,"\nRunUF2 The 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 < split_deletes; i++)    /* START AT NODE 0 */
      /* for (i=1; i <= split_deletes; i++)   */     /* start at node 1
*/
      {
#ifdef TPCD_V2
      sprintf (myoutstreamfile, UF2OUTSTREAMPATTERN, tempdir,
PATH_DELIM,
               updatePair, i);
      if ( (myoutstream = fopen (myoutstreamfile, READMODE)) == NULL)
      {
         fprintf (outstream,
                  "\nThe output file '%s' for update pair %d set %d
could not be opened during runUF2\n",
                  myoutstreamfile,updatePair,i);
         exit (-1);
      }
#else
      /* we have chunks of data on each node, so gather all these files
DELjen*/
      for (c=0; c < numChunks; c++ )
      {
         sprintf (myoutstreamfile, UF2OUTSTREAMPATTERN, tempdir,
PATH_DELIM,
                  updatePair, i,c);
         if ( (myoutstream = fopen (myoutstreamfile, READMODE)) == NULL)
         {
            fprintf (outstream,
                     "\nThe output file '%s' for update pair %d node %d
chunk %d could not be opened during runUF2 summing.\n",
                     myoutstreamfile,updatePair,i,c);
            exit (-1);
         }
#endif
      /* 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);
#ifndef TPCD_V2
      /* need to close off loop if we have chunks of data on each node
DELjen*/
      }
#endif
   }

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

}

/* DELjen  added deleteChunk parm to indicate which chunk we are to
delete */
/*         removed existing unused code that calculated splitDeletes */
/*         deleteChunk = 0 means we're coming in in uni/smp mode */
void runUF2_fn ( int updatePair, int i, int deleteChunk, int copyOnOrOff
)
{
   char sourcefile[256];
```

```
   int rc = 0;
   int j;

   long sqlcode;
   int maxwait;

#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;
   char *inlist=NULL;
   char key[32];
   int done = FALSE;

#ifdef TPCD_V2
   sprintf (myoutstreamfile, UF2OUTSTREAMPATTERN, tempdir, 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 runUF2_fn.\n",
               myoutstreamfile,updatePair,i);
      rc=-1;
      goto UF2_exit;
   }

   fprintf( myoutstream,
            "\nUF2 for update pair %d set %d starting at %18.18s\n",
            updatePair, i, get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
#else
   /*DELjen not tpcd_v2 use different outstream name */
   sprintf (myoutstreamfile, UF2OUTSTREAMPATTERN, tempdir, PATH_DELIM,
            updatePair, i,deleteChunk);
   if ( (myoutstream = fopen (myoutstreamfile, WRITEMODE)) == NULL)
   {
      fprintf (stderr,
               "\nThe output file '%s' for update pair %d node %d chunk
%d could not be opened in runUF2_fn.\n",
               myoutstreamfile,updatePair,i,deleteChunk);
      rc=-1;
      goto UF2_exit;
   }

   fprintf( myoutstream,
            "\nUF2 for update pair %d node %d chunk %d starting at
%18.18s\n",

updatePair,i,deleteChunk,get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
   /* endDELjen */
#endif

   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_V2
   if (getenv("TPCD_FLATFILES") != NULL)
      sprintf(sourcefile, "%s%cdelete.%d.%d",
              getenv("TPCD_FLATFILES"),
              PATH_DELIM, updatePair, i);
#else
   sprintf(sourcefile,"%s%cdelete.%d.%05d.%d",     /*DELjen added
deleteChunk */
           getenv("TPCD_FLATFILES"),
           PATH_DELIM, updatePair, i,deleteChunk);
#endif

   if ( (src_fh = fopen (sourcefile, READMODE)) == NULL )
   {
#ifdef TPCD_V2
      fprintf (myoutstream, "\nThe source file '%s' for update pair %d
set %d could not be opened.\n",
               sourcefile, updatePair, i);
#else
      fprintf (myoutstream,
               "\nThe source file '%s' for update pair %d node %d chunk
%d could not be opened.\n",
               sourcefile, updatePair, i, deleteChunk);
#endif
      rc = -1;
      goto UF2_conn_reset;
   }

   if (( inlist = malloc (inlistmax * (16))) == NULL )
   {
#ifdef TPCD_V2
      fprintf (myoutstream, "\nMalloc failed in run_UF2 for update pair
%d set %d.\n",
               updatePair, i);
#else
      fprintf (myoutstream,
               "\nMalloc failed in run_UF2 for update pair %d node %d
chunk %d.\n",
               updatePair, i, deleteChunk);
#endif
      rc = -1;
      goto UF2_conn_reset;
   }

   while ( ! done )
   {
      /* read inlistmax keys from the delete source file in */
      *inlist = '\0';
      for (j=0; j < inlistmax; j++)
      {
         fscanf (src_fh, "%s\n", key);
         if (j) strcat (inlist, ",");
         strcat (inlist, key);
         if (feof (src_fh))
         {
            done = TRUE;
            fclose (src_fh);
            break;
         }
      }
      strcat (inlist, ")");
      if (strlen (inlist) > sizeof(stmt_str) - 100)
      {
         fprintf (myoutstream, "In runUF2_fn: reduce inlistmax and
recompile\n");
         rc = -1;
         goto UF2_conn_reset;
      }
#ifdef TPCD_V2
      strcpy (stmt_str, "delete from tpcd.orders where o_orderkey in
(");
#else
      strcpy (stmt_str, "delete from tpcd.orders where
nodenumber(o_orderkey) = current node and o_orderkey in (");
#endif
      strcat (stmt_str, inlist);
      /*       fprintf (stderr, "stmt_str = %s\n", stmt_str); */

      /* wlc  062797 */
      sqlcode = SQL_RC_E911;
      maxwait = 1;

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

         sqlcode = 0;
#ifdef TPCD_PREPARETIME
         /* temp code to test timing START*/
         fprintf( myoutstream,
            "\nUF2 orders pair %d node %d chunk %d prep starting at
%18.18s\n",

updatePair,i,deleteChunk,get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
         EXEC SQL PREPARE DELO FROM :stmt_str;
         fprintf( myoutstream,
            "\nUF2 orders pair %d node %d chunk %d prep finished at
%18.18s\n",

updatePair,i,deleteChunk,get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
         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 DELO;
         fprintf( myoutstream,
            "\nUF2 Orders pair %d node %d chunk %d execute finished at
%18.18s\n",

updatePair,i,deleteChunk,get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
         /* temp code to test timing STOP   unifdef next line*/
#else
         EXEC SQL EXECUTE IMMEDIATE :stmt_str;
#endif

         if (sqlca.sqlcode < 0)
            sqlcode = error_check();        /* Don't bother printing the
SQL0100W
                                               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...Retrying...\n");
#ifndef SQLWINT
            sleep (UF_DEADLOCK_SLEEP);
#else
            Sleep (UF_DEADLOCK_SLEEP*1000);           /* 10 Seconds Des
*/
#endif
            maxwait++;                                /* jen DEADLOCK */
         }
      }

      if (sqlcode < 0)
      {
         fprintf (myoutstream,"\nAn error occurred deleting from
TPCD.ORDERS\n");
         fprintf (myoutstream,
                  "for update pair number %d node %d chunk
%d..Exiting\n",
                  updatePair, i,deleteChunk);
         if (sqlcode == -911)
         {
            fprintf (myoutstream,"# of deadlocks exceeds %i\n",
MAXWAIT);
         }
         rc=-1;
         goto UF2_rollback;
      }
      /* report the number of row deleted */
      fprintf(myoutstream, "%ld rows deleted from TPCD.ORDERS at
%18.18s\n",
              sqlca.sqlerrd[2],
              get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));

#ifdef TPCD_V2
      strcpy (stmt_str, "delete from tpcd.lineitem where l_orderkey in
(");
```

```
#else
      strcpy (stmt_str, "delete from tpcd.lineitem where
nodenumber(l_orderkey) = current node and l_orderkey in (");
#endif
      strcat (stmt_str, inlist);
      /*       fprintf (stderr, "stmt_str = %s\n", stmt_str); */

      /* wlc 062797 */
      sqlcode = SQL_RC_E911;
      maxwait = 1;

      while (sqlcode == SQL_RC_E911 && maxwait <= MAXWAIT)
      {
         sqlcode = 0;
#ifdef TPCD_PREPARETIME
         /* temp code to test timing START*/
         fprintf( myoutstream,
            "\nUF2 Lineitem pair %d node %d chunk %d prep starting at
%18.18s\n",

updatePair,i,deleteChunk,get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
         EXEC SQL PREPARE DELL FROM :stmt_str;
         fprintf( myoutstream,
            "\nUF2 Lineitem pair %d node %d chunk %d prep finished at
%18.18s\n",

updatePair,i,deleteChunk,get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
         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 DELL;
         fprintf( myoutstream,
          "\nUF2 Lineitem pair %d node %d chunk %d execute finished at
%18.18s\n",

updatePair,i,deleteChunk,get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
         /* temp code to test timing STOP   unifdef next line*/
#else
         EXEC SQL EXECUTE IMMEDIATE :stmt_str;
#endif

         if (sqlca.sqlcode < 0)
            sqlcode = error_check();    /* Don't bother printing the
SQL0100W
                                           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...Retrying...\n");
#ifndef SQLWINT
            sleep (UF_DEADLOCK_SLEEP);
#else
            Sleep (UF_DEADLOCK_SLEEP*1000);           /* 10 Seconds Des
*/
#endif
            maxwait++;                                /* jen DEADLOCK */
         }
      }

      if (sqlcode < 0)
      {
         fprintf (myoutstream,"\nAn error occurred deleting from
TPCD.LINEITEM\n");
         fprintf (myoutstream,"for update pair number %d node %d chunk
%d..Exiting\n",
                  updatePair, i, deleteChunk);
         if (sqlcode == -911)
         {
            fprintf (myoutstream,"# of deadlocks exceeds %i\n",
MAXWAIT);
         }
         rc=-1;
         goto UF2_rollback;
      }
      /* report the number of row deleted */
      fprintf(myoutstream, "%ld rows deleted from TPCD.LINEITEM at
%18.18s\n",
              sqlca.sqlerrd[2],
              get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));

      rc=0;

      EXEC SQL COMMIT WORK;
      error_check();
   }
   /* if we drop out of the loop successfully, then skip the rollback
971103jen*/
   goto UF2_conn_reset;             /*successful skip rollback
971101jen*/

UF2_rollback:                                                  /*
971021jen */
   EXEC SQL ROLLBACK WORK;
   error_check();                                             /* @d22275
tjg */


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

UF2_exit:
   if (inlist) free (inlist);
   fclose (myoutstream);
   /* exiting, increment the semaphore */
#ifdef TPCD_V2
#ifndef SQLWINT
   /* we used the first flat file to generate the semaphore key */
   if (getenv("TPCD_FLATFILES") != NULL)
      sprintf(sourcefile, "%s%cdelete.%d.0",
              getenv("TPCD_FLATFILES"), PATH_DELIM, updatePair);
   else
      sprintf(sourcefile,".%cdelete.%d.0", 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",
                 i, errno);
         exit(1);
      }
   }
   if (sem_op (su_semid, 0, 1) != 0 )
/*jenSEM*/
   {
/*jenSEM*/
      fprintf(stderr,"Failure to increment semaphone UF2 set %d\n",i);
      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);

      Sleep(1000);
   }

   if (! ReleaseSemaphore(su_hSem,
                          1,
                          (LPLONG)(NULL)))
   {
      fprintf(stderr, "ReleaseSemaphore failed, LastError: %d, quit\n",
              GetLastError());
      exit(-1);
   }
#endif
#endif /* TPCD_V2 */
   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",getenv("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 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 */



   while ((sqlcode == SQL_RC_E911) && (max_wait < MAXWAIT)) {

      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;

         if (error_check() < 0) {
            fprintf (stderr,"\nA serious error condition
occurred...Exiting\n");
            exit (-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;
         error_check();

/*********************************************************************
/
/* 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 == 0) {
               rows_fetch++;
               if ( (rows_fetch <= s_info_ptr->max_rows_out) ||
                    (s_info_ptr->max_rows_out == -1) )
                  echo_sqlda(sqlda,col_lengths);
            }

         }  while ( (sqlca.sqlcode == 0) && \
                    ( (s_info_ptr->max_rows_fetch == -1) || \
                      (rows_fetch < s_info_ptr->max_rows_fetch) ) );

      } /** End of block for handling SELECT statements **/

      else {           /** SQL statement is not a SELECT **/


         EXEC SQL EXECUTE IMMEDIATE :stmt_str;
      }                                                         /* @d22817
tjg */

      if ( (sqlca.sqlcode < 0) ||
           ( (sqlca.sqlcode > 0) && (rows_fetch == 0) ) )
         sqlcode = error_check();            /* Don't bother printing
the SQL0100W
                                                error from going through
the loop
                                                above one extra time */

      if (sqlcode == SQL_RC_E911) {          /* we've hit a deadlock */
         max_wait ++;
         fprintf (stderr, "\nA deadlock has been
detected...Retrying...\n");
#ifndef SQLWINT
         sleep (10);
#else
         Sleep (10000);           /* 10 Seconds Des */
#endif
      }
   }

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


}

/*********************************************************************/
/* 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;

#ifdef 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_1,(time_t)NULL) );

   /* write the start timestamp to the file */
   fprintf( outstream,"\n\nStop timestamp %18.18s \n",
            s_info_ptr->end_stamp);



   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);

   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 %18.18s\n",

get_time_stamp(T_STAMP_FORM_1,g_struct->stream_end_time));
      }
      else
      {
         /* query streams */
         fprintf(g_struct->stream_report_file,
                 "Stream number %d stopping at %18.18s\n",
                 g_struct->c_l_opt->intStreamNum,

get_time_stamp(T_STAMP_FORM_1,g_struct->stream_end_time));
      }
      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");
}

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 eq "no" )
{
  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");
}

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");

}

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 load_update

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

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

die $usage if (@ARGV < 2);

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

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

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

# Use the macros in here so that they can handle the platform
differences.
```

```
# macro.pl should be sourced from cmvc, other people wrote and maintain
it.
require "macro.pl";

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

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

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

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

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

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

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

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

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

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

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

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

  system("$s");
}

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


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

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

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

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

if ( $platform eq 'nt' )
{
    die "not implemented on nt yet!\n";
}
else
{
    if ( $function == 1 ) {
        $return = &runInsert;
    }
    elsif ( $function == 2 ) {
        $return = &runDelete;
    }
    else {
        die "unknown function\n";
    }
}
if ($return == 0) {
    die("error occurred in load_update\n");
}
1;
```

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];   /* = "tpcd"; */
EXEC SQL END DECLARE SECTION;

/*--------------------------------------------------------------------*/
/*        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 = getpid(); */
   mypid = acid->tag;
   sprintf(out_fn, "/tmp/tpcd/acidQ.out.%d",mypid);
   out=fopen(out_fn,"a");
   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));

   EXEC SQL SELECT SUM(DECIMAL(L_EXTENDEDPRICE,20,2)*100) 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 = (float)lEprice / 100.00;  /* translate to float 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): %0.2f\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);
   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;
   long int_cost;
   long lfint_cost;
   float fint_cost;

   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 = getpid(); */
      mypid = acid->tag;
      sprintf(out_fn, "/tmp/tpcd/acidT.out.%d",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;
      fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      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;
      fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      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;
      fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      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->termination == 0) {
      EXEC SQL CLOSE L_CURSOR;
      EXEC SQL CLOSE O_CURSOR;
      EXEC SQL COMMIT;
      if (sqlca.sqlcode != 0) {
         if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
         rc = sqlca.sqlcode;
         fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
         sqlerror("acidT: COMMIT", &sqlca);
         goto Terror;
      }
   } */

   if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, before 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;
      fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      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;
      fprintf(out,"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;
      fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      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->termination == 0) {
      EXEC SQL CLOSE L_CURSOR;
      EXEC SQL CLOSE O_CURSOR;
      EXEC SQL COMMIT;
      if (sqlca.sqlcode != 0) {
         if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
         rc = sqlca.sqlcode;
         fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
         sqlerror("acidT: COMMIT", &sqlca);
         goto Terror;
      }
   }

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

   EXEC SQL INSERT INTO tpcd.history values
      (:l_partkey, :l_suppkey, :o_key, :l_key, :delta, CURRENT
TIMESTAMP);
   if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      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 */
   if ( ((random() % (100)) + 1) < 80 ) sleep(1);

   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;
         fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
         sqlerror("acidT: COMMIT", &sqlca);
         goto Terror;
      }
      if (acid->logging) {
         gettimeofday(&tv, &tz);
         time(&timeT);
         fprintf(out,"acidT tag: %d, 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;
         fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
         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);
      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;

   EXEC SQL BEGIN DECLARE SECTION;
   double    acctbal;
   double    discount;
   double    price;
   long      availqty;
   long      size;
   EXEC SQL END DECLARE SECTION;

   qnum = us->qnum;

   /*sprintf(out_fn, "/tmp/tpcd/update.out.%d",qnum); */
   out=fopen("/tmp/tpcd/update.out","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;
             fprintf(out,"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;
             fprintf(out,"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;
             fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                     qnum, i, sqlca.sqlcode);
             sqlerror("update query number 3", &sqlca);
             goto Uerror;
          }
          discount = discount * (-1);
          secs2sleep = 300;
          break;
       }
       case 4:
       {
          if ( i ==1 ) {
             EXEC SQL
                UPDATE TPCD.ORDERS set O_ORDERDATE = O_ORDERDATE - 6
MONTHS
                WHERE O_ORDERKEY =  67461;
             /* WHERE O_ORDERKEY IN
(22400,28515,34338,46596,67461,92644,98307);*/
          } else {
             EXEC SQL
                UPDATE TPCD.ORDERS set O_ORDERDATE = O_ORDERDATE + 6
MONTHS
                WHERE O_ORDERKEY =  67461;
          }
          if (sqlca.sqlcode != 0) {
             rc = sqlca.sqlcode;
             fprintf(out,"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;
             fprintf(out,"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;
             fprintf(out,"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;
             fprintf(out,"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;
             fprintf(out,"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;
             fprintf(out,"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;
            fprintf(out,"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;
            fprintf(out,"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;
            fprintf(out,"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;
            fprintf(out,"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;
            fprintf(out,"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;
            fprintf(out,"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;
            fprintf(out,"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;
            fprintf(out,"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);
   fprintf(out,"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);
   fprintf(out,"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");
fclose(out);
return(rc);
}

/*-------------------------------------------------------------*/
/*        connect_to_TM                                        */
/*-------------------------------------------------------------*/
void connect_to_TM( void )
{
   char *dbname_ptr;
   if ((dbname_ptr = getenv("TPCD_QUAL_DBNAME")) != NULL) {
      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[] = "/tmp/tpcd/acid.err.out";
   int j,k;

   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");
   fclose(err_fp);
}
```