HITACHI
DATA SYSTEMS

TPC Benchmark™ D
Full Disclosure Report

Hitachi Data Systems VisionBase 8460 Server
Using
IBM DB2 UDB 5.2.0
and
Microsoft Windows NT 4.0

Second Edition
Submitted for review
15 February 1999

Second Edition -- 29 April 1999 [server name change]

Hitachi Data Systems, the Sponsor of this benchmark test, believes that the information in this document is accurate as of the publication date. The information in this document is subject to change without notice. The Sponsor assumes no responsibility for any errors that may appear in this document.

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

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

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

Printed in U.S.A, April 29, 1999.

HITACHI
DATA SYSTEMS

Hitachi VisionBase 8460 Server

with

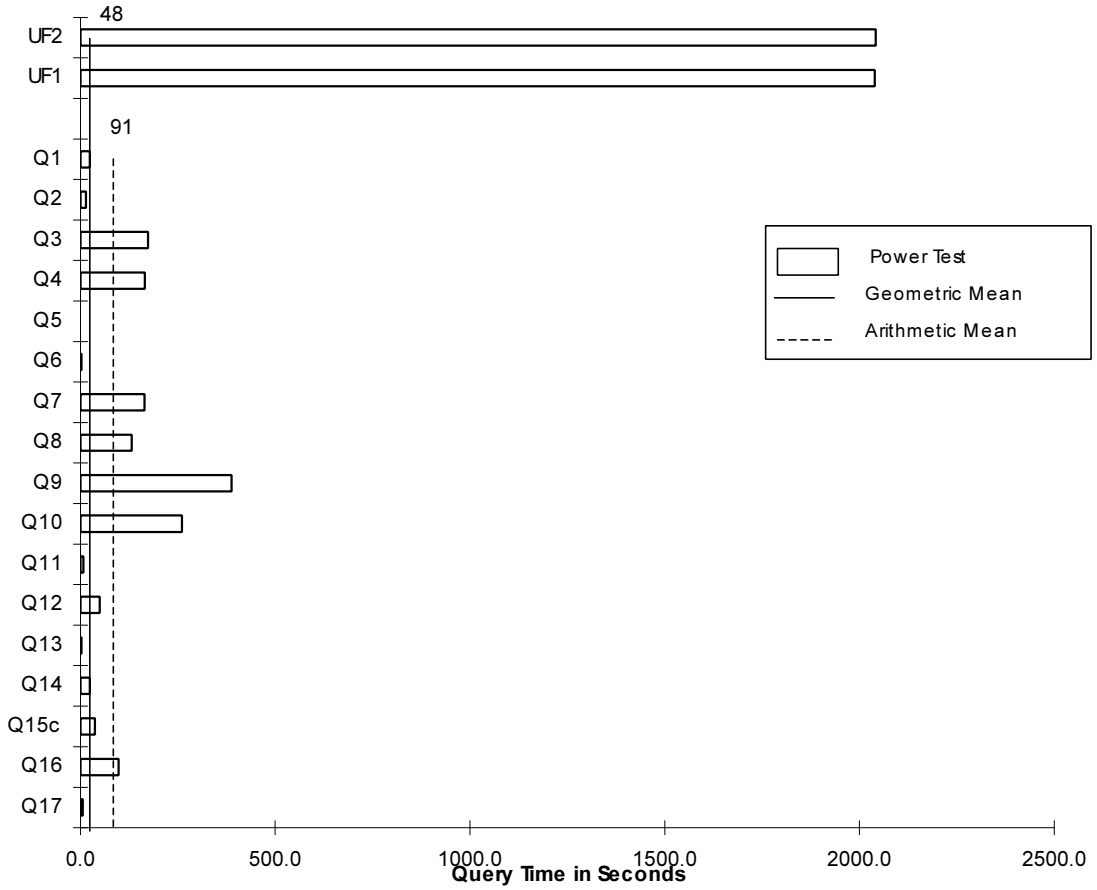
IBM DB2 UDB 5.2.0

TPC-D Rev. 1.3.1

Report Date: 15-Feb-99

Total System Cost	TPC-D Power	TPC-D Throughput	Price/Performance
$199,847	**2,261.2** QppD@30GB	**325.9** QthD@30GB	**$233** QphD@30GB

Database Size	Database Manager	Operating System	Other Software	Availability Date
30GB	**IBM DB2 UDB 5.2.0**	**Microsoft NT 4.0/SP4**	**Microsoft Visual C++ 5.0**	**30-Jun-99**



Database Load Time = 17:08:41	Total Data Storage / Database Size = 19.4	RAID: Y

System Configuration

Processors:	4 X 400MHz Pentium II Xeon Processors w/ 1MB cache
Memory:	2GB Main Memory
Disk Drives:	31 X 17.9GB Disk Drives
	3 X 9GB Disk Drives

HITACHI
DATA SYSTEMS

Hitachi VisionBase 8460 Server

with

IBM DB2 UDB 5.2.0

TPC-D Rev. 1.3.1

Report Date:

15-Feb-99

Description	Part Number	Source	Reference Price	Qty	Disc	Extended Price	5 Yr. Maint. Cost
Server Hardware							
VisionBase 8460 NT Server (512MB, 1 400Mhz Xeon CPU, NT 4.0, Ethernet controller, 3x9G Drive, RAID contoller)	VB-D8460-100	1	21,518	1		21,518	5,995
512 MB memory kit	MEM4-512	1	1,600	3		4,800	-
400 Mhz Xeon CPU (1MB L2 cace)	CPUX400-01M	1	4,050	4		16,200	-
Credit for base CPU (512KB L2)	CPUC400-512	1	-1,653	1		(1,653)	-
Emulex LP7000 Fiber Adapter	LP7000E/N1	1	1,995	2		3,990	-
Command Package (see note 1)	CM620-U	1	450	1		450	-
Subtotal						**45,305**	**5,995**
Storage							
5846 Freedom Storage	5846-A0001.S	1	41,000	1		41,000	23,491
Cache Memory (32 MB)	5800-C0032.S	1	280	32		8,960	-
17.9G Drives (5 pack)	5800-R0003.S	1	7,700	6		46,200	-
Additional Storage Control Unit	5800-F0001.S	1	6,900	1		6,900	-
Fiber Channel Interface Kit	5800-I0004.S	1	5,000	2		10,000	
Duplex Fiber Optic Cable	CBL-FP002.S	1	50	2		100	
Sommet Storage Management Pkg	5800-S0001.S	1	850	1		850	
Sommet Cable (5m)	5800-F0009.S	1	30	1		30	
HDS Volume Discount	(see note 2)				19%	(21,668)	
Subtotal						**92,372**	**23,491**
Hardware Subtotal						**137,677**	**29,486**
Server Software							
Microsoft Windows NT with SP4		1	-	1		-	-
DB2 UDB 5.2.0 Enterprise Edition	(see note 3)	2	24,998	1	13%	21,748	-
Microsoft Visual C++ and Subscription Service		3	460	1		460	
Microsoft Priority Plus Database Server Support		3	2,095	5			10,475
Subtotal						**22,208**	**10,475**

Notes

1. Includes Monitor/Keyboard/Mouse/Cables
2. Based on Sales volume in excess of $100,000
3. IBM Passport/Advantage Discount on DB2 UDB

Sources: (1) Hitachi Data Systems (2) IBM (3) Microsoft

Total		**159,886**	**39,961**
5 Year Cost of Ownership	$	199,847	
QphD@30GB		858	
$/QphD@30GB	**$**	**233**	

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.

HITACHI DATA SYSTEMS

Hitachi VisionBase 8460 Server

with

IBM DB2 UDB 5.2.0

TPC-D Rev. 1.3.1

Report Date: 15-Feb-99

Measurement Results

Scale Factor		=	30
	Total Data Storage / Database Size	=	19.40
	Database Load Time	=	17:08:41
	Query Streams for Throughput Test	=	0
	TPC-D Power Metric(QppD@30GB)	=	2261.2
	TPC-D Throughput Metric(QthD@30GB)	=	325.9
	Composite QphD@30GB	=	858.4
	Total System Price Over 5 Years	=	199847
	TPC-D Price/Performance Metric	=	$ 233

Measurement Intervals

	Measurement Interval in Throughput Test (Ts)	=	5634

Duration of stream execution

Stream ID	Seed	Query Start Date/Time Query End Date/Time	UF1 Start Date/Time UF1 End Date/Time	UF2 Start Date/Time UF2 End Date/Time	Duration
Stream 0	16494	02/13/99 12:38:47 02/13/99 14:12:41	02/13/99 12:38:47 02/13/99 13:12:46	02/13/99 13:38:38 02/13/99 14:12:41	1:33:54

TPC-D Timing Intervals (in seconds):

Query	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Stream 0	23.6	14.1	173.7	165.2	0.5	2.6	164.2	132.0
	Q9	**Q10**	**Q11**	**Q12**	**Q13**	**Q14**	**Q15c**	**Q16**
	388.3	261.4	8.1	49.9	2.9	24.2	37.8	97.8
	Q17	**UF1**	**UF2**					
	6.7	2038.9	2042.3					

Information Paradigm

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL

Certified Auditor

Test Sponsor: Dave Davenport
Vice President, Arch. and Tech. Group
Hitachi Data Systems, Inc.
750 Central Expressway
Santa Clara, CA 95050

February 15, 1999

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

Platform: Hitachi VisionBase 8460
DataBase Manager: IBM DB2 UDB Version 5.2.0
Operating System: Microsoft Windows NT Server Version 4.0/SP4

The results were:

CPU's Speed	Memory	Disks	QppD@30GB	QthD@30GB
Hitachi VisionBase 8460				
4 x Pentium II Xeon (400 Mhz)	1MB L2/CPU 2 GB Main	31 x 17.9 GB 3 x 9.0 GB	2,261.2	325.9

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

- The TIME table was not used
- The input variables were generated by QGEN
- The database was populated using DBGEN
- The database was maintained by the "Evolve" method
- The throughput metric was computed using the results from three query streams
- 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
- The database records were defined with the proper layout and size
- The database was properly scaled to 30GB 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

Abstract

Hitachi Data Systems and International Business Machines Corporation conducted the TPC Benchmark™ D on the Hitachi Data Systems VisionBase 8460 Server. This report documents the full disclosure of the results of that benchmark in accordance with the TPC Benchmark™ D Standard Specification, Revision 1.3.1.

The software used on the Hitachi Data Systems VisionBase 8460 Server included IBM DB2 UDB version 5.2.0 and Microsoft Windows NT 4.0. Hitachi Data Systems 5800's were used to store all the data required by the benchmark and to satisfy the data integrity required by the benchmark standard.

The results achieved are summarized below.

Hardware	Software	Total System Cost	QppD@30GB	QthD@30GB	$/QphD@30GB
Hitachi Data Systems VisionBase 8460 Server	IBM DB2 UDB version 5.2.0 Microsoft Windows NT 4.0	$199,847	2261.2	325.9	$233

The benchmarked configuration will be available on or before June 30, 1999.

The results of the benchmark test were audited by Francois Raab of Information Paradigm. The auditor's attestation letter can be found in Section 9 of this Full Disclosure Report.

Copies of this Full Disclosure Report can be obtained from either the Transaction Processing Performance Council (TPC) or the benchmark sponsors at the following addresses:

TPC
c/o Shanley Public Relations
777 No. First Street, Suite 600
San Jose, CA 95112
(408) 295-8894, FAX (408) 295-9768
http://www.tpc.org

or

Hitachi Data Systems
750 Central Expressway
Santa Clara, California 95050-2627
U.S.A.
(408) 970-1000

Document Structure

The TPC Benchmark™ D Standard Specification requires test sponsors to publish, submit to the TPC, and make available to the public, a full disclosure report for any result considered to be compliant with the specification. The required contents for the full disclosure report are specified in Clause 8.

This report is submitted to satisfy the specification's requirement for full disclosure. It documents the compliance of the benchmark implementation and execution reported for the Hitachi Data Systems VisionBase 8460 Server using IBM DB2 UDB version 5.2.0.

In the specification, the main headings in Clause 8 are keyed to the other clauses. The headings in this report use the same sequence, so that they correspond to the titles or subjects referred to in Clause 8.

Each section in this report begins with the text of the corresponding item from Clause 8 of the specification, printed using italic type. The following plain text explains how this benchmark complies with that specific portion of the specification. In sections where Clause 8 requires extensive listings the appropriate appendix in this report is referenced.

TPC Benchmark™ D Overview

The TPC Benchmark™ D (TPC-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. The 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 that most OLTP transactions;
- Include a rich breadth of operators and selectivity constraints;
- Generate intensive activity on the part of the database server components 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.

Table of Contents

PREFACE VIII

TABLE OF CONTENTS XI

LIST OF TABLES AND DIAGRAMS XIV

0 General Information 1
1 Logical Database Design 3
2 Queries and Update Functions 4
3 Clause 3 Database System Properties 6
4 Clause 4 Scaling and Database Population 13
5 Clause 5 Performance Metrics and Execution Rules 17
6 Clause 6: SUT and Driver Implementation 19
7 Clause 7: Pricing 21
8 Clause 9: Audit 22

APPENDIX A:DBMS AND SYSTEM PARAMETERS 23

A 1: Database TPCD Configuration Parameters 23
A 2: Database Manager Configuration Parameters 24
A 3: NT Parameters 26
A 4: Compiler Options for tpcdbatch 26
A 5: Compiler Options for ACID driver 26
A 6: Dss.scattered-read.config.hitachi.30GB 27

APPENDIX B:DATABASE CREATION STATEMENTS 28

B 1: Buildtpcd 28
B 2: tpcd.setup 38
B 3: dss.dbconfig.load.hitachi.30GB 42
B 4: dss.dbmconfig.load.hitachi 42
B 5: db2set.build.bat 42
B 6: dss.ddl.tbsp.hitachi.30GB 42
B 7: dss.ddl.tables.hitachi.30GB 43
B 8: dss.load30GB.hitachi 44
B 9: dss.ddl.indexes.hitachi 44
B 10: dss.ddl.runstats.hitachi 46
B 11: dss.ddl.ri.hitachi 46
B 12: dss.dbconfig.hitachi.30GB 47
B 13: dss.dbmconfig.hitachi.30GB 47

B 14: DB2SET.RUN.BAT 47
B 15: LOADDUMMYUFDATA.BAT 47
B 16: DOUFLOAD.BAT 47
B 17: LOAD_UF1_DATA 47
B 18: LOAD_UF2_DATA 48
B 19: DELDUMMYUFDATA.SQL 49
B 20: CREATE_UF_STAGING_TABLES 49
B 21: DSS.MULTIPLE.AST 50
B 22: PLOADUF1 51
B 23: PLOADUF2 52

APPENDIX C:QUERY VALIDATION EQT AND OUTPUT 54

C 1: QUERY 1 54
C 2: QUERY 2 54
C 3: QUERY 3 55
C 4: QUERY 4 55
C 5: QUERY 5 55
C 6: QUERY 6 56
C 7: QUERY 7 56
C 8: QUERY 8 56
C 9: QUERY 9 57
C 10: QUERY 10 57
C 11: QUERY 11 58
C 12: QUERY 12 58
C 13: QUERY 13 58
C 14: QUERY 14 58
C 15: QUERY 15 59
C 16: QUERY 16 59
C 17: QUERY 17 59

APPENDIX D:SUBSTITUTION PARAMETERS AND SEEDS 60

D 1: QUERY STREAM SEEDS 60
D 2: QUERY SUBSTITUTION PARAMETERS 60

APPENDIX E:IMPLEMENTATION SPECIFIC LAYER AND DRIVERS 61

E 1: TPCDBATCH.SQC 61
E 2: TPCDUF.SQC 90
E 3: RUNPOWER 96

APPENDIX F:ACID TEST SOURCE CODE 99

F 1: ACID.SQC 99
F 2: ACIDT.SQC 101

F 3: ACID.H 110

APPENDIX G:DATABASE CONTENTS 111

APPENDIX H:PRICE QUOTATIONS 116

List of Tables and Diagrams

Figure 1: Table Cardinalities 13

Figure 2: Disk Usage Summary 14

Figure 3: Flat File Usage Summary 15

Figure 4: Load Test Process Summary 16

Figure 5: Metric Variability 18

0 General Information

0.1 Benchmark Sponsor

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

Hitachi Data Systems is the sponsor of this TPC-D benchmark.

0.2 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:

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

This requirement can be satisfied by providing a full list of all parameters and options.

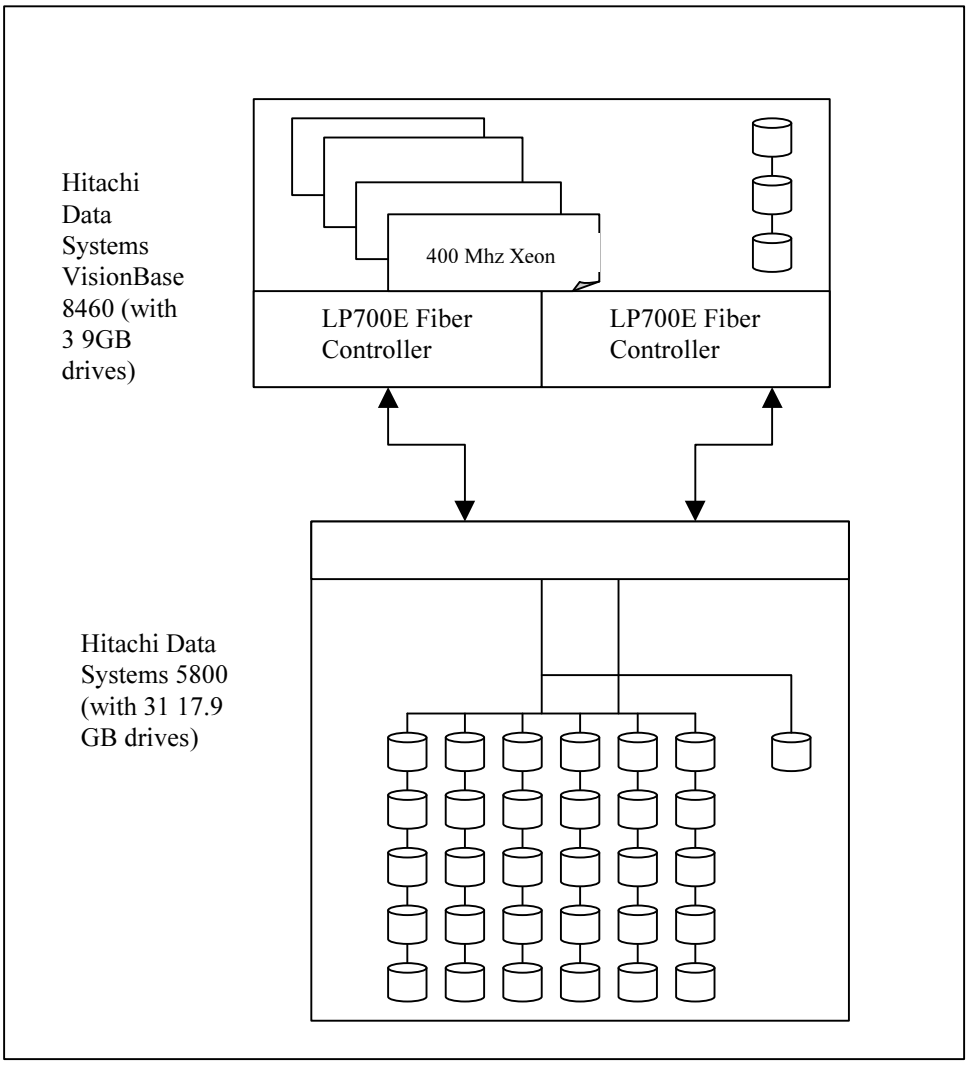
Appendix A: DBMS and System Parameters contains the IBM DB2 UDB and Microsoft Windows NT 4.0 parameters used in this benchmark. Session level parameter settings are included in the source listing for tpcdbatch, and can be found in Appendix E: Implementation Specific Layer and Drivers.

0.3 Configuration Diagrams

Provide diagrams of both the measured and priced configurations, accompanied by a description of the differences. This includes, but is not limited to:

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

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



The priced consisted of a single Hitachi Data Systems VisionBase 8460 Server, configured with 4 Xeon processors running at 400Mhz and including 1MB L2 cache. The system contained 2GB of main memory, 1 Hitachi Data Systems RAID module, managing 2 Fiber controllers and 31 17.9GB disks, and an additional SCSI controller managing 3 9GB disk drives drive. The benchmark configuration was connected to additional Hitachi Data Systems 5800's which were used solely to hold the flat files used in the database load.

1 Logical Database Design

1.1 Database Definition Statements

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 Creation Statements contains the programs and scripts that create and analyze the tables and indexes for the qualification and test databases used in this benchmark.

1.2 Physical Organization

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

No record clustering or index clustering was used in this benchmark. No changes were made to the column ordering presented in Clause 1.4 of the specification. Further details can be found in Appendix B: Database Creation Statements.

1.3 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 not used in this benchmark.

1.4 Replication

While there are some restrictions placed upon the physical replication of objects in the test and qualification database (see Clause 1.5.6), any such replication must be disclosed.

No replication was used in this benchmark beyond the creation of indices and automatic summary tables. The statements and scripts used to create these auxiliary data structures may be found in Appendix B: Database Creation Statements.

2 Queries and Update Functions

2.1 Query Language

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

SQL was used to implement the queries and update functions used in this benchmark.

2.2 Random Number Generation

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

This benchmark used version 1.3.1 of DBGEN and QGEN without modification for random number generation.

2.3 Substitution Parameter 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.

This benchmark used version 1.3.1 of QGEN.

2.4 Query Text and Output Data used for Query Validation

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 query modifications have been applied to any functional query definition or approved variant in order to obtain executable query text, these modifications must be disclosed and justified. The justification for a particular minor query modification can be applies collectively to all queries for which it has been used.)

Appendix C: Query Validation EQT and Output contains the executable query text used during query validation and the resulting output. The following minor query modifications were used to obtain the executable query text that was used for query validation and the performance runs:

- All date expressions were rewritten using equivalent IBM DB2 UDB syntax as allowed by Clause 2.2.3.3c.

- Table and view names are fully qualified. For example, the nation table is referred to as “TPCD.NATION”.

2.5 Substitution Parameters and QGEN Seeds

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

Appendix D: Substitution Parameters and Seeds includes the QGEN seed value and resulting substitution parameters used in the performance tests.

2.6 Query Isolation Level

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

The queries were run at isolation level 2 (as defined in Clause 3.4), Repeatable Read.

2.7 Source Code of Update Functions

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

Appendix E: Implementation Specific Layer and Drivers contains the full source code for the update functions.

2.8 Database Maintenance Option

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

This benchmark used the evolve option. Source code for the associated scripts can be found in Appendix E: Implementation Specific Layer and Drivers.

3 Clause 3 Database System Properties

3.1 ACID Properties

The ACID (Atomicity, Consistency, Isolation and Durability) properties of transaction processing must be supported by the system under test during the timed portion of this benchmark. Since TPC-D is not a transaction processing benchmark, the ACID properties must be evaluated outside the timed portion of the test.

The ACID tests were conducted in accordance with the TPC-D specification. Complete source code for the ACID test is included in Appendix F: ACID Test Source Code.

3.2 Atomicity

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.

3.2.1 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 ORDERS, LINEITEM, and HISTORY tables.

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

1. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for a randomly selected order key. The number of records in the HISTORY table was also retrieved.

2. The ACID Transaction was performed using the order key from Step 1.

3. The ACID Transaction committed.

4. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for the same order key. It was verified that the appropriate rows had been changed, and that a record had been written to the HISTORY table.

3.2.2 Aborted Transaction

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 ORDERS, LINEITEM and HISTORY tables.

1. The ACID application is passed parameter to execute a rollback of the transaction instead of a commit.

2. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for a randomly selected order key. The number of records in the HISTORY table was also retrieved.

3. The ACID Transaction was performed using the order key from Step 1. The transaction was rolled back.

4. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for the same order key. It was verified that the appropriate rows had not been changed.

3.3 Consistency

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

```
SELECT DECIMAL(SUM(DECIMAL(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
```

3.3.1 Consistency Test

Verify that the ORDER and LINEITEM tables are initially consistent, submit the prescribed number of ACID Transactions with randomly selected input parameters, and re-verify the consistency of the ORDERS and LINEITEM tables.

1. The consistency of the ORDERS and LINEITEM tables was verified using the query defined above.

2. 100 ACID Transactions were submitted from each of 2 execution streams.

3. The consistency of the ORDERS and LINEITEM tables was re-verified.

3.4 Isolation

Operation of concurrent transactions must yield results which are indistinguishable from the result which would have been obtained by forcing each transaction to be serially executed to completion in the proper order.

3.4.1 Read-Write Conflict with Commit

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

1. First Session: An ACID Transaction was started for a randomly selected O_KEY, L_KEY and DELTA. The ACID Transaction was suspended for 60 seconds just prior to COMMIT.

2. Second Session: An ACID Query was started for the same O_KEY used in Step 1.

3. Second Session: The ACID query attempts to read the file but is locked out by the ACID transaction waiting to complete.

4. First Session: The ACID Transaction resumed, and executed a COMMIT releasing the record.

5. Second Session: The ACID Query completed. It returned the data as committed by the ACID Transaction.

3.4.2 Read-Write Conflict with Rollback

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

1. First Session: An ACID Transaction was started for a randomly selected O_KEY, L_KEY and DELTA. The ACID Transaction was suspended for 60 seconds just prior to ROLLBACK.

2. An ACID Query was started for the same O_KEY used in Step 1.

3. Second Session: The ACID query attempts to read the file but is locked out by the ACID transaction waiting to complete.

4. First Session: The ACID Transaction resumed, and executed a ROLLBACK WORK.

5. The ACID Query completed. It returned the data as seen prior to the start of the ACID Transaction.

3.4.3 Write-Write Conflict with Commit

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

1. An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY and DELTA, DELTA1. T1 was suspended for 60 seconds just prior to COMMIT.

2. Another ACID Transaction, T2, was started using the same O_KEY and L_KEY and a randomly selected DELTA, DELTA2. The transaction is forced to wait until the requested record is freed by the completion of the first session's transaction.

3. T1 was allowed to COMMIT and T2 completed.

4. It was verified that T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE + (DELTA1 * (T1.L_EXTENDEDPRICE / T1.L_QUANTITY))

3.4.4 Write-Write Conflict with Rollback

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

1. An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY and DELTA, DELTA1. T1 was suspended for 60 seconds just prior to ROLLBACK.

2. Another ACID Transaction, T2, was started using the same O_KEY and L_KEY and a randomly selected DELTA, DELTA2. . The transaction is forced to wait until the requested record is freed by the completion of the first session's transaction.

3. T1 was allowed to ROLLBACK WORK and T2 completed.

4. It was verified that T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE.

3.4.5 Concurrent Progress of Read and Write Transactions

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

1. First Session: An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY and DELTA. T1 was suspended prior to COMMIT.

2. Second Session: Another ACID Transaction, T2, was started and, using randomly selected values of PS_PARTKEY and PS_SUPPKEY, selected all rows in the PARTSUPP table for which match the selected values of PS_PARTKEY and PS_SUPPKEY.

3. T2 completed.

4. T1 was allowed to complete.

5. It was verified that appropriate rows in the ORDERS, LINEITEM and HISTORY tables were changed.

3.4.6 Read-only Query Conflict with Update Transaction

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

Isolation Test 6 specified that Q1 (from TPC-D specification Clause 2.4) must b used as Txn1. However, this query's execution time is too short to be able to complete after Step 2 has completed. A longer-running modification of Q1, called Q1x, was used instead.

Q1x:

```
SELECT
        L_RETURNFLAG, L_LINESTATUS,
        CAST(SUM(L_QUANTITY) AS NUMERIC(31,3)) AS SUM_QTY,
        CAST(SUM(L_EXTENDEDPRICE) AS NUMERIC(31,3))
                AS SUM_BASE_PRICE
```

```
        CAST(SUM(L_EXTENDEDPRCE * (1 – L_DISCOUNT))
                AS NUMERIC(31,3)) AS SUM_DISC_PRICE
        CAST(SUM(L_EXTENDEDPRICE * (1 – L_DISCOUNT)
                * (1 + L_TAX)) AS NUMERIC(31,3))
                        AS SUM_CHARGE
        CAST(AVG(L_QUANTITY) AS NUMERIC(31,3)) AS AVG_QTY,
        CAST(AVG(L_EXTENDEDPRICE) AS NUMERIC(31,3))
                AS AVG_PRICE,
        CAST(AVG(L_DISCOUNT) AS NUMERIC(31,3)) AS AVG_DISC,
        COUNT(*) AS COUNT_ORDER
FROM TPCD.LINEITEM
WHERE L_SHIPDATE <= DATE('1998-12-01') – [DELTA] DAYS
        AND L_QUANTITY !=
                (SELECT COUNT(*) FROM TPCD.LINEITEM AS L2
                WHERE L2.L_DISCOUNT = L_DISCOUNT)
        AND L_DISCOUNT !=
                (SELECT COUNT(*) FROM TPCD.LINEITEM AS L3
                WHERE L3.L_DISCOUNT = L_DISCOUNT)
        AND L_EXTENDEDPRICE !=
                (SELECT COUNT(*) FROM TPCD.LINEITEM AS L4
                WHERE L4.L_QUANTITY = L_QUANTITY)
GROUP BY L_RETURNFLAG, L_LINESTATUS,
ORDER BY L_RETURNFLAG, L_LINESTATUS;
```

The following steps comprised the test:

1. A database session, S1, began an execution of Q1x against the qualification database with a DELTA of 100.

2. While S1 was executing, a second database session, S2, began an ACID Transaction, T1, using randomly selected values for O_KEY, L_KEY and DELTA.

3. Before T1 completed, a third database session, S3, .began another execution of Q1x with a randomly selected DELTA (not equal to 0).

4. T1 waited for the first execution of Q1 by S1 to complete, and then completed its ACID Transaction. It was verified that the appropriate rows in the ORDER, LINEITEM and HISTORY tables had been changed.

5. The second execution of Q1 by S1 waited for T1 to complete and then it proceeded to completion.

3.4.7 Isolation of Automatic Summary Tables

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

1. First session: Verify that the AST is not defined and execute the ACID query. Verify that the ACID query access wevery row of the LINEITEM table.

2. Second session: Create the AST (and bind the SQL) and execute the ACID query. Verify that the ACID query accesses theAST and gets the same results as when it accessed the LINEITEM table.

3. Third session: Start an ACID transaction with a randomly selected O_KEY, L_KEY and DELTA that will update the LINEITEM table (and automatically refresh the AST). The transaction is delayed for 60 seconds just prior to the commit.

4. Fourth session: Start the same ACID query as in the first session.

5. Fourth session: The ACID query attempts to read from the AST but is lock out by the ACID transaction waiting to complete.

6. Third session: The ACID transaction is released and the Commit is executed releasing the record. With the LINEITEM record and the associated AST record(s) now released, the ACID query can now comlete.

7. Fourth session: Verify that the ACID query delays for approximately 60 seconds, that it accesses the AST, and that the result is different from the results in the fist and second sessions.

8. Fifth session: Drop the AST (and bind SQL) and execute the same ACID query. Verify that the ACID query accessses every row of the base tale and gets the same answer as when it accessed the AST (in the fourth session).

3.5 Durability

The tested system 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.

3.5.1 Failure of a Durable Medium

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

3.5.2 System Crash and Memory Failure

Guarantee the database and committed updates are preserved across an instantaneous interruption (system crash/system hang) in process which requires the system to reboot to recover… Guarantee the database and committed updates are preserved across failure of all or part of memory (loss of contents).

The system durability, system crash and memory failure tests were combined. The following steps were executed:

1. The consistency of the qualification database was verified.

2. The current count of the total number of records in the HISTORY table was determined (hist1).

3. A test to run 100 ACID transactions on each of 6 execution streams was started.

4. After more than 100 transactions had completed, one of the disks in the RAID parity group containing the DB2 transaction logs was disabled. Transaction continued to complete successfully and the HDS 5800 began to rebuild the parity group simultaneously on the installed "hot spare", while data integrity was assured by the 4 remaining drives in the parity group.

5. One of the disks in a RAID parity group containing database tables was disabled. Transactions continued, with database integrity assured by the remaining four disks in the parity group.

6. Power was removed from both the server and the RAID arrays, causing the system to crash.

7. The power was restored and the system and database were restarted.

8. The disks removed in steps 4 and 5 were restored and their re-integration into their respective RAID parity groups began automatically.

9. Step 2 was repeated to provide an update count of the records in the HISTORY table (hist2). It was verified that this count was larger than the original hist1 and met or exceed the number of records in the success file.

10. This consistency of the database was verified again.

4 Clause 4 Scaling and Database Population

4.1 Table Cardinalities

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

Table	Rows
Order	45,000,000
Lineitem	#########
Customer	4,500,000
Part	6,000,000
Supplier	300,000
Partsupp	24,000,000
Nation	25
Region	5

Figure 1: Table Cardinalities

4.2 Distribution of Tables and Logs Across Media

The distribution of tables and logs across all media must be explicitly described.

This benchmark used RAID 5 parity groups to hold all database files and logs, as well as all DBMS and OS executables. The Hitachi Data Systems 5800's configured in the system contains 6 parity groups comprised of 5 17.9 GB disk drives, and a single "hot spare" which was also a 17.9GB disk drive. An additional 5800 contained only flat-file data, and was not priced. The Hitachi Data Systems VisionBase 8460 Server includes 3 17.9GB drives configured as a single RAID 5 parity group. The allocation of disk to the various tasks is detailed in the table below.

Drive	Spindles Count	Size	Usage
C:	3	9	OS, DBMS, Benchmark Tools
G:	5	17.9	DBMS logs
17	1.67	17.9	Lineitem Table
20	1.67	17.9	Lineitem Table
16	1.67	17.9	Lineitem Index
24	1.67	17.9	Lineitem Index
19	1.67	17.9	Lineitem Index
27	1.67	17.9	Lineitem Index
22	2.5	17.9	Lineitem Index
18	1.67	17.9	Other Tables and Indexes
26	1.67	17.9	Other Tables and Indexes
21	1.67	17.9	Other Tables and Indexes
29	1.67	17.9	Other Tables and Indexes
23	1.67	17.9	Other Tables and Indexes
25	1.67	17.9	Temporary Tables
28	1.67	17.9	Temporary Tables
Spare	1	17.9	
Total	**34**	**581.9**	

Figure 2: Disk Usage Summary

4.3 Database Partition/Replication Mapping

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

The database used in this benchmark employed no replication, beyond the employment of indexes and automatic summary tables as allowed by the specification under its definition of auxiliary data structures.

4.4 RAID Usage

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

RAID 5 was used for all database tables, logs and indexes. Refer to "4.2 Distribution of Tables and Logs Across Media" for details.

4.5 Modifications to DBGEN

Any modification 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 supplied DBGEN (version 1.3.1) was used without modification.

4.6 Database Contents Validation

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

Appendix G: , "Database Contents" contains the first 10 rows of each table in the test database.

4.7 Database Load Time

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

The database load time was 17:08:41.

4.8 Data Storage Ratio

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

The data storage ratio reported in the Executive Summary in the Preface to this report was based on the information provided in Section 4.2, "Distribution of Tables and Logs Across Media":

4.9 Database Load Description

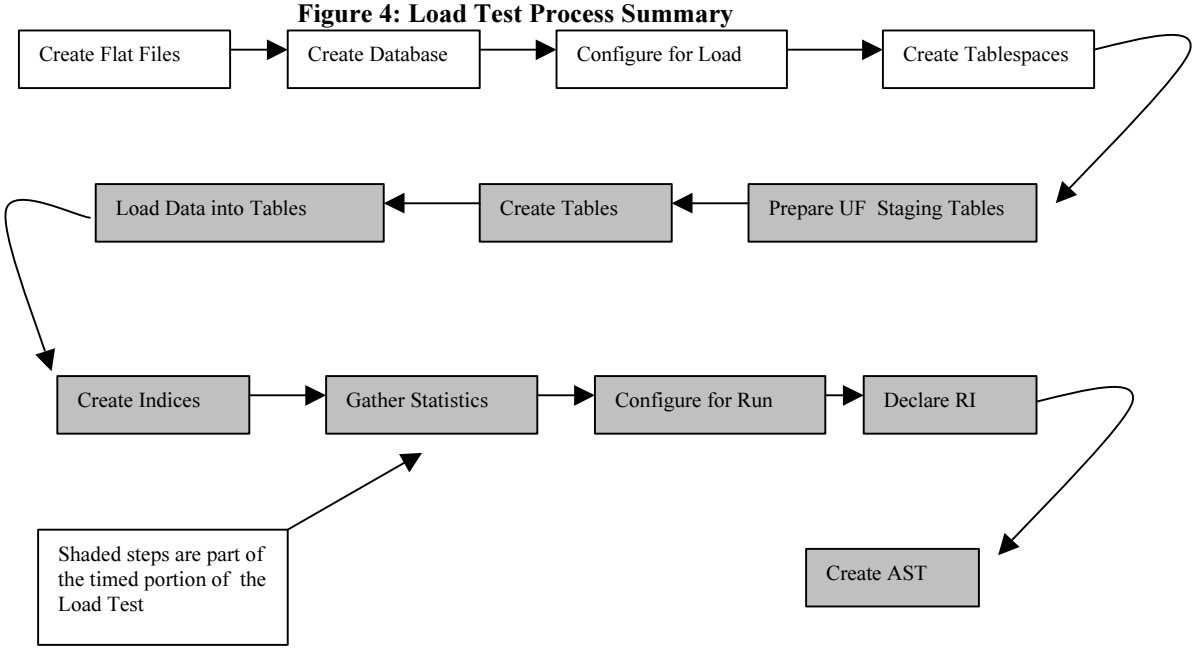
The details of the database load mechanism must be described and illustrated with a block diagram.

The database was loaded using the DB2 LOAD command, which used flat files generated by DBGEN as input. The same mechanism was used to load both the qualification database and the test database. The number of flat files used to load each table within each data set is detailed below, along with a depiction of the complete load process. All scripts and command files used during the load are contained in Appendix B: , "Database Creation Statements".

Table	Qualification	Test
NATION	1	10
REGION	1	10
PART	1	10
PARTSUPP	1	10
ORDERS	1	10
LINEITEM	1	10
CUSTOMER	1	10
SUPPLIER	1	10

Figure 3: Flat File Usage Summary

Figure 4: Load Test Process Summary



5 Clause 5 Performance Metrics and Execution Rules

5.1 Power Test Overview

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

The following procedure was used to implement the power test:

1. Database Restart
2. UF1 Update Function Executed
3. Query Stream 00 Executed
4. UF2 Update Function Executed

tpcdbatch is used to submit all queries and is disclosed in Appendix E: Implementation Specific Layer and Drivers.

5.2 Power Test Timing Intervals

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

The timing intervals for each of the queries in the Power Test are disclosed as part of the Numerical Quantities Summary contained in the Executive Summary in the preface of this report.

5.3 Throughput Test Configuration

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

The throughput test was not executed during this benchmark. The timing intervals from the power test were used to calculate the throughput metric as allowed by Clause 5.3.1.4.

5.4 Query Stream Timings

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

The start and end time for each query stream are contained in the Numerical Quantities Summary of the Executive Summary in the preface of this report.

5.5 Elapsed Time of the Measurement Interval

The total elapsed time of the measurement interval must be disclosed.

The total elapsed time of the measurement interval is contained in the Numerical Quantities Summary of the Executive Summary in the preface of this report.

5.6 Timing Intervals for the Throughput Test

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

Since this benchmark did not execute the throughput test, the timing intervals for the power test were used to calculate the throughput metric. The timing intervals for each query and update function in the power test are contained in the Numerical Quantities Summary of the Executive Summary in the preface of this report.

5.7 Performance Metrics

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

The performance metrics, the price performance metric and all the underlying data on which they are based are contained in the Numerical Quantities Summary of the Executive Summary in the preface of this report.

5.8 Metric Reproducibility

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

Performance results from consecutive runs of the performance test revealed the following variability in the performance metrics:

	Run 1	Run 2	Reported
30 GB	2351.7	2261.2	**2,261.22**
30 GB	331.9	325.9	**325.88**
30 GB	883.5	858.4	**858.42**

Figure 5: Metric Variability

6 Clause 6: SUT and Driver Implementation

6.1 Driver Overview

A detailed textual description of how the driver performs its functions, how its various components interact and any product functionality or environmental setting on which it relies must be disclosed.

Appendix E: Implementation Specific Layer and Drivers contains the source code used for the driver and all scripts used in conjunction with it.

The power test in invoked by calling tpcdbatch with the stream number 0 (to indicate the execution must include the update functions), as well as the name of the SQL file that contains the proper permutation of the TPC-D queries.

6.2 Implementation Specific Layer Overview

If an implementation specific layer is used, then a detailed textual description of how the driver performs its functions, how its various components interact and any product functionality or environmental setting on which it relies must be disclosed.

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 TPCD-D queries, although it is completely capable of processing any arbitrary SQL statement (both DML and DDL).

A separate instance of tpcdbatch is invoked for each stream. Each instance establishes a distinct connection to the database server through which the EQT is transmitted to the database and the results are returned through the implementation-specific layer to the driver. When an instance of tpcdbatch is invoked, its command line arguments determine whether it will be running a power test, query stream or update stream, as well as the input file from which it will draw the permutation of queries or updates that it is to execute. Upon invocation, tpcdbatch validates its command line arguments and the existence of any required files, then connects to the database, performs any required session initialization and gathers any data required for the auditor. It then traverses its input file, executing each query or update function in turn.

For query executions, each query is prepared, described, and a cursor is opened and used to fetch the required number of rows. After the last row is retrieved, a commit is issued. For UF1, the data is allocated to one of the *n* concurrent updates that will be launched by tpcdbatch to execute a single UF1, and staged in two separate tables, one of ORDERS and one for LINEITEM. For UF2, the ORDERKEYS to be deleted are similarly allocated and staged. During the run when tpcdbatch is called upon to execute a UF1, it calls a script that loads this data into their staging tables, and then forks off children to execute equal portions of the work required, using a subselect from the staging tables.

The complete source code for the driver script and all related non-commercial tools is provided in Appendix Appendix E: , “Implementation Specific Layer and Drivers”.

Since standard Informix tools provide all the functionality needed to execute the benchmark, no implementation specific layer was required.

6.3 Profile-directed Optimization

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

No profile-directed optimization was used in this benchmark.

7 Clause 7: Pricing

7.1 Pricing Summary

A detail list of hardware and software used in the priced system must be reported.

Please refer to the Pricing Spreadsheet found in the executive summary at the beginning of this report. Required substantiating price quotations are included in Appendix H: Price Quotations.

7.2 Total System Cost

The total 5-year price of the entire configuration must be reported, including hardware, software and maintenance charges.

Please refer to the Pricing Spreadsheet found in the executive summary at the beginning of this report.

7.3 System Availability

The committed delivery date for general availability (availability date) of products used in the priced calculations must be reported.

All components of the System Under Test referred to this in this report are now genarlly available with one exception. IBM DB2 UDB 5.2.0 is currently generally available, but some of the Automatic Summary Table functionality employed during this benchmark will not be generally available until June 30, 1999. As a result, the System Availability Date (as reported in the Executive Summary in the preface) is June 30, 1999.

7.4 Country-Specific Pricing

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

The configuration is priced for the United States of America.

8 Clause 9: Audit

8.1 Attestation Letter

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.

The attestation letter for this benchmark is included at the beginning of this report.

8.2 Availability of the Full Disclosure Report

The Full Disclosure Report must e readily available to the public at a reasonable charge, similar to the charges for similar documents by the test sponsor. The report must be made available when results are made public. In order to use the phrase "TPC Benchmark™ D", the Full Disclosure Report must have been submitted to the TPC Administrator as well as written permission obtained to distribute same.

Requests for the TPC Benchmark™ D Full Disclosure Report should be sent to:

Transaction Processing Performance Council
c/o Shanley Public Relations
777 North First Street, Suite 600
San Jose, CA 95112-6311

Or

Hitachi Data Systems
750 Central Expressway
Santa Clara, CA 95050-2627

Appendix A: DBMS and System Parameters

A 1: Database TPCD Configuration Parameters

**** Denotes a change from the default parameter setting***

Database Configuration for Database tpcd

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

 Database territory                                  = US
 Database code page                                  = 1252
 Database code set                                   = IBM-1252
 Database country code                               = 1

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

*  Degree of parallelism                (DFT_DEGREE) = 20
 * Default query optimization class     (DFT_QUERYOPT) = 5
 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                      = NO
 User exit for logging status                        = NO

 Datalink Access Token Expiry Interval (sec) (DL_EXPINT) = 60
 Datalink Number of Copies              (DL_NUM_COPIES) = 1
 Datalink Number of Backups             (DL_NUM_BACKUP) = 3
 Datalink Time after Drop (days)        (DL_TIME_DROP) = 1

* Database heap (4KB)                   (DBHEAP) = 6654
* Catalog cache size (4KB)              (CATALOGCACHE_SZ) = 386
* Log buffer size (4KB)                 (LOGBUFSZ) = 64
* Utilities heap size (4KB)             (UTIL_HEAP_SZ) = 40000
* Buffer pool size (4KB)                (BUFFPAGE) = 20000
 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) = 642

* Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 256

* Sort list heap (4KB)                  (SORTHEAP) = 50000
* SQL statement heap (4KB)              (STMTHEAP) = 4096
* Default application heap (4KB)        (APPLHEAPSZ) = 768
* Package cache size (4KB)              (PCKCACHESZ) = 320
* Statistics heap size (4KB)            (STAT_HEAP_SZ) = 4384

 Interval for checking deadlock (ms)    (DLCHKTIME) = 10000
 Percent. of lock lists per application (MAXLOCKS) = 75
 Lock timeout (sec)                     (LOCKTIMEOUT) = -1

* Changed pages threshold               (CHNGPGS_THRESH) = 10
* Number of asynchronous page cleaners  (NUM_IOCLEANERS) = 20
* Number of I/O servers                 (NUM_IOSERVERS) = 20
```

```
 Index sort flag                        (INDEXSORT) = YES
 Sequential detect flag                 (SEQDETECT) = YES
 Default prefetch size (4KB)            (DFT_PREFETCH_SZ) = 16

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

* Max number of active applications      (MAXAPPLS) = 40
 Average number of active applications  (AVG_APPLS) = 1
 Max DB files open per application      (MAXFILOP) = 64

* Log file size (4KB)                   (LOGFILSIZ) = 4094
* Number of primary log files           (LOGPRIMARY) = 100
* Number of secondary log files         (LOGSECOND) = 25
* Changed path to log files             (NEWLOGPATH) =
* Path to log files                              = g:\logs\
 Next active log file                           =
 First active log file                          =

* Group commit count                    (MINCOMMIT) = 1
* Percent log file reclaimed before soft chckpt (SOFTMAX) = 100
* Log retain for recovery enabled       (LOGRETAIN) = OFF
 User exit for logging enabled          (USEREXIT) = OFF

 Auto restart enabled                   (AUTORESTART) = ON
 Index re-creation time                 (INDEXREC) = SYSTEM (ACCESS)
 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: Database Manager Configuration Parameters

**** Denotes a change from default parameter setting***

```
Database Manager Configuration

   Node type = Database Server with local and remote clients

 Database manager configuration release level           = 0x0800

 Maximum total of files open            (MAXTOTFILOP) = 16000
* CPU speed (millisec/instruction)       (CPUSPEED) = 2.600000e-006

 Max number of concurrently active databases    (NUMDB) = 8
 Transaction processor monitor name     (TP_MON_NAME) =

 Default charge-back account        (DFT_ACCOUNT_STR) =

 Java Development Kit 1.1 installation path (JDK11_PATH) =

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

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

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

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

Default database path                    (DFTDBPATH) = C:

Database monitor heap size (4KB)          (MON_HEAP_SZ) = 24
UDF shared memory set size (4KB)          (UDF_MEM_SZ) = 256
Java Virtual Machine heap size (4KB)     (JAVA_HEAP_SZ) = 512
Audit buffer size (4KB)                (AUDIT_BUF_SZ) = 0

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

* Agent stack size                    (AGENT_STACK_SZ) = 16
Minimum committed private memory (4KB)   (MIN_PRIV_MEM) = 32
Private memory threshold (4KB)        (PRIV_MEM_THRESH) = 1296

* Sort heap threshold (4KB)              (SHEAPTHRES) = 100000

Directory cache support                  (DIR_CACHE) = YES

* Application support layer heap size (4KB)   (ASLHEAPSZ) = 15
* Max requester I/O block size (bytes)        (RQRIOBLK) = 32767
DOS requester I/O block size (bytes)     (DOS_RQRIOBLK) = 4096
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) = 1000
* Agent pool size                    (NUM_POOLAGENTS) = 4
*Initial number of agents in pool       (NUM_INITAGENTS) = 4
Max number of coordinating agents     (MAX_COORDAGENTS) = (MAXAGENTS - NUM_INITAGENTS)
Max no. of concurrent coordinating agents  (MAXCAGENTS) = MAX_COORDAGENTS

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

Index re-creation time                   (INDEXREC) = ACCESS

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

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

NetBIOS Workstation name                    (NNAME) =

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) =
Default client adapter number         (DFT_CLIENT_ADPT) = 0
```

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

* No. of int. communication buffers(4KB)(FCM_NUM_BUFFERS) = 1024
* Number of FCM request blocks              (FCM_NUM_RQB) = 512
 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)
```

A 3: NT Parameters

```
System Environment Variables
  BOOKSHELF=C:\ifor\WIN\BIN\EN_US
  CLASSPATH=.;C:\SQLLIB\java\db2java.zip;C:\SQLLIB\java\runtime.zip;C:\SQLLIB\java\sqlj.zip;
  ComSpec=C:\WINNT\system32\cmd.exe
  HELP=C:\ifor\WIN\BIN
  HOME=C:\
  I4_INSTALL_DRIVE=C:
  I4_LANG=EN_US
  IPF_PATH32=C:\ifor\WIN\BIN\EN_US
  NLSPATH=C:\ifor\LS\MSG\%L\%N
  NUMBER_OF_PROCESSORS=4
  OS=Windows_NT
  Os2LibPath=C:\WINNT\system32\os2\dll;
  Path=C:\Perl\5.00502\bin\MSWin32-x86-
object;C:\Perl\5.00502\bin;c:\sqllib\misc;C:\Perl\54A85~1.005\bin\MSWIN3~1;C:\Perl\54A85~1.005\bin;C:\WINNT\system32;C:\W
INNT;C:\ifor\WIN\BIN;C:\ifor\WIN\BIN\EN_US;w:\Nucleus\run;C:\IMNNQ_NT;C:\SQLLIB\BIN;C:\SQLLIB\FUNCTION;C:\SQL
LIB\SAMPLES\REPL;C:\SQLLIB\HELP
  PROCESSOR_ARCHITECTURE=x86
  PROCESSOR_IDENTIFIER=x86 Family 6 Model 5 Stepping 2, GenuineIntel
  PROCESSOR_LEVEL=6
  PROCESSOR_REVISION=0502
  ROOTDIR=C:\
  SHELL=C:\winnt\system32\cmd.exe
  TMPDIR=C:\TEMP
  windir=C:\WINNT
  IMNINSTSRV=C:\IMNNQ_NT
  IMNINST=help
  DB2INSTANCE=DB2
  LIB=C:\SQLLIB\LIB
  INCLUDE=C:\SQLLIB\INCLUDE
```

A 4: Compiler Options for tpcdbatch

```
erase tpcdUF.c
erase tpcdUF.obj
erase tpcdbatch.c
erase tpcdbatch.obj
erase tpcdbatch.map

set db2options=+c -t +p -v
db2start
db2 connect to TPCD
db2 prep tpcdbatch.sqc bindfile package isolation rr blocking all datetime iso
db2 prep tpcdUF.sqc    bindfile package isolation rs blocking all datetime iso
db2 connect reset
db2 terminate
REM make sure LIBPATH is set to include the compiler libraries and db2 libraries
cl -c -Z7 -G6 -DTPCD_NONPARTITIONED -DWIN32 -DSQLWINT -D_X86_=1 -W3 -J tpcdbatch.C
cl -c -Z7 -G6 -DTPCD_NONPARTITIONED -DWIN32 -DSQLWINT -D_X86_=1 -W3 -J tpcdUF.C
link -debug -out:tpcdbatch.exe tpcdbatch.obj tpcdUF.obj -align:0x1000 user32.lib kernel32.lib db2api.lib -subsystem:console
```

A 5: Compiler Options for ACID driver

```
db2start
db2 connect to tpcdqual
db2 prep acid.sqc bindfile package isolation rr
db2 prep acidT.sqc bindfile package isolation rs
db2 connect reset
```

```
db2 terminate

cl -c -Zi -DTPCD_V2 -DWIN32 -DSQLWINT -D_X86_=1 -W3 -J mainacid.c acid.c acidT.c

link -debug -out:mainacid.exe commode.obj mainacid.obj acid.obj acidT.obj -align:0x1000 "user32.lib" "kernel32.lib"  db2api.lib -
subsystem:console
```

A 6: Dss.scattered-read.config.hitachi.30gb

```
# turn on scatter read for these types
NT_SCATTER_SMS=1
NT_SCATTER_DMSFILE=1
NT_SCATTER_DMSDEVICE=1

#diaglevels at which it writes configuration
LOG_CFG=4
```

Appendix B: Database Creation Statements

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 ;
&fvt_redirect_output("build.out");

# verify that necessary environment variables for building the
database
# are present.  Default those that aren't necessary
require "version";
$instance=$ENV{"DB2INSTANCE"};
if (length($ENV{"TPCD_PLATFORM"}) <= 0)
{
  die "TPCD_PLATFORM environment variable not set\n";
}
$platform=$ENV{"TPCD_PLATFORM"};
if ( $platform eq "nt" )
{
   $sep="&";
}
else
{
   $sep=";";
}
if ((length($ENV{"TPCD_BUILD_STAGE"}) <= 0) ||
($ENV{"TPCD_BUILD_STAGE"}) eq "NULL" )
{
   $ENV{"TPCD_BUILD_STAGE"} = "ALL";
}
if (length($ENV{"TPCD_PRODUCT"}) <= 0)
{
   die "Must set TPCD_PRODUCT env't var.\n";
}
if ( length($ENV{"TPCD_DBNAME"}) <= 0 )
{
  die "TPCD_DBNAME environment variable not set\n";
}
if (length($ENV{"TPCD_MODE"}) <= 0)
{
  die "TPCD_MODE environment variable not set -
uni/smp/mln \n";
}
if (length($ENV{"TPCD_SF"}) <= 0)
{
  die "TPCD_SF environment variable not set\n";
}
if (length($ENV{"TPCD_DBPATH"}) <= 1)
{
   # if no db pathname specified, build the db in the home
directory
   if ( $platform eq "aix" )
   {
      $ENV{"TPCD_DBPATH"} = $ENV{"HOME"};
   }
   elsif ( $platform eq "nt" )
   {
      $ENV{"TPCD_DBPATH"} =
$ENV{"HOMEDRIVE"};
   }
   else
   {
      die "platform $platform not supported yet\n";
   }
}
if (length($ENV{"TPCD_DDLPATH"}) <= 0)
{
   # if no db pathname specified, use default
  $ENV{"TPCD_DBPATH"} =
"/afs/tor/groups/dbp/perf/benchmark/tpcd/ddl/vanilla";
}
if (length($ENV{"TPCD_DDL"}) <= 0)
{
   $ENV{"TPCD_DDL"} = "dss.ddl";
}
if (length($ENV{"TPCD_TBSP_DDL"}) <= 0)
{
   $ENV{"TPCD_TBSP_DDL"} = "dss.tbsp.ddl";
}
if (length($ENV{"TPCD_INDEXDDL"}) <= 0)
{
   $ENV{"TPCD_INDEXDDL"} = "dss.index";
}
if (length($ENV{"TPCD_RUNSTATS"}) <= 0)
{
   $ENV{"TPCD_RUNSTATS"} = "dss.runstats";
}
if (length($ENV{"TPCD_RUNSTATSHORT"}) <= 0)
{
   $ENV{"TPCD_RUNSTATSHORT"} = "NULL";
}
if (length($ENV{"TPCD_ADD_RI"}) <= 0)
{
   $ENV{"TPCD_ADD_RI"} = "NULL";
}
if (length($ENV{"TPCD_AST"}) <= 0)
{
   $ENV{"TPCD_AST"} = "NULL";
}
if ( ($ENV{"TPCD_INPUT"}) eq "NULL" )
{
   if (length($ENV{"TPCD_DBGEN"}) <= 0)
   {
```

```
      die "Must set TPCD_DBGEN if pregenerated flatfiles
are not provided (TPCD_INPUT=NULL)\n";
   }
}
if ( $qual eq "QUAL" )
{
   if ( ($ENV{"TPCD_QUAL_INPUT"}) eq "NULL" )
   {
      if (length($ENV{"TPCD_DBGEN"}) <= 0)
      {
         die "Must set TPCD_DBGEN if pregenerated flatfiles
are not provided (TPCD_QUAL_INPUT=NULL)\n";
      }
   }
}

if (length($ENV{"TPCD_TEMP"}) <= 1)
{
   $ENV{"TPCD_TEMP"} = "/u/$instance/sqllib/tmp";
}
if (length($ENV{"TPCD_SORTBUF"}) <= 0)
{
   $ENV{"TPCD_SORTBUF"} = 4096;
}
if (length($ENV{"TPCD_LOAD_PARALLELISM"}) <= 0)
{
   $ENV{"TPCD_LOAD_PARALLELISM"} = 0;
}
if (length($ENV{"TPCD_LOADSTATS"}) <= 0)
{
   $ENV{"TPCD_LOADSTATS"} = "no";
}
if (length($ENV{"TPCD_COPY_DIR"}) <= 0)
{
   $ENV{"TPCD_COPY_DIR"} =
"${delim}dev${delim}null";
}
if (length($ENV{"TPCD_FASTPARSE"}) <= 0)
{
   $ENV{"TPCD_FASTPARSE"} = "no";
}
if (length($ENV{"TPCD_BACKUP_DIR"}) <= 0)
{
   $ENV{"TPCD_BACKUP_DIR"} =
"${delim}dev${delim}null";
}
if (length($ENV{"TPCD_LOG"}) <= 0)
{
   $ENV{"TPCD_LOG"} = "no";
}
if (length($ENV{"TPCD_LOGPRIMARY"}) <= 0)
{
   $ENV{"TPCD_LOGPRIMARY"} = "NULL";
}
if (length($ENV{"TPCD_LOGSECOND"}) <= 0)
{
   $ENV{"TPCD_LOGSECOND"} = "NULL";
}
if (length($ENV{"TPCD_LOGFILSIZ"}) <= 0)
{
   $ENV{"TPCD_LOGFILSIZ"} = "NULL";
}
if (length($ENV{"TPCD_LOG_DIR"}) <= 0)
{
   $ENV{"TPCD_LOG_DIR"} = "NULL";
}
if (length($ENV{"TPCD_CONFIGFILE"}) <= 0)
{
   $ENV{"TPCD_CONFIGFILE"} = "dss.dbconfig";
}
if (length($ENV{"TPCD_DBM_CONFIG"}) <= 0)
{
   $ENV{"TPCD_DBM_CONFIG"} = "NULL";
}
if (length($ENV{"TPCD_MACHINE"}) <= 0)
{
   $ENV{"TPCD_MACHINE"} = "medium";
}
if (length($ENV{"TPCD_SMPDEGREE"}) <= 0)
{
   $ENV{"TPCD_SMPDEGREE"} = 1;
}
if (length($ENV{"TPCD_AGENTPRI"}) <= 0)
{
   $ENV{"TPCD_AGENTPRI"} = NULL;
}
if (length($ENV{"TPCD_ACTIVATE"}) <= 0)
{
   $ENV{"TPCD_ACTIVATE"} = "no";
}
if (length($ENV{"TPCD_AUDIT"}) <= 0)
{
   die "Must set TPCD_AUDIT env't var.  Real audit timing
sequence run if yes\n";
}
if (length($ENV{"TPCD_AUDIT_DIR"}) <= 0)
{
   die "TPCD_AUDIT_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_LOAD_DB2SET_SCRIPT"}) <=
0)
{
   $ENV{"TPCD_LOAD_DB2SET_SCRIPT"}="NULL"
}
if (length($ENV{"TPCD_DB2SET_SCRIPT"}) <= 0)
{
   $ENV{"TPCD_DB2SET_SCRIPT"}="NULL"
}
if (length($ENV{"TPCD_BUFFERPOOL_DEF"}) <= 0)
{
   $ENV{"TPCD_BUFFERPOOL_DEF"}="NULL"
}

#set up local variables
$tpcdVersion=$ENV{"TPCD_VERSION"};
$buildStage=$ENV{"TPCD_BUILD_STAGE"};
$product=$ENV{"TPCD_PRODUCT"};
$dbname=$ENV{"TPCD_DBNAME"};
$mode=$ENV{"TPCD_MODE"};
$sf=$ENV{"TPCD_SF"};
$sfReal=$sf;          # need a "saved" one for qualification
stuff
$dbpath=$ENV{"TPCD_DBPATH"};
$ddlpath=$ENV{"TPCD_DDLPATH"};
$ddl=$ENV{"TPCD_DDL"};
$buffpooldef=$ENV{"TPCD_BUFFERPOOL_DEF"};
$tbspddl=$ENV{"TPCD_TBSP_DDL"};
$indexddl=$ENV{"TPCD_INDEXDDL"};
$extraindex=$ENV{"TPCD_EXTRAINDEX"};
$runstats=$ENV{"TPCD_RUNSTATS"};
$runstatShort=$ENV{"TPCD_RUNSTATSHORT"};
$dbgen=$ENV{"TPCD_DBGEN"};
$input=$ENV{"TPCD_INPUT"};
$earlyindex=$ENV{"TPCD_EARLYINDEX"};
$ldtemp=$ENV{"TPCD_TEMP"};
$sortbuf=$ENV{"TPCD_SORTBUF"};
```

```
$load_parallelism=$ENV{"TPCD_LOAD_PARALLELISM"
};
$loadstats=$ENV{"TPCD_LOADSTATS"};
$copydir=$ENV{"TPCD_COPY_DIR"};
$fparse=$ENV{"TPCD_FASTPARSE"};
$addRI=$ENV{"TPCD_ADD_RI"};
$astFile=$ENV{"TPCD_AST"};
if ( $fparse eq "yes" )
{
   $fastparse="FASTPARSE";
}
else
{
   $fastparse=" ";
}
$backupdir=$ENV{"TPCD_BACKUP_DIR"};
$logprimary=$ENV{"TPCD_LOGPRIMARY"};
$logsecond=$ENV{"TPCD_LOGSECOND"};
$logfilsiz=$ENV{"TPCD_LOGFILSIZ"};
$log=$ENV{"TPCD_LOG"};
$logDir=$ENV{"TPCD_LOG_DIR"};
$machine=$ENV{"TPCD_MACHINE"};
$configfile=$ENV{"TPCD_CONFIGFILE"};
$dbmconfig=$ENV{"TPCD_DBM_CONFIG"};
$loadconfigfile=$ENV{"TPCD_LOAD_CONFIGFILE"};
$loadDBMconfig=$ENV{"TPCD_LOAD_DBM_CONFIGFI
LE"};
$smpdegree=$ENV{"TPCD_SMPDEGREE"};
$agentpri=$ENV{"TPCD_AGENTPRI"};
$activate=$ENV{"TPCD_ACTIVATE"};
$RealAudit=$ENV{"TPCD_AUDIT"};
$loadscript=$ENV{"TPCD_LOAD_SCRIPT"};
$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$loadsetScript=$ENV{"TPCD_LOAD_DB2SET_SCRIPT"}
;
$setScript=$ENV{"TPCD_DB2SET_SCRIPT"};

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

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

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

$loadDBMconfig=$ENV{"TPCD_LOAD_DBM_QUALCO
NFIGFILE"};
   if (length($ENV{"TPCD_LOG_DIR"}) <= 0)
   {
      $ENV{"TPCD_LOG_DIR"} = "NULL";
   }
   $logDir=$ENV{"TPCD_LOG_QUAL_DIR"};
}

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

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

  if ( $machine eq "small" )
```

```
 {
    $buffpage = 5000;
    $sortheap = 3000;
    $sheapthres = 8000;
    $util_heap_sz = 5000;
    $ioservers = 6;
 }
 elsif ( $machine eq "medium" )
 {
    $buffpage = 10000;
    $sortheap =  8000;
    $sheapthres = 20000;
    $util_heap_sz = 10000;
    $ioservers = 10;
 }
 elsif ( $machine eq "big" )
 {
    $buffpage = 30000;
    $sortheap = 20000;
    $sheapthres = 50000;
    $util_heap_sz = 30000;
    $ioservers = 20;
 }
 elsif ( $machine eq "sunsmp" )
 {
    $buffpage = 60000;
    $sortheap = 20000;
    $sheapthres = 80000;
    $util_heap_sz = 30000;
    $ioservers = 80;
 }
 elsif ( $machine eq "eastwood" )
 {
    $buffpage = 80000;
    $sortheap = 50000;
    $sheapthres = 81000;
    $ioclnrs = 4;
    $chngpgs = 30;
    $ioservers = 21;
    $util_heap_sz = 50000;
 }
}
# echo parameter settings to acknowledge what is being built
if ( $buildStage ne "ALL" )
{
   print " ***** STARTING the build process at the
$buildStage Stage *****\n";
}
print "Building a TPC-D Version $tpcdVersion $sf GB
database on $dbpath with: \n";
print "   Mode = $mode \n";
print "   Tablespace ddl in $ddlpath${delim}$tbspddl \n";
if ( $buffpooldef ne "NULL" )
{
  print "   Bufferpool ddl in $ddlpath${delim}$buffpooldef
\n";
}
print "   Table ddl in $ddlpath${delim}$ddl \n";
print "   Index ddl in
$ddlpath${delim}$indexddl\n";
if ( $extraindex ne "no" )
{
 print " Indices to create after the load
$ddlpath${delim}$extraindex\n";
}
if ( $loadscript eq "NULL")
{
   if ( $input eq "NULL" )
   {
      print "   Data generated by DBGEN in $dbgen\n";
   }
   else
   {
      print "   Data loaded from flat files in $input\n";
   }
}
if ( $earlyindex eq "yes" )
{
 print "   Indexes created before loading\n";
}
else
{
 print "   Indexes created after loading\n";
}
if ( $addRI ne "NULL" )
{
   print "   RI being used from $ddlpath${delim}$addRI\n";
}
print "ASTFILE = $astFile \n";
if ( $astFile ne "NULL" )
{
   print "   AST being used from
$ddlpath${delim}$astFile\n";
}
if ( $loadstats eq "yes" )
{
 if ( $earlyindex eq "yes" )
 {
   print "   Statistics for tables and indexes gathered during
load\n";
 }
 else
 {
  if ( $runstatShort eq "NULL" )
  {
    print "   Statistics for tables and indexes gathered after
load using $ddlpath${delim}$runstats \n";
  }
  else
  {
     print "   Statistics for tables and indexes gathered after
load using $ddlpath${delim}$runstats and
$ddlpath${delim}$runstatShort\n";
  }
 }
}
else
{
 if ( $runstatShort eq "NULL" )
 {
   print "   Statistics for tables and indexes gathered after
load using $ddlpath${delim}$runstats \n";
 }
 else
 {
   print "   Statistics for tables and indexes gathered after
load using $ddlpath${delim}$runstats and
$ddlpath${delim}$runstatShort\n";
 }
}
if ( $loadconfigfile ne "NULL" )
{
 print "   Database Configuration parameters for LOAD
taken from $ddlpath${delim}$loadconfigfile\n";
}
if ( $loadDBMconfig ne "NULL" )
```

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

if ((($loadconfigfile eq "") || ($loadconfigfile eq "NULL"))
{
  print "   Machine size set to $machine so the following
configuration\n";
  print "   parameters are used for load, create index and
runstats: \n";
  print "      BUFFPAGE = $buffpage \n";
  print "      SORTHEAP = $sortheap \n";
  print "      SHEAPTHRES = $sheapthres\n";
  print "      NUM_IOSERVERS = $ioservers\n";
  print "      NUM_IOCLEANERS = $ioclnrs\n";
  print "      CHNGPGS_THRESH = $chngpgs\n";
  print "      UTIL_HEAP_SZ = $util_heap_sz\n";
  print "      Degree of parallelism (dft_degree and
max_querydegree) set to $smpdegree\n";
  print "      Parameters for load are: temp file      =
$ldtemp\n";
  print "                               sort buf       = $sortbuf\n";
  print "                               ld parallelism =
$load_parallelism\n";
  if ( $fparse eq "yes" )
  {
    print "                               FASTPARSE used on load\n";
  }
}
if ( $loadscript ne "NULL")
{
  print "   Load commands  in
$ddlpath${delim}$loadscript\n";
}
else
{
  if (( $mode eq "mln" ) || ( $mode eq "mpp"))
  {
    die "   Loading script must be specified for mln or mpp
invocations\n";
  }
}
print "   Degree of parallelism (dft_degree and
max_querydegree) set to $smpdegree\n";
if ( $agentpri ne "NULL" )
{
    print "   AGENTPRI set to $agentpri\n";
}
if ( $activate eq "yes" )
{
    print "   Database will be activated when build is
complete\n";
}

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

#### Alex
#### Set RAHSLEEPTIME high to ensure synchronous
commands.
#system("set RAHSLEEPTIME=999999");
#### This didn't work for some reason, so be sure to set
RAHSLEEPTIME
#### before calling buildtpcd.

# set db2options so all usages work
# @db980723wlc
if (( $mode eq "mln" ) || ( $mode eq "mpp"))
{
```

```
  ### Alex - spaces needed for NT
  #$rc=&dodb_noconn("db2set DB2OPTIONS=\\\"-t -v
+c\\\"",$all_ln);
  $rc=&dodb_noconn("db2set DB2OPTIONS=\\\"   -t -v
+c\\\"",$all_ln);
}
else
{
  ### Alex - spaces needed for NT
  #$rc=&dodb_noconn("db2set DB2OPTIONS=\\\"-t -v
+c\\\"",$once);
  $rc=&dodb_noconn("db2set DB2OPTIONS=\\\"   -t -v
+c\\\"",$once);
}

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

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

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

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

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

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

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

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

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

```
   $setLogString.="db2 update db cfg for $dbname using
logbufsz 128";
   &dodb_noconn("$setLogString",$all_ln);

# 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" ) || ( $mode eq "mpp"))
      {
#$rc=system("db2_all \'}]<<+000< db2 \"backup database
$dbname to $backupdir without prompting\" \' ");
         $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\' ");
                  $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");
   }

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

&dodb2file($dbname,"$ddlpath${delim}$loadconfigfile",$al
l_ln);
&dodb2file($dbname,"$ddlpath${delim}$loadDBMconfig",
$once);
   }
   &dodb_noconn("db2 terminate",$once);
   $rc=system("db2stop");
   if ( $rc != 0 )
   {
      die "failure during db2stop after resetting for load
config rc = $rc \n";
   }
   $rc=system("db2start");
   if ( $rc != 0 )
   {
      die "failure during db2start rc = $rc \n";
   }
}  # end of "CREATE DATABASE" phase
if (( $buildStage eq "ALL" ) || ( $buildStage eq "CRTTBSP"
) ||
   ( $buildStage eq "LOAD" ) ||
   (( $buildStage eq "INDEX" ) && ( $earlyindex eq "yes" ))
)
{
      #### Alex - I put the create bufferpool before the
EXPLAIN.DDL
      #### because the latter has a dependency on a
CREATE NODEGROUP
      #### statement in the former.
   # do the bufferpool stuff
   if ( $buffpooldef ne "NULL" )
   {
      #run the create bufferpool ddl
      &outtime("*** Creating the bufferpools");

&dodb2file($dbname,"$ddlpath${delim}$buffpooldef",$onc
e);

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

```
   &outtime("*** Ready to start creating the tablespaces");

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

   #### Alex - This is old stuff, left here for posterity.
   #### The loading of dummy UF data has been moved
   #### to the top of the timed interval.
   # if we are in audit mode, then we must create the the
tablespaces and
   # tables for the update functions and we must generate the
data for the
   # update functions before we start timing the load.  (All
activity
   # on the database after the table creation is started and
before the performance
   # tests are run must be included in load time
   # NOTE: we do not have to do this if we are building the
qualification database
   if (( $RealAudit eq "yes" ) && ( $qual ne "QUAL" ))
   {
      # build the update function staging tables
      if ( $product eq "pe" )
      {
         print "update files for pe v1.2 style are build through
bhatta's scripts\n";
      }
   }

   &outtime("*** Start Load Timing now - starting to create
tables");
   # Create the Update Staging tables and preload them with
   # dummy data for the purposes of updating statistics.  The
   # dummy data that is loaded will not be used in any
potential
   # update functions.  This data will be deleted immediately
after
   # the load is completed.  The dummy data that is loaded is
from
   # a pair that will not be used in the run, i.e. the pair
number
   # is greated than (( #streams + 1 ) * 3 ) (where 3 is the
number
   # of runs that could potentially be run on the database).

&dodb2file($dbname,"$ddlpath${delim}create_UF_staging_
tables");
   system("db2start");
   #### Alex changed the path and name.
   $rc =
system("$auditDir${delim}tools${delim}loaddummyUFdata
.bat");
   if ( $rc != 0 )
   {
      die "Preload of dummy UF data failed rc=$rc\n";
   }
   # delete the dummy rows now that the sample statistics
have been captured.
   #### Alex changed the path and name.

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

   # Create the main tables.
   &dodb2file($dbname,"$ddlpath${delim}$ddl",$once);

   # update the locksize on the non-updated tables to be table
level locking
   # update the tables for appendmode
   &dodb_conn($dbname,
           "db2 alter table tpcd.nation locksize table $sep \
            db2 alter table tpcd.region locksize table $sep \
            db2 alter table tpcd.customer locksize table $sep \
            db2 alter table tpcd.supplier locksize table $sep \
            db2 alter table tpcd.part locksize table $sep \
            db2 alter table tpcd.partsupp locksize table $sep \
            db2 alter table tpcd.lineitem append on $sep \
            db2 alter table tpcd.orders append on",
           $once);

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

&dodb2file($dbname,"$ddlpath${delim}$loadconfigfile",$al
l_ln);

&dodb2file($dbname,"$ddlpath${delim}$loadDBMconfig",
$once);
     }
     &dodb_noconn("db2 terminate",$once);
     $rc=system("db2stop");
     if ( $rc != 0 )
     {
      die "failure during db2stop after resetting for load
config rc = $rc \n";
     }
     $rc=system("db2start");
     if ( $rc != 0 )
     {
      die "failure during db2start rc = $rc \n";
```

```
    }
  }

  # if earlyindex requested, create indexes
  if ( $earlyindex eq "yes" )
  {
    &outtime("*** Starting to create indexes");

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

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

  # start the dbgen and load.....call the specific mode for
loading (uni,smp,mln)
  if (( $mode eq "uni" ) || ( $mode eq "smp" ))
  {
    &outtime("*** Starting the load");
    # call the appropriate dbgen/load for uni/smp
    if (( $loadscript ne "NULL") && ( $loadscript ne "" ))
    {

&dodb2file($dbname,"$ddlpath${delim}$loadscript",$once)
;
    }
    else
    {
      $rc = system("perl genloaduni $qual");
      if ( $rc != 0 )
      {
        die "genloaduni failed rc = $rc\n";
      }
    }
  }
  elsif (( $mode eq "mln" ) || ( $mode eq "mpp" ))
  {
    &outtime("*** Starting the load");
    # call the appropriate dbgen/split/(sort)/load for
mln/mpp
    $rc = system("$ddlpath${delim}$loadscript $sf");
    if ( $rc != 0 )
    {
      die "doload for $dbname failed rc = $rc\n";
    }
  }
  else
  {
    die "TPCD_MODE not set to one of uni, smp, mln or
mpp\n";
  }
}  # end all and load stage (and create index if early index
was specified

if (( $buildStage eq "ALL" ) || ( $buildStage eq "CRTTBSP"
) ||
  ( $buildStage eq "LOAD" ) ||
  ( $buildStage eq "INDEX" ))
{
  if ( $buildStage eq "INDEX")
  { # we're restarting a build so reapply the load config
   &outtime("*** Setting LOAD configuration.");
   if (($loadconfigfile eq "") || ($loadconfigfile eq "NULL"))
   {
    &dodb_noconn("db2 update db cfg for $dbname using
buffpage $buffpage $sep \
          db2 update db cfg for $dbname using sortheap
$sortheap $sep \
          db2 update db cfg for $dbname using
num_iocleaners $ioclnrs $sep \
          db2 update db cfg for $dbname using num_ioservers
$ioservers $sep \
          db2 update db cfg for $dbname using util_heap_sz
$util_heap_sz $sep \
          db2 update db cfg for $dbname using
chngpgs_thresh $chngpgs",
          $all_ln);
    &dodb_noconn("db2 update dbm cfg using sheapthres
$sheapthres",$once);
    }
    else
    {

&dodb2file($dbname,"$ddlpath${delim}$loadconfigfile",$al
l_ln);

&dodb2file($dbname,"$ddlpath${delim}$loadDBMconfig",
$once);
    }
    &dodb_noconn("db2 terminate",$once);
    $rc=system("db2stop");
    if ( $rc != 0 )
    {
     die "failure during db2stop after resetting for load
config rc = $rc \n";
    }
    $rc=system("db2start");
    if ( $rc != 0 )
    {
     die "failure during db2start rc = $rc \n";
    }
  }
  # if indexes haven't been created, do so now
  if ( $earlyindex ne "yes" )
  {
    &outtime("*** Create index started");

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

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

&dodb2file($dbname,"$ddlpath${delim}$extraindex",$once)
;
    &outtime("*** Create index (part 2) completed");
  }

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

if (( $buildStage eq "ALL" ) ||  ( $buildStage eq "CRTTBSP"
) ||
  ( $buildStage eq "LOAD" ) ||
  ( $buildStage eq "INDEX" ) || ( $buildStage eq
"RUNSTATS" ))
{
  if ( $buildStage eq "RUNSTATS")
  { # we're restarting a build so reapply the load config
   &outtime("*** Setting LOAD configuration.");
   if (($loadconfigfile eq "") || ($loadconfigfile eq "NULL"))
   {
    &dodb_noconn("db2 update db cfg for $dbname using
buffpage $buffpage $sep \
          db2 update db cfg for $dbname using sortheap
```

```
$sortheap $sep \
          db2 update db cfg for $dbname using
num_iocleaners $ioclnrs $sep \
          db2 update db cfg for $dbname using num_ioservers
$ioservers $sep \
          db2 update db cfg for $dbname using util_heap_sz
$util_heap_sz $sep \
          db2 update db cfg for $dbname using
chngpgs_thresh $chngpgs",
          $all_ln);
     &dodb_noconn("db2 update dbm cfg using sheapthres
$sheapthres",$once);
    }
    else
    {

&dodb2file($dbname,"$ddlpath${delim}$loadconfigfile",$al
l_ln);

&dodb2file($dbname,"$ddlpath${delim}$loadDBMconfig",
$once);
    }
    &dodb_noconn("db2 terminate",$once);
    $rc=system("db2stop");
    if ( $rc != 0 )
    {
     die "failure during db2stop after resetting for load
config rc = $rc \n";
    }
    $rc=system("db2start");
    if ( $rc != 0 )
    {
     die "failure during db2start rc = $rc \n";
    }
   }
   # if statistics not gathered on the load, run runstats  (we
have to run the
   # stats at the same time as the index creation whether it be
both during load,
   # or after load)
   # We need to run the runstats as well if we have specifed
an extra index file
   # for "after load" indexes
   if (( $loadstats eq "no" ) || ( $earlyindex eq "no" ) || (
$extraindex ne "no" ))
   {
      # if loadstats not gathered, then index stats not gathered
either.
      &outtime("*** Runstats started");

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

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

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


# Whichever build phase was requested, it is appropriate to
apply the runtime configuration now.

# set the runtime configuration
&outtime("*** Set Configuration started");
if ( $smpdegree ne "NULL")
{
      &outtime("*** Setting degree of parallelism");
      &dodb_noconn("db2 update database configuration for
$dbname using dft_degree $smpdegree",$all_ln);
      &dodb_noconn("db2 update database manager
configuration using max_querydegree $smpdegree",$once);
}
&dodb2file(${dbname},"${ddlpath}${delim}$configfile",$a
ll_ln);
&dodb2file(${dbname},"${ddlpath}${delim}$dbmconfig",$
once);

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

# set the db2 environment variables for running the
benchmark
if ( $setScript ne "NULL" )
{
$ret=system("${ddlpath}${delim}$setScript");
   #### $ret=system("rah \"cd ${ddlpath} & $setScript \" ");
   if ( $ret != 0 )
   {
      die "failure setting runtime db2set parms from
$setScript \n";
   }
}

# stop and restart the database to get config parms
recognized
$rc=system("db2stop");
if ( $rc != 0 )
```

```
{
   die "failure during db2stop rc = $rc \n";
}
$rc=system("db2start");
if ( $rc != 0 )
{
   die "failure during db2start rc = $rc \n";
}


if (( $buildStage eq "ALL" ) ||  ( $buildStage eq "CRTTBSP"
) ||
   ( $buildStage eq "LOAD" ) || ( $buildStage eq "INDEX" )
||
   ( $buildStage eq "RUNSTATS" ) || ( $buildStage eq "RI"
))
{
   if ( $addRI ne "NULL" )
   {
      &outtime("*** Adding RI contraints started");

&dodb2file($dbname,"$ddlpath${delim}$addRI",$once);
      &outtime("*** Adding RI contraints completed");
   }
}

if (( $buildStage eq "ALL" ) ||  ( $buildStage eq "CRTTBSP"
) ||
   ( $buildStage eq "LOAD" ) || ( $buildStage eq "INDEX" )
||
   ( $buildStage eq "RUNSTATS" ) || ( $buildStage eq "RI" )
||
   ( $buildStage eq "AST" ))
{
   #add the AST if it has been requested
   if ( $astFile ne "NULL" )
   {
      &outtime("*** Adding AST started");

&dodb2file($dbname,"$ddlpath${delim}$astFile",$once);
      &outtime("*** Adding AST completed");
   }
}

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

if ( $RealAudit eq "yes" )
{
   # if we are in real audit mode then we have to do a number
of things
   # set up the audit directory structure and the run directory
structure
   # so that once we have completed the buildtpcd, we are
ready to run.
   # first remove any old "update pair number" file so we
won't get prompted
   # doing setupDir

&rm("$auditDir${delim}$dbname.$user.update.pair.num");
   &rm("$auditDir${delim}tools${delim}tpcd.runsetup");
   system("perl setupDir");
   system("perl setupRun");
   # before we stop the database for the final time
   # if we are in the real audit mode then compile and bind
tpcdbatch, and
   # run dbtables and dbcheck before we print out the final
notice that
   # we are ready to run the performance tests
   # if we are building the qualification database then we will
bind to both
   # the dbname database and the qualification database
   $rc = system("perl buildtpcdbatch $qual");
   if ( $rc != 0 )
   {
      die "buildtpcdbatch failed rc=$rc\n";
   }
   if ( $qual eq "QUAL" )
   {
      $verifyType="q";
   }
   else
   {
      $verifyType="t";
   }
   system("perl tablesdb $verifyType");

&dodb2file($dbname,"$auditDir${delim}tools${delim}first
10rows.sql",$once);
}
# stop, restart and activate the database, if necessary
$rc=system("db2stop");
if ( $rc != 0 )
{
   die "failure during db2stop rc = $rc \n";
}

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

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

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

1;
```

B 2: 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=nt                    # aix, nt, ....
TPCD_VERSION=1                      # 1 or 2 (Version of
tpcd). Default 1
TPCD_DBNAME=TPCD                    # name to create
database under
TPCD_AUDIT_DIR=c:\tpcd_jan23\tpcd_audit_dir       # top
```

```
level directory of tar file for
                        # all the tpcd scripts
TPCD_PRODUCT=v5                 # v5 or pe
                        # Use pe if you really are using pe
v1.2!
                        # but I won't guarantee that it will
work!
TPCD_MODE=smp                   # uni/smp/mln/mpp
TPCD_PHYS_NODE=1                # number of physical
nodes
TPCD_LN_PER_PN=1                # number of logical
nodes per physical node
TPCD_SF=30                      # size of the database
(1=1GB,...) to
                        # get test size databases use:
                        #    0.012 = 12MB
                        #    0.1   = 100MB
TPCD_BUILD_STAGE=ALL            # where to start the
build - currently the
                        # following is possible:
                        #  ALL - do everything (create,load,
                        #       index,stats,config) (Default)
                        #  CRTTBSP - start after create db
and
                        #     config setting.  Start righ at
                        #     create tbsp
                        #  LOAD - start from the load of the
tables
                        #  INDEX - start from the index
creation
                        #      (NOTE if earlyindex is
specified,
                        #       then this will do the create
index
                        #       followed by the load...)
                        #  RUNSTATS - start from the
runstats
                        #      (NOTE Do not use this option
if
                        #       distribution stats are gathered
                        #       as part of the load, this will
                        #       start after the load and indices
                        #       have been created.
                        #  CONFIG - start from the setting
up of
                        #       the benchmark runs config
setup
                        #
TPCD_BUILD_DATABASE=NULL        # whether to
build a new database ???
                        # NOT CURRENTLY USED
TPCD_DBPATH=c:                  # path for database
(defaults to home)
TPCD_DDLPATH=c:\tpcd_jan23\tpcd_audit_dir\ddl\hitachi
_30 # path for all ddl files and customized
                        # scripts (load script), config
files,etc
TPCD_BUFFERPOOL_DEF=dss.ddl.bufferpools # name of
file with bufferpool definitions and
                        # sizes
TPCD_TBSP_DDL=dss.ddl.tbsp.hitachi.30GB     # ddl file
for tablespaces
TPCD_DDL=dss.ddl.tables.hitachi.30GB        # ddl file for
tables
TPCD_QUAL_TBSP_DDL=dss.ddl.tbsp.qual  # ddl file for
tablespaces for qual
TPCD_QUAL_DDL=dss.ddl.tables.qual     # ddl file for
qualification database
                        # tablespaces and tables should be
identical
                        # to regular ddl except container
names
TPCD_INDEXDDL=dss.ddl.indexes.hitachi  # ddl file for
indexes
TPCD_EXTRAINDEX=no              # no = no extra
indexes
                        # filename = If you want to create
some
                        # indices before
                        # the load, and some indices after,
then
                        # use this additional file to specify
the
TPCD_ADD_RI=dss.ddl.ri.hitachi      # file name that
contains any RI
                        # constraints to add after index
creation
                        # set to NULL (default) if unused
                        # indices to create after the load.
TPCD_AST=dss.multiple.AST           # file name that
contains complete AST
                        # definition including connection to
                        # the database, summary table
creation,
                        # population, indexing and runstats.
TPCD_RUNSTATS=dss.ddl.runstats.hitachi # ddl file for
runstats.  If you have
                        # created indices before the load (ie
                        # TPCD_EARLYINDEX=yes), have
specified to
                        # gather stats on the load command
(either
                        # through your own load script or by
using
                        # TPCD_LOADSTATS=yes, AND
you have
                        # specified a file for
TPCD_EXTRAINDEX
                        # then this runstats file should
include
                        # the runstats commands
specifically for
                        # the extra indices.
TPCD_RUNSTATSHORT=NULL          # NOTE!!
THIS IS BUGGY....I can't get it to
                        # work on UNI successfully
                        # ddl file for short runstats that are
                        # run in the background while the
                        # TPCD_RUNSTATS are run in the
foreground
                        # of the build.  If this is used, then
                        # TPCD_RUNSTATS should have
the runstats
                        # command for lineitem and
                        # TPCD_RUNSTATSHORT should
have runstats
                        # commands for all other tables.
TPCD_DBGEN=c:\tpcd_jan23\tpcd_audit_dir\appendix\dbg
en # path name to data generation code

                        # Parameters used to specify source
of
                        # data for load scripts
TPCD_INPUT=z:\30GB_flatdata     # NULL - use dbgen
generated data  OR
                        # path name -  to the pre-generated
```

```
                        #           flat files
                        # /gwl/dss/12MB - path for
pregenerated 12MB
                        # /gwl/dss/100MB - path for
pregen'd 100MB
                        #
TPCD_QUAL_INPUT=z:\qual_data      # NULL - use dbgen
generated data  OR
                        # path name -  to the pre-generated
                        #           flat files

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

TPCD_EARLYINDEX=no                # create indexes
before the load

                        # LOAD specific parameters follow:
TPCD_LOAD_DB2SET_SCRIPT=db2set.build.bat # Script
that contains the db2set commands
                        # for the load process Use NULL if
not
                        # specified
TPCD_LOAD_CONFIGFILE=dss.dbconfig.load.hitachi.30G
B      # config file with specific database config
                        # parms for the load/index/runstats
part
                        # of the build.
                        # set to NULL if use defaults
TPCD_LOAD_DBM_CONFIGFILE=dss.dbmconfig.load.hit
achi.30GB # config file with specific
                        # database manager config parts for
the
                        # load/index/runstats part of the
build.
                        # set to NULL if use defaults
TPCD_LOAD_QUALCONFIGFILE=dss.dbconfig.load.qual
# config file with specific database config
                        # parms for the load/index/runstats
part
                        # of the build for qualification db.
                        # set to NULL if use defaults
TPCD_LOAD_DBM_QUALCONFIGFILE=dss.dbmconfig.l
oad.qual # config file with specific
                        # database manager config parts for
the
                        # load/index/runstats part of the
build.
                        # set to NULL if use defaults
TPCD_LOADSTATS=no                 # gather statistics
during load
                        # ignored if EARLYINDEX is not
set
                        # due to runstats limitation
TPCD_TEMP=                        # path for LOAD temp
files
                        # defaults to
/u/<instance>/sqllib/tmp
                        # used in load script only
TPCD_SORTBUF=                     # sortbuf size for LOAD
                        # used in load script only
TPCD_LOAD_PARALLELISM=0           # degree of
parallelism to use on load
                        # 0 = use the "intelligent default"
that
                        # the load will chose at run time
                        # used in load script only
TPCD_COPY_DIR=                    # directory where copy
image is created
                        # on load command CURRENTLY
UNUSED
                        # used in load script only
TPCD_FASTPARSE=                   # use fastparse on load
                        # used in load script only

                    # Backup and logfile specific
parameters follow:
TPCD_BACKUP_DIR=g:\backup         # directory
where backup files are placed
TPCD_LOGPRIMARY=100               # NULL/value =
how many primary log files
                        # to configure.  If NULL is specified
then
                        # the default is not changed.
TPCD_LOGFILSIZ=4094               # NULL/value = how
4KB pages to use for
                        # logfilsiz db cfg parameter.  If
NULL is
                        # specified then the default is not
changed
TPCD_LOGSECOND=25                 # NULL/value =
how many secondary log files
                        # to configure.  If NULL is specified
then
                        # the default is not changed.
TPCD_LOG_DIR=g:\logs              # directory where
log files stored..
                        # NULL leaves them in the dbpath
TPCD_LOG_QUAL_DIR=z:\logs         # directory
where qual log files stored
                        # NULL leaves them in the dbpath
TPCD_LOG=no                       # yes/no - whether to turn
LOG_RETAIN on
                        # i.e. are backups needed to be
taken



                        # CONFIG specific parameters
TPCD_DB2SET_SCRIPT=db2set.run.bat      # Script that
contains the db2set commands
                        # for the benchmark run.  Use
NULL if not
                        # specified
TPCD_CONFIGFILE=dss.dbconfig.hitachi.30GB    # name
of configuration file in ddl path
                        # that will be used for the
benchmark run
TPCD_DBM_CONFIG=dss.dbmconfig.hitachi.30GB   #
name of config file for database manager
                        # cfg parms
TPCD_QUALCONFIGFILE=dss.dbconfig.qual # name of
database cfg file in ddl path
                        # for qualification database
TPCD_DBM_QUALCONFIG=dss.dbmconfig.qual # name
of config file for database
                        # manager cfg parms

TPCD_MACHINE=NULL                 # set to NULL if
using load config file
                        # big/medium/small  size of
machine used to
                        # determine buffpage,
sortheap,sheapthres
```

```
                        # and ioservers parms for load,
create
                        # index and runstats
                        # NOTE that this parameter is
ignored if
                        # a TPCD_LOAD_CONFIGFILE

TPCD_SMPDEGREE=32               # 1...# of degrees of
parallelism to run
                        # with
TPCD_AGENTPRI=NULL              # set agentpri to this
value (default
                        # is SYSTEM)

TPCD_ACTIVATE=no                # activate the database
upon build
                        # completion



                        # run specific parameters
TPCD_AUDIT=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=g:\tmp             # place to put temp
working files

TPCD_QUERY_TEMPLATE_DIR=standard      #
subdirectory in AUDIT_DIR/queries
                        # to use as the source of the query
                        # templates.  Currently there are
                        # v2 ones and pe ones.  You can
make
                        # your own directory following the
same
                        # form as in the v2 directory using
                        # any variant you wish
TPCD_QUAL_DBNAME=TPCD           # name of
qualification database
TPCD_NUMSTREAM=5                # number of streams
for the throughput test
TPCD_FLATFILES=x:\              #
#TPCD_FLATFILES=Z:\UF_Flatfiles       # where to
generate flat files
                        # for update functions
TPCD_UPDATE_IMPORT=false        # true = use
import for the staging tables
                        #   for UNI/SMP mode only (code
change in
                        #   tpcdbatch) (if not uni mode then
must
                        #   change load_update)
                        # false = use load for staging tables
                        # The default is false if not set.
                        # NOTE that this parm is only for
UNI/SMP
                        # it is not for multi node invocation

TPCD_SPLIT_UPDATES=2000         # number of
chunks to split the update
                        # function into.
TPCD_CONCURRENT_INSERTS=40      # number of
insert chunks that are run
#TPCD_CONCURRENT_INSERTS=320     # number
of insert chunks that are run
                        # concurrently.  This number should
be
                        # evenly divisible by
TPCD_SPLIT_UPDATES
TPCD_CONCURRENT_INSERTS_LOAD=40       #
number of insert chunks that are loaded
                        # concurrently.  This number should
be
                        # evenly divisible by
TPCD_SPLIT_UPDATES
                        # this controls the load portion of
the
                        # insert routine for partitioned
databases
#TPCD_CONCURRENT_DELETES=400     # number
of concurrent portions to run
TPCD_CONCURRENT_DELETES=50      # number of
concurrent portions to run
TPCD_SPLIT_DELETES=6000         # number of
portions to split the delete
                        # function into.
                        # this variable is only valid in
UNI/SMP
                        # mode.
TPCD_GEN_UPDATEPAIRS=40         # number of
pairs of update function data
                        # to generate
                        # if 0 the update data generation and
                        # setup will not be done.  use this if
                        # you don't want to run the update
                        # functions   (Update functions not
                        # fully tested in new env't yet)
TPCD_GENERATE_SEED_FILE=yes     # yes/no
These are the seed files for
                        # generating the query substitution
values
                        # yes - generate a seed file base on
                        #       year/month/day (for audited
runs)
                        # no  - use qgen's default seeds
TPCD_RUN_ON_MULTIPLE_NODES=NULL       # pe
V1.2 only - will we be running each
                        # query stream of throughput
starting at
                        # different nodes or from same node
TPCD_STATS_INTERVAL=30          # timing interval
for vmstats/iostats
TPCD_STATS_THRU_INT=300         # timing interval
for vmstats/iostats for
                        # throughput run
TPCD_GATHER_STATS=off           # on/off   - only
implement for AIX yet
                        # on = gather statistics around
power
                        #       test run  (vmstat,iostat,netstat)
                        # off = no stats gathered during
power run

TPCD_UFTEMP=UF_TEMP             # base name of
tablespace(s) where the
                        # staging tables for the update
functions
                        # are created
                        # this name will be used as the
                        # basename for the tablespaces...eg
```

```
                        # UFTEMP1 UFTEMP2 ....
TPCD_HAVECOMPILER=yes           # rebuild
tpcdbatch executable
                        # yes/no
TPCD_SLEEP=5                    # ?
TPCD_INLISTMAX=default          # max num of keys
to delete at a time
                        # for UF2, use "default" for default.
TPCD_LOAD_SCRIPT=dss.load30GB.hitachi # script to
run for loading tables
                        # in TPCD_DDLPATH directory
under mln/mpp
                        # leave as NULL if using default
genloaduni
TPCD_LOAD_SCRIPT_QUAL=dss.load100MB.qual #
script to run for loading tables in
                        # TPCD_DDLPATH directory
under mln/mpp
                        # for QUAL db
    #  acid test specific information
TPCD_DB2LOG=c:\sqllib\db2       # directory wehre the
db2diag.log can
                        # be found for the durability tests
```

B 3: dss.dbconfig.load.hitachi.30GB

```
update database configuration for tpcd using
buffpage 20000
catalogcache_sz 386
chngpgs_thresh 40
dbheap 6654
locklist 642
logbufsz 64
logfilsiz 4094
logprimary 100
logsecond 25
newlogpath g:\logs
maxappls 40
maxlocks 75
mincommit 1
num_iocleaners 20
num_ioservers 20
pckcachesz 320
softmax 100
sortheap 50000
stat_heap_sz 4384
stmtheap 4096
util_heap_sz 40000
applheapsz 768
app_ctl_heap_sz 256
dft_degree 20
chngpgs_thresh 10;

get database configuration for tpcd;

--connect reset;
```

B 4: dss.dbmconfig.load.hitachi

```
update database manager configuration using
cpuspeed 2.6e-006
sheapthres 100000
agent_stack_sz 16
aslheapsz 15
rqrioblk 32767
intra_parallel yes
max_querydegree -1
maxagents 1000
num_poolagents 4
num_initagents 4
diaglevel 3;

get database manager configuration;

--connect reset;
```

B 5: db2set.build.bat

```
db2set DB2_NEW_TF_RANGE_FF=
db2set DB2_NEW_CORR_SQ_FF=
db2set DB2_ORDERED_NLJN=
db2set DB2BPNUMRANGEPREF=
db2set DB2BPNUMRANGEPQS=
db2set DB2BPNUMHATESTACKS=
db2set DB2BPNUMPREFQUEUES=
db2set DB2BPPREFQUEUESIZE=
db2set DB2_SORT_BUFF=
db2set DB2_SORT_NEWORDER=
db2set DB2_MGJN_WITH_SORT=
db2set "DB2_LIKE_VARCHAR        "=
db2set DB2_CORRELATED_PREDICATES=
db2set DB2_VECTOR=
db2set DB2_RR_TO_RS=
db2set DB2OPTIONS="-t -v +c"
db2set DB2NTNOCACHE=ON

db2set
```

B 6: dss.ddl.tbsp.hitachi.30GB

```
create regular tablespace LINEITEM_TABLE
  pagesize 8k
  managed by database
  using (device '\\.\PhysicalDrive17'  2750000,
        device '\\.\PhysicalDrive20'  2750000)
  bufferpool   BP8K
  extentsize     16
  prefetchsize  320
  overhead       8
  transferrate  1.5;

create regular tablespace LINEITEM_INDEX
  pagesize 8k
  managed by database
  using (device '\\.\PhysicalDrive16'  2750000,
        device '\\.\PhysicalDrive24'  2750000,
                device '\\.\PhysicalDrive19'  2750000,
                device '\\.\PhysicalDrive27' 2750000,
                device '\\.\PhysicalDrive22'  2750000)
  bufferpool   BP8K
  extentsize     16
  prefetchsize  320
  overhead       8
  transferrate  1.5;

create regular tablespace OTHER
  pagesize 8k
  managed by database
  using (device '\\.\PhysicalDrive18'  2750000,
```

```
      device '\\.\PhysicalDrive26' 2750000,
             device '\\.\PhysicalDrive21'  2750000,
             device '\\.\PhysicalDrive29' 2750000,
             device '\\.\PhysicalDrive23'  2750000)
  bufferpool   BP8K
  extentsize     16
  prefetchsize  320
  overhead        8
  transferrate  1.5;

create temporary tablespace TEMP_TABLES
  pagesize 8k
  managed by database
  using (device '\\.\PhysicalDrive25'   2750000,
       device '\\.\PhysicalDrive28'  2750000)
  bufferpool   BP8K
  extentsize     16
  prefetchsize  320
  overhead        8
  transferrate  1.5;

--drop tablespace USERSPACE1;

-- drop tablespace TEMPSPACE1;

commit work;
```

B 7: dss.ddl.tables.hitachi.30GB

```
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 OTHER;

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

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 OTHER;

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 OTHER;

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 OTHER;

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 OTHER;

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 OTHER;

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 LINEITEM_TABLE
           INDEX IN LINEITEM_INDEX;

COMMIT WORK;
```

B 8: dss.load30GB.hitachi

```
LOAD FROM w:\region.tbl OF DEL
  MODIFIED BY COLDEL| FASTPARSE
  MESSAGES \tmp\TPCD\region.msg
  REPLACE INTO TPCD.REGION
  STATISTICS NO
  NONRECOVERABLE;

LOAD FROM w:\nation.tbl OF DEL
  MODIFIED BY COLDEL| FASTPARSE
  MESSAGES \tmp\TPCD\nation.msg
  REPLACE INTO TPCD.NATION
  STATISTICS NO
  NONRECOVERABLE;

LOAD FROM
  w:\partsupp.tbl.1,
  w:\partsupp.tbl.2 ,
  w:\partsupp.tbl.3 ,
  w:\partsupp.tbl.4 ,
  w:\partsupp.tbl.5 ,
  w:\partsupp.tbl.6 ,
  w:\partsupp.tbl.7 ,
  w:\partsupp.tbl.8 ,
  w:\partsupp.tbl.9 ,
  w:\partsupp.tbl.10
  OF DEL
  MODIFIED BY COLDEL| FASTPARSE
  MESSAGES \tmp\TPCD\partsupp.msg
  REPLACE INTO TPCD.PARTSUPP
  STATISTICS NO
  NONRECOVERABLE;

LOAD FROM
  w:\supplier.tbl.1,
  w:\supplier.tbl.2 ,
  w:\supplier.tbl.3 ,
  w:\supplier.tbl.4 ,
  w:\supplier.tbl.5 ,
  w:\supplier.tbl.6 ,
  w:\supplier.tbl.7 ,
  w:\supplier.tbl.8 ,
  w:\supplier.tbl.9 ,
  w:\supplier.tbl.10
  OF DEL
  MODIFIED BY COLDEL| FASTPARSE
  MESSAGES \tmp\TPCD\supplier.msg
  REPLACE INTO TPCD.SUPPLIER
  STATISTICS NO
  NONRECOVERABLE;

LOAD FROM
  w:\part.tbl.1,
  w:\part.tbl.2 ,
  w:\part.tbl.3 ,
  w:\part.tbl.4 ,
  w:\part.tbl.5 ,
  w:\part.tbl.6 ,
  w:\part.tbl.7 ,
  w:\part.tbl.8 ,
  w:\part.tbl.9 ,
  w:\part.tbl.10
  OF DEL
  MODIFIED BY COLDEL| FASTPARSE
  MESSAGES \tmp\TPCD\part.msg
  REPLACE INTO TPCD.PART
  STATISTICS NO
  NONRECOVERABLE;

LOAD FROM
  w:\customer.tbl.1,
  w:\customer.tbl.2 ,
  w:\customer.tbl.3 ,
  w:\customer.tbl.4 ,
  w:\customer.tbl.5 ,
  w:\customer.tbl.6 ,
  w:\customer.tbl.7 ,
  w:\customer.tbl.8 ,
  w:\customer.tbl.9 ,
  w:\customer.tbl.10
  OF DEL
  MODIFIED BY COLDEL| FASTPARSE
  MESSAGES \tmp\TPCD\customer.msg
  REPLACE INTO TPCD.CUSTOMER
  STATISTICS NO
  NONRECOVERABLE;

LOAD FROM
  w:\order.tbl.1,
  w:\order.tbl.2 ,
  w:\order.tbl.3 ,
  w:\order.tbl.4 ,
  w:\order.tbl.5 ,
  w:\order.tbl.6 ,
  w:\order.tbl.7 ,
  w:\order.tbl.8 ,
  w:\order.tbl.9 ,
  w:\order.tbl.10
  OF DEL
  MODIFIED BY COLDEL| FASTPARSE
  MESSAGES \tmp\TPCD\order.msg
  REPLACE INTO TPCD.ORDERS
  STATISTICS NO
  NONRECOVERABLE;

LOAD FROM
  w:\lineitem.tbl.1,
  w:\lineitem.tbl.2 ,
  w:\lineitem.tbl.3 ,
  w:\lineitem.tbl.4 ,
  w:\lineitem.tbl.5 ,
  w:\lineitem.tbl.6 ,
  w:\lineitem.tbl.7 ,
  w:\lineitem.tbl.8 ,
  w:\lineitem.tbl.9 ,
  w:\lineitem.tbl.10
  OF DEL
  MODIFIED BY COLDEL| FASTPARSE
  MESSAGES \tmp\TPCD\lineitem.msg
  REPLACE INTO TPCD.LINEITEM
  STATISTICS NO
  NONRECOVERABLE;
```

B 9: dss.ddl.indexes.hitachi

```
CREATE UNIQUE INDEX "TPCD  "."C_MS_CK" ON
"TPCD    "."CUSTOMER"
          ("C_MKTSEGMENT" ASC,
```

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

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

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

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

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

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


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

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

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

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

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

values(current timestamp);
CREATE INDEX "TPCD    "."O_CLERK" ON "TPCD
"."ORDERS"
           ("O_CLERK" ASC)
           PCTFREE 4 ;
commit work;

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

values(current timestamp);
CREATE UNIQUE INDEX "TPCD  "."P_TP_PK" ON
"TPCD    "."PART"
           ("P_TYPE" ASC,
```

```
            "P_PARTKEY" ASC)
            PCTFREE 0 ;
commit work;

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

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

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

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

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

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

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

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

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

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

values(current timestamp);
```

B 10: dss.ddl.runstats.hitachi

```
RUNSTATS ON TABLE TPCD.NATION    AND INDEXES
ALL;
commit work;
RUNSTATS ON TABLE TPCD.REGION    AND INDEXES
ALL;
commit work;
RUNSTATS ON TABLE TPCD.SUPPLIER  AND
INDEXES ALL;
commit work;
RUNSTATS ON TABLE TPCD.PART      AND INDEXES
ALL;
commit work;
RUNSTATS ON TABLE TPCD.PARTSUPP  AND
INDEXES ALL;
commit work;
RUNSTATS ON TABLE TPCD.CUSTOMER  AND
INDEXES ALL;
commit work;
RUNSTATS ON TABLE TPCD.ORDERS    AND
INDEXES ALL;
commit work;
RUNSTATS ON TABLE TPCD.LINEITEM  AND
INDEXES ALL;
COMMIT WORK;
```

B 11: dss.ddl.ri.hitachi

```
alter table TPCD.CUSTOMER add constraint pk primary
key (C_CUSTKEY);
alter table TPCD.NATION add constraint pk primary key
(N_NATIONKEY);
alter table TPCD.ORDERS add constraint fk foreign key
(O_CUSTKEY) references TPCD.CUSTOMER;
alter table TPCD.CUSTOMER add constraint fk foreign key
(C_NATIONKEY) references TPCD.NATION;
commit work;
```

B 12: dss.dbconfig.hitachi.30GB

```
update database configuration for tpcd using
buffpage 125000
catalogcache_sz 386
chngpgs_thresh 40
dbheap 6654
locklist 24000
maxlocks 5
logbufsz 64
logfilsiz 4094
logprimary 100
logsecond 25
maxappls 62
mincommit 1
num_iocleaners 10
num_ioservers 20
DLCHKTIME 5000
pckcachesz 640
softmax 100
sortheap 8000
stat_heap_sz 4384
stmtheap 10000
util_heap_sz 10000
applheapsz 3000
app_ctl_heap_sz 256
dft_degree 32
dft_queryopt 7
chngpgs_thresh 40;

get database configuration for tpcd;

--connect reset;
```

B 13: dss.dbmconfig.hitachi.30GB

```
update database manager configuration using
cpuspeed 2.6e-006
sheapthres 250000
agent_stack_sz 16
aslheapsz 15
rqrioblk 32767
intra_parallel yes
max_querydegree -1
maxagents 1500
num_poolagents 4
num_initagents 4
fcm_num_rqb 1024
fcm_num_buffers 1024
diaglevel 1;

get database manager configuration;

--connect reset;
```

B 14: db2set.run.bat

```
db2set DB2_ORDERED_NLJN=
db2set DB2_NEW_CORR_SQ_FF=Y
db2set
DB2BPVARS=d:\tpcd_audit_dir\ddl\osceola\dss.scattered-
read.config.hitachi.30GB
rem db2set DB2_NEW_TF_RANGE_FF=Y
rem db2set DB2BPNUMRANGEPREF=9
rem db2set DB2BPNUMRANGEPQS=1
rem db2set DB2BPNUMHATESTACKS=3
rem db2set DB2BPNUMPREFQUEUES=2
rem db2set DB2BPPREFQUEUESIZE=100
rem db2set DB2_SORT_BUFF=Y
rem db2set DB2_SORT_NEWORDER=yes
rem db2set DB2_MGJN_WITH_SORT=Y
db2set DB2_LIKE_VARCHAR=4.1
db2set DB2_HASH_JOIN=Y
db2set DB2_CORRELATED_PREDICATES=Y
db2set DB2_VECTOR=Y
db2set DB2_RR_TO_RS=ON
db2set DB2OPTIONS="-t -v +c"
db2set DB2NTNOCACHE=ON
```

B 15: loadDummyUFData.bat

```
echo "into loaddummyufdata.bat = %TPCD_DBNAME%"
call doUFload.bat 1 9
call doUFload.bat 2 9
```

B 16: doUFLoad.bat

```
REM Takes UFtype and update_pair as parameters.
echo "into doUFload.bat = %TPCD_DBNAME%"
call UFload.bat %1 %2
```

B 17: Load_UF1_data

```
:   # -*-Perl-*-
eval 'exec perl5 -S $0 ${1+"$@"}' # Horrible kludge to
convert this
    if 0;                       # into a "portable" perl script

# usage  perl loadUFD [update pair number]

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 (length($ENV{"TPCD_AUDIT_DIR"}) <= 0)
{
  die "TPCD_AUDIT_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_DBNAME"}) <= 0)
{
  die "TPCD_DBNAME 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
$auditDir=$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"};
$flatfilepath=$ENV{"TPCD_FLATFILES"};
$coldel="|";
$dblquote='"';


$PairNum=$ARGV[0];
#DJD $current_node=$ARGV[1];
#DJD print("current_node=$current_node\n");
#DJD for ($i=0, $nodestr="000"; $i<$current_node;
$i++,$nodestr++) {};

print "Beginning ....Preload of Update Function Data.
Lineitem \n";
system("db2 connect to $dbname\n");
$str="db2 \" load from  ";
$str="$str$flatfilepath${delim}lineitem.tbl.u" ;
$str="$str$PairNum.new";
$str="$str of del modified by coldel| fastparse messages
\\tmp\\TPCD\\line.msg.u";
$str="$str$PairNum.new replace into
TPCDTEMP.LINEITEM_NEW statistics yes nonrecoverable
\" " ;
print "$str \n";
$ret=system($str);
system("db2 commit\n");
if ($ret == 0)
{
   print "Preload Lineitem updates completed
successfully.\n";
}
else
{
   print "Preload Lineitem updates failed. ret=$ret\n";
   exit -1;
}

print "Beginning ....Preload of Update Function Data. Orders
\n";
system("db2 connect to $dbname\n");
$str="db2 \" load from  ";
$str="$str$flatfilepath${delim}order.tbl.u" ;
$str="$str$PairNum.new";
$str="$str of del modified by coldel| fastparse messages
\\tmp\\TPCD\\order.msg.u";
$str="$str$PairNum.new replace into
TPCDTEMP.ORDERS_NEW statistics yes nonrecoverable
\" " ;
print "$str \n";
$ret=system($str);
system("db2 commit\n");
if ($ret == 0)
{
   print "Preload Orders updates completed successfully.\n";
}
else
{
   print "Preload Orders updates failed. ret=$ret\n";
   exit -1;
}
```

B 18: Load_UF2_data

```
:  # -*-Perl-*-
eval 'exec perl5 -S $0 ${1+"$@"}' # Horrible kludge to
convert this
   if 0;                    # into a "portable" perl script

# usage  perl loadUFD [update pair number] [nodenumber]
if ( $ENV{TPCD_DBNAME} ==
${TPCD_QUAL_DBNAME} )
{
$db = $ENV{TPCD_QUAL_DBNAME};
}
else
{
$db = $ENV{TPCD_DBNAME};
}

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 (length($ENV{"TPCD_AUDIT_DIR"}) <= 0)
{
  die "TPCD_AUDIT_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_DBNAME"}) <= 0)
{
  die "TPCD_DBNAME 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
$auditDir=$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"};
$flatfilepath=$ENV{"TPCD_FLATFILES"};
$coldel="|";
$dblquote='"';


$PairNum=$ARGV[0];
#DJD $current_node=$ARGV[1];
#DJD print("current_node=$current_node\n");
#DJD for ($i=0, $nodestr="000"; $i<$current_node;
$i++,$nodestr++) {};


print "Beginning ....Preload of Update Function Data.
Deletes \n";
print "into load_uf2_data = $ENV{TPCD_DBNAME}\n";
$dbname=$ENV{TPCD_DBNAME};
system("db2 connect to $db \n");
$str="db2 \" load from  ";
$str="$str$flatfilepath${delim}delete." ;
$str="$str$PairNum.new ";
$str="$str of del modified by coldel| fastparse messages
\\tmp\\TPCD\\del.msg.u";
$str="$str$PairNum.new replace into
TPCDTEMP.ORDERS_DEL statistics yes nonrecoverable \"
" ;
print "$str \n";
$ret=system($str);
system("db2 commit\n");
if ($ret == 0)
{
   print "Preload Deletes updates completed successfully.\n";
}
else
{
   print "Preload Deletes updates failed. ret=$ret\n";
   exit -1;
}
```

B 19: DelDummyUFData.sql

```
DELETE FROM TPCDTEMP.ORDERS_NEW;
DELETE FROM TPCDTEMP.ORDERS_DEL;
DELETE FROM TPCDTEMP.LINEITEM_NEW;
commit;
```

B 20: Create_UF_Staging_Tables

```
-- connect to tpcd;

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

commit;

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

create unique index tpcdtemp.i_orders_new
 on tpcdtemp.orders_new
 (o_orderkey) include (app_id);

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

CREATE TABLE TPCDTEMP.LINEITEM_NEW ( APP_ID
INTEGER NOT NULL,
                L_ORDERKEY    BIGINT NOT NULL,
                L_PARTKEY     INTEGER NOT NULL,
                L_SUPPKEY     INTEGER NOT NULL,
                L_LINENUMBER  INTEGER NOT
NULL,
                L_QUANTITY    FLOAT NOT NULL,
```

```
                L_EXTENDEDPRICE  FLOAT NOT
NULL,
                L_DISCOUNT    FLOAT NOT NULL,
                L_TAX         FLOAT NOT NULL,
                L_RETURNFLAG  CHAR(1) NOT
NULL,
                L_LINESTATUS  CHAR(1) NOT NULL,
                L_SHIPDATE    DATE NOT NULL,
                L_COMMITDATE  DATE NOT NULL,
                L_RECEIPTDATE DATE NOT NULL,
                L_SHIPINSTRUCT CHAR(25) NOT
NULL,
                L_SHIPMODE     CHAR(10) NOT NULL,
                L_COMMENT      VARCHAR(44) NOT
NULL WITH DEFAULT)
      IN LINEITEM_TABLE
      INDEX IN LINEITEM_INDEXES
      PARTITIONING KEY(L_ORDERKEY) USING
HASHING;

COMMIT WORK;

alter table tpcdtemp.orders_new locksize table;
alter table tpcdtemp.orders_del locksize table;
alter table tpcdtemp.lineitem_new locksize table;

COMMIT WORK;

-- connect reset;
```

B 21: Dss.multiple.AST

```
connect reset;

change isolation to RR;
connect to tpcd;

drop table tpcd.l_summary;

create summary table tpcd.l_summary
as
(select
  l_shipdate, l_returnflag, l_linestatus, l_discount, l_quantity,
  sum(l_quantity)                          as s1,
  sum(l_extendedprice)                     as s2,
  sum(l_extendedprice*(1-l_discount))      as s3,
  sum(l_extendedprice*(1-l_discount)*(1+l_tax)) as s4,
  sum(l_discount)                          as s5,
  sum(l_extendedprice * l_discount)        as revenue,
  count(*)                                 as count

from tpcd.lineitem
group by l_shipdate, l_returnflag, l_linestatus, l_discount,
l_quantity)
data initially deferred
refresh immediate
IN LINEITEM_TABLE INDEX IN LINEITEM_INDEX
NOT LOGGED INITIALLY
;

refresh table tpcd.l_summary;
commit work;


create index tpcd.idx_l_sum on tpcd.l_summary (l_shipdate,
l_returnflag,
                              l_linestatus, l_discount,
l_quantity)
        pctfree 0;
commit work;

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


drop table tpcd.l_summary2;
commit;


connect reset;

change isolation to RR;
connect to tpcd;
drop table tpcd.l_summary2;
commit;


CREATE SUMMARY TABLE tpcd.L_SUMMARY2
  AS (
SELECT
  N_NAME, O_ORDERDATE, R_NAME, count(*) as
count,
  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
GROUP BY O_ORDERDATE, N_NAME, R_NAME)
DATA INITIALLY DEFERRED
REFRESH IMMEDIATE
IN LINEITEM_TABLE INDEX IN LINEITEM_INDEX
NOT LOGGED INITIALLY
;

refresh table tpcd.l_summary2;
commit;

create index tpcd.l_summary2_idx on tpcd.l_summary2 (
o_orderdate, r_name);
commit;

runstats on table tpcd.l_summary2 and indexes all;
commit;


drop table tpcd.PSSN_SUMMARY;
commit;

drop table tpcd.pssn_summary2;
commit;


CREATE SUMMARY TABLE TPCD.PSSN_SUMMARY
AS
(SELECT PS_PARTKEY, N_NAME,
SUM(PS_SUPPLYCOST*PS_AVAILQTY) AS VALUE,
      COUNT(*) AS COUNT
 FROM TPCD.PARTSUPP, TPCD.SUPPLIER,
TPCD.NATION
```

```
 WHERE PS_SUPPKEY = S_SUPPKEY
   AND S_NATIONKEY = N_NATIONKEY
 GROUP BY PS_PARTKEY, N_NAME)
DATA INITIALLY DEFERRED
REFRESH IMMEDIATE
IN OTHER INDEX IN OTHER
NOT LOGGED INITIALLY
;

REFRESH TABLE TPCD.PSSN_SUMMARY;
commit;


CREATE SUMMARY TABLE TPCD.PSSN_SUMMARY2
AS
(SELECT N_NAME,
SUM(PS_SUPPLYCOST*PS_AVAILQTY) AS SUM,
COUNT(*) AS COUNT
 FROM TPCD.PARTSUPP, TPCD.SUPPLIER,
TPCD.NATION
 WHERE PS_SUPPKEY = S_SUPPKEY
   AND S_NATIONKEY = N_NATIONKEY
 GROUP BY N_NAME)
DATA INITIALLY DEFERRED
REFRESH IMMEDIATE
IN OTHER INDEX IN OTHER
NOT LOGGED INITIALLY
;


REFRESH TABLE TPCD.PSSN_SUMMARY2;
commit;


CREATE INDEX TPCD.PSSN_SUMMARY_IDX ON
TPCD.PSSN_SUMMARY(N_NAME, PS_PARTKEY, value
desc);
commit;


CREATE INDEX TPCD.PSSN_SUMMARY2_IDX ON
TPCD.PSSN_SUMMARY2(N_NAME, sum);
commit;

RUNSTATS ON TABLE TPCD.PSSN_SUMMARY AND
INDEXES ALL;
commit;

RUNSTATS ON TABLE TPCD.PSSN_SUMMARY2 AND
INDEXES ALL;
commit;


connect reset;
terminate;
```

B 22: PloadUF1

```
:   # -*-Perl-*-
eval 'exec perl5 -S $0 ${1+"$@"}' # Horrible kludge to
convert this
    if 0;                       # into a "portable" perl script

# usage  perl

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)
{
    $PairNum=$ARGV[0];
}
else
{
    print "Update Pair not specified.\n";
}

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"};
$flatfilepath=$ENV{"TPCD_FLATFILES"};
$coldel="|";

  print "Beginning ....Load of Update Function Data for pair
$PairNum \n";
  system("db2 connect to $dbname\n");
  $str="db2 \" load from  ";
  $str="$str$flatfilepath${delim}lineitem.tbl.u" ;
  $str="$str$PairNum.new of del modified by coldel|
fastparse messages line.msg.u";
  $str="$str$PairNum.new replace into
TPCDTEMP.LINEITEM_NEW nonrecoverable \" " ;
  print "$str \n";
  $ret=system($str);
  if ($ret == 0)
  {
     print "Lineitem updates Loaded successfully.\n";
  }
  else
  {
     print "Lineitem updates failed. ret=$ret\n";
     exit -1;
  }


  $str="db2 \" load from  ";
  $str="$str$flatfilepath${delim}order.tbl.u" ;
  $str="$str$PairNum.new of del modified by coldel|
fastparse messages order.msg.u";
  $str="$str$PairNum.new replace into
TPCDTEMP.ORDERS_NEW nonrecoverable \" " ;
  print "$str  \n";
  $ret=system($str);
  if ($ret == 0)
  {
     print "Orders updates Loaded successfully.\n";
  }
  else
  {
     print "Orders updates failed. ret=$ret\n";
     exit -2;
  }
  system("db2 connect reset\n");

exit (0);
```

B 23: PloadUF2

```
:   # -*-Perl-*-
eval 'exec perl5 -S $0 ${1+"$@"}' # Horrible kludge to
convert this
    if 0;                       # into a "portable" perl script

# usage  perl ploadUF [pair_num]

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)
{
    $PairNum=$ARGV[0];
}
else
{
    print "Update Pair not specified.\n";
}

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"};
$flatfilepath=$ENV{"TPCD_FLATFILES"};
$coldel="|";
$dblquote="";


  print "Beginning ....Load of Update Function Data for pair
$PairNum \n";
  system("db2 connect to $dbname\n");
  $str="db2 \" load from ";
  $str="$str$flatfilepath${delim}\delete." ;
  $str="$str$PairNum.new of del modified by coldel|
fastparse messages del.msg.u";
  $str="$str$PairNum.new replace into
TPCDTEMP.ORDERS_DEL nonrecoverable \" " ;
  print "$str \n";
  $ret=system($str);
  if ($ret == 0)
  {
     print "Delete updates Loaded successfully.\n";
  }
  else
  {
     print "Delete updates failed. ret=$ret\n";
     exit -1;
  }

exit (0);
```

Appendix C: Query Validation EQT and Output

C 1: Query 1

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

Tag: Q1  Stream: 0  Sequence number: 1

SELECT
L_RETURNFLAG,
L_LINESTATUS,
SUM(L_QUANTITY) AS SUM_QTY,
SUM(L_EXTENDEDPRICE) AS SUM_BASE_PRICE,
SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)) AS
SUM_DISC_PRICE,
SUM(L_EXTENDEDPRICE*(1-
L_DISCOUNT)*(1+L_TAX)) AS SUM_CHARGE,
AVG(L_QUANTITY) AS AVG_QTY,
AVG(L_EXTENDEDPRICE) AS AVG_PRICE,
AVG(L_DISCOUNT) AS AVG_DISC,
COUNT(*) AS COUNT_ORDER
FROM TPCD.LINEITEM
WHERE L_SHIPDATE <= DATE('1998-12-01') - 90 DAYS
GROUP BY L_RETURNFLAG, L_LINESTATUS
ORDER BY L_RETURNFLAG, L_LINESTATUS

L_RETURNFLAG  L_LINESTATUS  SUM_QTY
SUM_BASE_PRICE       SUM_DISC_PRICE
SUM_CHARGE           AVG_QTY             AVG_PRICE
AVG_DISC             COUNT_ORDER
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------------------
A            F              3773034.000       5319329289.680
5053976845.784       5256336547.676            25.510
35964.013            0.050      147907
N            F               100245.000       141459686.100
134380852.769        139710306.872             25.625
36160.451            0.050      3912
N            O              7464940.000
10518546073.980      9992072944.461
10392414192.063            25.542            35990.126
0.050      292262
R            F              3779140.000       5328886172.990
5062370635.934       5265431221.821            25.549
36025.461            0.050      147920


Number of rows retrieved is:  4
```

C 2: Query 2

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

Tag: Q2  Stream: 0  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
LhiP0wiMzy72hlkP7l5in2y6RS6N130lz51nSRL5gOg5S26h
PCCQN2L
        9508.370  Supplier#000000070       FRANCE
3563  Manufacturer#1
M5C616R5h5SlMR3zzmLkSw24j2           16-821-608-
1166  m7z0CPShmBkhlChBAi3LkQ2CLw
mhl6QP362RPS3044CB2y41yhOhjlBin0CL7yhxmhS4hBM
07kQ1yyjOjz3C
        9508.370  Supplier#000000070       FRANCE
17268  Manufacturer#4
M5C616R5h5SlMR3zzmLkSw24j2           16-821-608-
1166  m7z0CPShmBkhlChBAi3LkQ2CLw
mhl6QP362RPS3044CB2y41yhOhjlBin0CL7yhxmhS4hBM
07kQ1yyjOjz3C
        9453.010  Supplier#000000802       ROMANIA
10021  Manufacturer#5
5yARQNSLNRAlOlBnkNQCik3SOlyClk7nmRhA2h0  29-
342-882-6463  65y3RQ2i0OP6Nz7mS hC
PxwLy7L1jQy6Ol63xO3iBCz52Rm1zm0MziCMLij2n6wky
51mBOwx Qh52iz QB1545Amxyj
        9453.010  Supplier#000000802       ROMANIA
13275  Manufacturer#4
5yARQNSLNRAlOlBnkNQCik3SOlyClk7nmRhA2h0  29-
342-882-6463  65y3RQ2i0OP6Nz7mS hC
PxwLy7L1jQy6Ol63xO3iBCz52Rm1zm0MziCMLij2n6wky
51mBOwx Qh52iz QB1545Amxyj
        9192.100  Supplier#000000115       UNITED
KINGDOM              13325  Manufacturer#1
h0m3lzlSPMw2B0ny7LNyNCkjRRn7iyMlLBLA     33-
597-248-1220  1QzQjhSyx
ixm2lgz2Ry7075RL3MS5z36x56hxmR0wLN0LBxm164Lz
CMmALzOAjn4kz7i4wjOlCONl1C51M7nCMx66SBRAQA
```

```
      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
[34 additional rows returned]


Number of rows retrieved is:  44
```

C 3: Query 3

Tag: Q3 Stream: 0 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
------------------------------
------------
```

C 4: Query 4

Tag: Q4 Stream: 0 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
```

C 5: Query 5

Tag: Q5 Stream: 0 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
------------------------------------------------
```

C 6: Query 6

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

Tag: Q6     Stream: 0   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
------------------------------------------------
```

C 7: Query 7

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

Tag: Q7     Stream: 0   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
------------------------------------------------
```

C 8: Query 8

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

Tag: Q8     Stream: 0   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
```

C 9: Query 9

Tag: Q9 Stream: 0 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
[165 additional rows returned]


Number of rows retrieved is:  175
```

C 10: Query 10

Tag: Q10 Stream: 0 Sequence number: 4

```
SELECT
C_CUSTKEY,
C_NAME,
SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)) AS
REVENUE,
C_ACCTBAL,
N_NAME,
C_ADDRESS,
C_PHONE,
C_COMMENT
FROM TPCD.CUSTOMER, TPCD.ORDERS,
TPCD.LINEITEM, TPCD.NATION
WHERE C_CUSTKEY = O_CUSTKEY
AND L_ORDERKEY = O_ORDERKEY
AND O_ORDERDATE >= DATE('1993-10-01')
AND O_ORDERDATE <  DATE('1993-10-01') + 3
MONTHS
AND L_RETURNFLAG = 'R'
AND C_NATIONKEY = N_NATIONKEY
GROUP BY C_CUSTKEY, C_NAME, C_ACCTBAL,
C_PHONE, N_NAME, C_ADDRESS, C_COMMENT
ORDER BY REVENUE DESC
FETCH FIRST 20 ROWS ONLY
```

```
C_CUSTKEY    C_NAME                    REVENUE
C_ACCTBAL           N_NAME                    C_ADDRESS
C_PHONE         C_COMMENT
------------------------------------------------------------------------
------------------------------------------------------------------------
------------------------------------------------------------------------
------------------------------------------------------------------------
      9722  Customer#000009722                  464618.258
474.040  CANADA                    l Mwzn4NAk6j
13-518-602-8070  5L 50Oy
RSgBAzPxmOSi5wk6xxOR7kh2nnPlgy7LBng2hOw5B01
RmCM120L24Pkg7PS1zwC11BCnz4L6i15PkixP26l66
     12800  Customer#000012800                  444265.642
1900.840  PERU
57zjB3CQx4P4OB2R2MBi2mwhSllM4mn 4 nC6     27-
142-205-3552
0hwglS77RB56Rx436lQ0N16CxhOPnmyhgwz
5z64wnj1kiC4jL350mM41y71hNxBllPjyA4hiN1wzjjM7SC
xAN244mk2A
      1025  Customer#000001025                  442028.022
3363.460  INDIA                     lkiSn154M5ROi
18-588-456-4616
0Bl45z233Rniw0OO064nPBgP16kimO0y74iLh73g1N4
m310 jQ yQzPA50iC 3MA75g2Bj162Nw4P
     13028  Customer#000013028                  441692.240
-452.660  UNITED KINGDOM
yP714ORSNgNN2LA3L5B                   33-253-660-2127
xPkmnhL2BkhkNyww4khlxwwAymN
h11PSjBCNMi50LkyOhO6CC 5nzOQCALzliOk2R66w
l05hRPO3iSP
      3694  Customer#000003694                  438180.070
2960.440  UNITED KINGDOM            2CCklmCBOCC
33-421-331-3127  MzLxQxLlLx3MPxlAwg1B5kg61zxkPnk
xiAm6PhMMAAQ2nzN3S6zzgP
x70w0lhhPx4QRzlMMy0204lAl3mBO7jh2jAP0N60wg367z
       976  Customer#000000976                  435897.632
7772.850  ROMANIA                   QzR 56Px1kgS
wANnAz02RS 30n Pm         29-436-660-4732  kzn32776
gwzkMzzzO4yxOAnkR7hR4R4x2SMwilz3x6h
nN7OnNLRMml3 kz5SLwi1yklOxiwS4g0wmA5A
4hmgBSwRRiQ1
      8206  Customer#000008206                  429905.110
6046.360  ARGENTINA                 P yMg30BBBBx
NMgC03AmzN2               11-571-859-1370
hLi122RMPmLC36Oy0kxO71zz2wCR0QQCl7z26hlQ3mM
     13532  Customer#000013532                  427731.804
-924.180  KENYA
6ij7M5PBMx2kwwyz62Oj4SL5S0mRCwl3m1Rmw     24-
525-332-7244  7ih7yRz214zO67AiNPx64nO5l5k
yj6i3jLA5PCL15Q4QlA3ll60iM1P iBxCixg6
1hCh2RCnjOzk5R OnO 1OhhC3m4631m5
     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
[10 additional rows returned]


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

C 11: Query 11

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

Tag: Q11   Stream: 0   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
[12 additional rows returned]


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

C 12: Query 12

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

Tag: Q12   Stream: 0   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
------------------------------------------------
```

C 13: Query 13

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

Tag: Q13   Stream: 0   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
------------------------------------------------
```

C 14: Query 14

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

Tag: Q14   Stream: 0   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
-----------------------------------------------
```

C 15: Query 15

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

Tag: Q15c   Stream: 0   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
-----------------------------------------------
```

C 16: Query 16

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

Tag: Q16    Stream: 0   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
[2752 additional rows returned]


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

C 17: Query 17

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

Tag: Q17    Stream: 0   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
-----------------------------------------------
```

Appendix D: Substitution Parameters and Seeds

D 1: Query Stream Seeds

The power streams executed and disclosed in this report used a seed value of 16494

D 2: Query Substitution Parameters

"Power stream Seed = 16494"
-- TPC-D Parameter Substitution (Version 1.3.1)
-- using 16494 as a seed to the RNG
Q1 DELTA 91
Q2 SIZE 26
TYPE STEEL
REGION AFRICA
Q3 SEGMENT FURNITURE
DATE 1995-03-30
Q4 DATE 1995-07-01
Q5 REGION ASIA
DATE 1997-01-01
Q6 DATE 1995-01-01
DISCOUNT 0.09
QUANTITY 24
Q7 NATION1 JAPAN
NATION2 UNITED KINGDOM
Q8 NATION JAPAN
REGION ASIA
TYPE PROMO POLISHED STEEL
Q9 COLOR magenta
Q10 DATE 1994-02-01
Q11 NATION JAPAN
FRACTION 0.0000033333
Q12 SHIPMODE1 TRUCK
SHIPMODE2 SHIP
DATE 1993-01-01
Q13 CLERK Clerk#000000513
Q14 DATE 1995-08-01
Q15 DATE 1995-07-01
Q16 BRAND Brand#53
TYPE SMALL ANODIZED
SIZE1 16
SIZE2 42
SIZE3 25
SIZE4 13
SIZE5 48
SIZE6 12
SIZE7 40
SIZE8 44
Q17 BRAND Brand#53
CONTAINER SM DRUM

Appendix E: Implementation Specific Layer and Drivers

E 1: tpcdbatch.sqc

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

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

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


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

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


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

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

#define TPCDBATCH_MAX_COLS    100
/* @d30369 tjg */

#define TPCDBATCH_CHAR char

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

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

#define PARALLEL_UPDATES 1

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


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


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

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

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

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

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

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

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

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


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

struct stmt_info
{
  long              max_rows_fetch;
  long              max_rows_out;
  int               query_block;                /* @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 SleepSome( int amount );
int get_env_vars(void);
int Get_SQL_stmt(struct global_struct *g_struct);

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

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

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

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

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

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

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


EXEC SQL INCLUDE SQLCA;

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

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

EXEC SQL END DECLARE SECTION;


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

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

```
char env_tpcd_copy_dir[150];
char env_tpcd_update_import[10];

/* Other globals */
FILE            *instream, *outstream;  /* File pointers
*/
int             verbose = 0;           /* Verbose option flag       */
int             updatePairStart;       /* update pair to start at
*/
int             currentUpdatePair;     /* update pair running
*/
int             updatePairStop;        /* update pair to stop
before */
char            newtime[50]="\0";      /* Des - moved from
get_time_stamp */
char            outstreamfilename[256]; /* store filename of
outstream
                                       wlc 081397 */
int             inlistmax = 400;       /* define # of keys to delete
at a time
                                       wlc 081897 */
int             sqlda_allocated = 0;   /* fixing free() problem in
NT
                                       wlc 090597 */
int            iImportStagingTbl=0;  /* IMPORT use import or
load (default) */


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


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

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

  /* TIME_ACC jen start */
#if defined (SQLUNIX) || defined (SQLAIX)
  struct global_struct g_struct =
  { NULL, NULL, NULL, NULL, {0,0}, {0,0}, "\0", 0.1,
"\0", 1, FALSE, 0,
    NULL, "\0", "\0", NULL };
#elif (defined (SQLOS2) || defined(SQLWINT) || defined
(SQLWIN) || defined(SQLDOS))
  struct global_struct g_struct =
  { NULL, NULL, NULL, NULL, {0,0,0,0}, {0,0,0,0}, "\0",
0.1, "\0", 1, FALSE, 0,
    NULL, "\0", "\0", NULL };
#else
#error Unknown operating system
#endif
  /* TIME_ACC jen end */

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


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

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

  init_setup(argc,argv,&g_struct);              /* @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_3,&start_time ));
/* TIME_ACC jen*/

    /* write the start timestamp to the file...if this is not a
qualification */
    /* run, then write the seed used as well */
    fprintf( outstream,"Start timestamp %*.*s \n",
           T_STAMP_3LEN,T_STAMP_3LEN,                /*
TIME_ACC jen*/
           g_struct.s_info_ptr->start_stamp);
    if (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  */

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

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

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

    return -1;
  }
#endif
  strcpy( env_tpcd_path_delim , "/" );  /*kmw*/
  if (strcpy(env_tpcd_run_on_multiple_nodes,
getenv("TPCD_RUN_ON_MULTIPLE_NODES")) ==
NULL) {
    fprintf(stderr, "\n The environment variable
$TPCD_RUN_ON_MULTIPLE_NODES");
```

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

   return 0;
}

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

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

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

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


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

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

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

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

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

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

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

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

       if (strstr(input_ln, "--#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  Stream: %d
Sequence number: %d\n",
                    g_struct->s_info_ptr->tag,g_struct->c_l_opt-
>intStreamNum,
                    g_struct->s_info_ptr->stmt_num);
/*jen0925*/
          fprintf(outstream,"\nTag: %-5.5s  Stream: %d
Sequence number: %d\n",
                  g_struct->s_info_ptr->tag,g_struct->c_l_opt-
>intStreamNum,
                  g_struct->s_info_ptr->stmt_num);
/*jen0925*/


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

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

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

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

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

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

    }  while (!eostmt);

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

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

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

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

    else                                        /* @d23684 tjg */
      return TPCDBATCH_NONSQL;

}  /*  Get_SQL_stmt */



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

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

  for (loopvar=0; loopvar<sqlda->sqld; loopvar++)
  {
    switch (sqlda->sqlvar[loopvar].sqltype)
    {
     case SQL_TYP_INTEGER:                      /*
INTEGER */
     case SQL_TYP_NINTEGER:
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)malloc(sizeof(long)))
== NULL)
         mem_error("allocating INTEGER");
       break;
     case SQL_TYP_BIGINT:                       /* BIGINT */
/*kmwBIGINT*/
     case SQL_TYP_NBIGINT:
#ifdef SQLWINT
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR
*)malloc(sizeof(__int64))) == NULL)
#else
       if ((sqlda->sqlvar[loopvar].sqldata=
```

```
                (TPCDBATCH_CHAR *)malloc(sizeof(signed
long long))) == NULL)
#endif
          mem_error("allocating BIGINT");
        break;
      case SQL_TYP_CHAR:                    /* CHAR */
      case SQL_TYP_NCHAR:
        if ((sqlda->sqlvar[loopvar].sqldata=
                (TPCDBATCH_CHAR
*)calloc(256,sizeof(char))) == NULL)
          mem_error("allocating CHAR/VARCHAR");
        break;
      case SQL_TYP_VARCHAR:                 /*
VARCHAR */
      case SQL_TYP_NVARCHAR:
        if ((sqlda->sqlvar[loopvar].sqldata=
                (TPCDBATCH_CHAR
*)calloc(4002,sizeof(char))) == NULL)
          mem_error("allocating CHAR/VARCHAR");
        break;
      case SQL_TYP_LONG:                    /* LONG
VARCHAR */
      case SQL_TYP_NLONG:
        if ((sqlda->sqlvar[loopvar].sqldata=
                (TPCDBATCH_CHAR
*)calloc(32702,sizeof(char))) == NULL)
          mem_error("allocating VARCHAR/LONG
VARCHAR");
        break;
      case SQL_TYP_FLOAT:                   /* FLOAT */
      case SQL_TYP_NFLOAT:
        if ((sqlda->sqlvar[loopvar].sqldata=
                (TPCDBATCH_CHAR
*)malloc(sizeof(double))) == NULL)
          mem_error("allocating FLOAT");
        break;
      case SQL_TYP_SMALL:                   /*
SMALLINT */
      case SQL_TYP_NSMALL:
        if ((sqlda->sqlvar[loopvar].sqldata=
                 (TPCDBATCH_CHAR *)malloc(sizeof(short)))
== NULL)
          mem_error("allocating SMALLINT");
        break;
      case SQL_TYP_DECIMAL:                 /*
DECIMAL */
      case SQL_TYP_NDECIMAL:
        if ((sqlda->sqlvar[loopvar].sqldata=
                 (TPCDBATCH_CHAR *)malloc(20)) ==
NULL)
          mem_error("allocating DECIMAL");
        break;
      case SQL_TYP_CSTR:                /* VARCHAR
(null terminated) */
      case SQL_TYP_NCSTR:
        if ((sqlda->sqlvar[loopvar].sqldata=
                (TPCDBATCH_CHAR
*)calloc(4001,sizeof(char))) == NULL)
          mem_error("allocating CHAR/VARCHAR");
        break;
      case SQL_TYP_DATE:                    /* DATE */
      case SQL_TYP_NDATE:
        if ((sqlda->sqlvar[loopvar].sqldata=
                (TPCDBATCH_CHAR
*)calloc(13,sizeof(char))) == NULL)
          mem_error("allocating DATE");
        break;
      case SQL_TYP_TIME:                    /* TIME */
      case SQL_TYP_NTIME:
        if ((sqlda->sqlvar[loopvar].sqldata=
                (TPCDBATCH_CHAR
*)calloc(11,sizeof(char))) == NULL)
          mem_error("allocating TIME");
        break;
      case SQL_TYP_STAMP:                   /*
TIMESTAMP */
      case SQL_TYP_NSTAMP:
        if ((sqlda->sqlvar[loopvar].sqldata=
                (TPCDBATCH_CHAR
*)calloc(29,sizeof(char))) == NULL)
          mem_error("allocating TIMESTAMP");
        break;
      }
      if ((sqlda->sqlvar[loopvar].sqlind=
              (short *)calloc(1,sizeof(short))) == NULL)
        mem_error("allocating indicator");

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


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


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

  int    col_type;                 /* Type of column           */

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

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


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

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

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

```
          break;

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

        case SQL_TYP_CHAR:
        case SQL_TYP_NCHAR:

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

        case SQL_TYP_SMALL:
        case SQL_TYP_NSMALL:

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

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

            temp_ptr = temp_string;

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

            else
              strcpy(decimal_string, " ");

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

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

          break;

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

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

    fprintf(outstream, "\n");

    return;
}


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

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

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

```
#endif

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



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


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

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

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

  }

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


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

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


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

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

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

    /* Decode the SQLCA in more detail  KBS 98/09/28 */
    if ((sqlca.sqlerrml)   /*  there's one or more tokens   */
       && (sqlca.sqlerrml < sizeof(sqlca.sqlerrmc)) /* and
field not full */
```

```
        )
    {
      char *tokptr;
      int  tokl;
      *(sqlca.sqlerrmc + sqlca.sqlerrml) = '\0';  /*  prevent
strtok from scanning beyond end */
      fprintf(stderr,"\n     SQLCA:   tokens:\n");
      fprintf(outstream,"\n     SQLCA:   tokens:\n");
      tokptr=strtok(sqlca.sqlerrmc, "\xff");
      while (  tokptr                          &&
            ( (tokl = (sizeof(sqlca.sqlerrmc) - (tokptr-
sqlca.sqlerrmc))) > 0)
            )
      {
        fprintf(stderr, "%.*s\n", tokl, tokptr);
        fprintf(outstream, "%.*s\n", tokl, tokptr);
        tokptr=strtok(NULL, "\xff");
      }
    }
    fprintf(stderr, "\n     SQLCA:   errp= %.8s, errd 1-6= %d
%d %d %d %d %d\n",
          sqlca.sqlerrp, sqlca.sqlerrd[0], sqlca.sqlerrd[1],
sqlca.sqlerrd[2],
          sqlca.sqlerrd[3], sqlca.sqlerrd[4], sqlca.sqlerrd[5]);
    fprintf(outstream, "\n     SQLCA:   errp= %.8s, errd 1-6=
%d %d %d %d %d %d\n",
          sqlca.sqlerrp, sqlca.sqlerrd[0], sqlca.sqlerrd[1],
sqlca.sqlerrd[2],
          sqlca.sqlerrd[3], sqlca.sqlerrd[4], sqlca.sqlerrd[5]);

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

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

    EXEC SQL CONNECT ;                                  /*
@d28763 tjg */

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

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


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

      for (i = 0; i < 6; i++)
      {
        fprintf(outstream, "sqlerrd[%d] = %lu \n", i,
temp_sqlca.sqlerrd[i]);
      }
      fprintf(outstream, "SQLWARN  = %0.11s\n",
temp_sqlca.sqlwarn);
      fprintf(outstream, "SQLSTATE = %0.5s\n",
temp_sqlca.sqlstate);

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

    }
  }
  return (temp_sqlca.sqlcode);

}  /* error_check */



/************************************************
*/
/* Displays a help screen                       */
/************************************************
*/
void display_usage()
{
  printf("\ntpcdbatch  --  version
%s",TPCDBATCH_VERSION);
  printf("\n\nSyntax is:\n");
  printf("tpcdbatch [-d dbname] [-f file_name] [-l file_name]
[-r on/off]");
  printf("\n       [-v on/off] [-b on/off] [-u p/t/t1/t2]");
  printf("\n       [-s scale_factor] [-n stream_num] [-m
inlistmax] [-h]\n");
  printf("\n  where: -d  Database name");
  printf("\n             Default - dbname set in
$DB2DBDFT");
  printf("\n       -f  Input file containing SQL statements");
  printf("\n             Default - stdin ");
  printf("\n       -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_1_opt-
>infile,argv[++loopvar]);
                     break;

            case 'l' :                                  /* @d26350 tjg */
            case 'L' :    strcpy(g_struct->c_1_opt-
>str_file_name,argv[++loopvar]);
                     break;

            case 'r' :                                  /* @d26350 tjg */
            case 'R' :
              if (!strcmp(uppercase(argv[++loopvar]),"ON"))
                g_struct->c_1_opt->outfile=1;
              else
                g_struct->c_1_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_1_opt->update=-1; /* init to invalid
number */
              if (!strcmp(uppercase(argv[++loopvar]),"P"))
                g_struct->c_1_opt->update=1;
              if (!strcmp(uppercase(argv[loopvar]),"T1"))
                   g_struct->c_1_opt->update=0;
              if (!strcmp(uppercase(argv[loopvar]),"T2"))
                   g_struct->c_1_opt->update=2;
              if (!strcmp(uppercase(argv[loopvar]),"T"))
                   g_struct->c_1_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]);
#ifdef UF2DEBUG
          fprintf (stderr, "updatePair = %d\n",updatePair);
          fflush(stderr);
#endif
          break;

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

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

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

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

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

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


#ifdef PARALLEL_UPDATES
    if (running_updates) {
      if (updatePair == -1) {
        fprintf (stderr, "The parameters to tpcdbatch have not
been passed correctly\n");
        exit (-1);
      }
      else {
        /* check to see if we are to use copy on for the load */
        if (( getenv("TPCD_LOG") != NULL ) &&

(!strcmp(uppercase(getenv("TPCD_LOG")),"YES")))
        {
          /* okay, we have set LOG_RETAIN on so we need to
use copy directory */
          copyOnOrOff = TRUE;
        }
        else
        {
          /* log retain off don't use copy directory */
          copyOnOrOff = FALSE;
        }

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

    /* If no database name is given, then use the one specified
in the
       environment variable DB2DBDFT, otherwise error */
    if (!strcmp(dbname,"\0")) {
      testptr = getenv("DB2DBDFT");
      if (testptr == NULL) {
```

```
        fprintf(stderr, "\nNo database name has been specified
on command ");
        fprintf(stderr, "line\nnor in environment variable
DB2DBDFT.");
        display_usage();
      }
      else
        strcpy(dbname,testptr);

    }

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

}



/************************************************
**/
/* Converts DECIMAL values to ASCII text          */
/************************************************
**/
int sqlrxd2a(                                  /*kmw*/
                                    /* C++ */char *decptr,
                                    /* C++ */char *asciiptr,
                                    short prec,
                                    short scal)
{/* */
  int allzero = TRUE;
  /* C++ */char *srcptr;
  unsigned char sign;
  /* C++ */char *targptr, decimal_point = '.';
  int rc = 0;                                  /*kmw*/
  int tmpint, src_nibble;
  int count, j, limit[3];

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


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


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

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

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

  }

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


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

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

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

/************************************************
*********************/
/* Start the mainline report processing.                  */
/************************************************
*********************/
  if (!strcmp(g_struct->c_l_opt->infile,"\0")) {
    instream=stdin;
  }
  else {
    instream=NULL;
    if ( (instream = fopen(g_struct->c_l_opt->infile,
```

```
READMODE)) == NULL ) {
        fprintf(outstream, "The input file could not be
opened.\n\n");
        fprintf(stdout,"Make sure that the filename is
correct.\n");
        fprintf(stdout,"filename = %s\n",g_struct->c_l_opt-
>infile);
        exit(-1);

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

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

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

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



   /* IMPORT (end) */

   /* we want to print the seed in the output files to show
what seed was */
   /* used to generate the queries.  */
   /* if intStreamNum is -1 then we are running a
qualification database */
   /* and the default seed has been used so skip this section */
   if (g_struct->c_l_opt->intStreamNum >= 0)
   {
     /* check to make sure the TPCD_RUNNUMBER
environment variable is set. We */
     /* use this and the stream number to determine which
seed was used to    */
     /* generate the current set of queries */
     if (getenv("TPCD_RUNNUMBER") == NULL)
     {
       fprintf(stderr,"\nThe TPCD_RUNNUMBER
environment variable is not set");
       fprintf(stderr,"....exiting\n");
       exit(-1);
     }
     if (getenv("TPCD_NUMSTREAM") == NULL)
     {
       fprintf(stderr,"\nThe TPCD_NUMSTREAM
environment variable is not set");
       fprintf(stderr,"....exiting\n");
       exit(-1);
     }


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

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

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

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

   /* check to see if we are to use copy on for the load */
   if (( getenv("TPCD_LOG") != NULL ) &&
```

```
      (!strcmp(uppercase(getenv("TPCD_LOG")),"YES")))
   {
     /* okay, we have set LOG_RETAIN on so we need to use
copy directory */
     g_struct->copy_on_load = TRUE;
   }
   else
   {
     /* log retain off don't use copy directory */
     g_struct->copy_on_load = FALSE;
   }
/************************************************
*********************/
/* Connect to the target database.  Start DBM if not already
done so. */
/************************************************
*********************/
   do {

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

     if (sqlca.sqlcode == SQLE_RC_NOSTARTG) {
       if (verbose)
         fprintf(stderr,"\nStarting the DB2 Database Manager
Now\n");
       sqlestar ();
       connect=0;
     }

     else
       connect=1;

   } while (!connect);

   error_check();

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

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

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

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

   /* if we are running a throughput test, then we must report
the */
   /* stream count information...we will report one file per
stream */
   /* and amalgamate them after all streams have completed
*/
   /* if the number of streams is greater than 0 then this is a
throughput test*/
   if ( g_struct->c_l_opt->intStreamNum > 0 )
   {
     if (g_struct->c_l_opt->update == 2 ||
        g_struct->c_l_opt->update == 5)
     {
       /* update function stream */
       sprintf(file_name,"%s%sstrcntuf.%s",g_struct-
>run_dir,
             env_tpcd_path_delim, g_struct->file_time_stamp);
     }
     else
     {
       /* query stream */
       sprintf(file_name, "%s%sstrcnt%d.%s",g_struct-
>run_dir, env_tpcd_path_delim,
             g_struct->c_l_opt->intStreamNum,g_struct-
>file_time_stamp);
     }
     if( (g_struct->stream_report_file = fopen(file_name,
WRITEMODE)) == NULL )
     {
       fprintf(stderr,"\nThe output file for the stream count
information\n");
       fprintf(stderr,"could not be opened, make sure the
filename is correct\n");
       fprintf(stderr,"filename = %s\n",file_name);
       exit(-1);
     }
     if (g_struct->c_l_opt->update == 2 ||
        g_struct->c_l_opt->update == 5)
     {
       /* update function stream */
       fprintf(g_struct->stream_report_file,
             "Update function stream starting at %*.*s\n",
             T_STAMP_3LEN,T_STAMP_3LEN, /*
TIME_ACC jen*/
             get_time_stamp(T_STAMP_FORM_3,&(g_struct-
>stream_start_time))); /* TIME_ACC jen*/
     }
     else
     {
       /* query stream */
       fprintf(g_struct->stream_report_file,
             "Stream number %d starting at %*.*s\n",
             g_struct->c_l_opt->intStreamNum,
             T_STAMP_3LEN,T_STAMP_3LEN,          /*
TIME_ACC jen*/
             get_time_stamp(T_STAMP_FORM_3,&(g_struct-
>stream_start_time))); /* TIME_ACC jen*/
     }
   }

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

   sprintf(g_struct->update_num_file,
"%s%s%s.%s.update.pair.num", env_tpcd_audit_dir,
         env_tpcd_path_delim, uppercase(env_tpcd_dbname),
lowercase(env_user));
   sprintf(g_struct->sem_file, "%s.%s.semfile",
env_tpcd_dbname, env_user);
   if (verbose) {   /* print out the update pair number file for
debugging */
     fprintf(stderr,"\n init_setup: strem %d update pair numb
```

```
file = %s\n",
          g_struct->c_l_opt->intStreamNum,g_struct-
>update_num_file);
  }

  /* update the
$TPCD_AUDIT_DIR/$TPCD_DBNAME.$USER.update.pa
ir.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_BIGINT:   /*kmwBIGINT*/
     case SQL_TYP_NBIGINT:
       col_lengths[col] = TPCDBATCH_MAX
(col_name_length,19);
       break;
     case SQL_TYP_CSTR:
     case SQL_TYP_NCSTR:
     case SQL_TYP_DATE:
     case SQL_TYP_NDATE:
     case SQL_TYP_TIME:
     case SQL_TYP_NTIME:
     case SQL_TYP_STAMP:
     case SQL_TYP_NSTAMP:
     case SQL_TYP_CHAR:
     case SQL_TYP_NCHAR:
     case SQL_TYP_VARCHAR:
     case SQL_TYP_NVARCHAR:
     case SQL_TYP_LONG:
     case SQL_TYP_NLONG:
       col_lengths[col] = TPCDBATCH_MAX
(col_name_length,col_width);
       break;

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

     case SQL_TYP_DECIMAL:
     case SQL_TYP_NDECIMAL:

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

       col_lengths[col] = TPCDBATCH_MAX ((m+n),
```

```
col_name_length);
        /* Special handling for DECIMAL */    /* @d26350 tjg
*/
        break;

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

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

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

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

}


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

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

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

   if ((form == T_STAMP_FORM_1) || (form ==
T_STAMP_FORM_3))
   {
     timeLength = strftime(newtime,50,"%x %X",tp);
     /* TIME_ACC jen start*/
     if (form == T_STAMP_FORM_3)
     {
       /* concatenate the microsecond/milliseconds  on the
end of the */
        /*timestamp jen1006 */
#if defined (SQLUNIX) || defined (SQLAIX)

sprintf(newtime+timeLength,".%0.6d",temp_stamp.tv_usec);
#elif (defined (SQLOS2) || defined(SQLWINT) || defined
(SQLWIN) || defined(SQLDOS))

sprintf(newtime+timeLength,".%0.3d",temp_stamp.millitm);
#else
#error Unknown operating system
#endif
        /* TIME_ACC jen stop*/
     }
   }
   else
     if (form == T_STAMP_FORM_2)
       strftime(newtime,50,"%y%m%d-%H%M%S",tp);

   return (newtime);

}



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

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

   double adjusted_a_mean = 0;
   double adjusted_g_mean = 0;

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

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

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


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


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

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

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

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

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

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

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

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

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

     s_info_ptr = s_info_ptr->next;

  }

  s_info_ptr = s_info_head_ptr;

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

     if (s_info_ptr->elapse_time != -1)  {
       s_info_ptr->adjusted_time = s_info_ptr->elapse_time;
       /* determine whether the elapsed times have to be
adjusted or not */
       /* if this is an update function, we do not adjust the
elapsed time*/
       if ( strstr(s_info_ptr->tag,"UF") == NULL )
       {
         /* this is not an update function, adjust time if
necessary */
         if (max->elapse_time/min->elapse_time > 1000)
           if (s_info_ptr->elapse_time < (max-
>elapse_time/1000))
             s_info_ptr->adjusted_time = max-
>elapse_time/1000;
       }

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

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

       if (s_info_ptr->elapse_time > 0) {  /* don't bother
finding log of
```

```
                                numbers < 0 */
          geo_mean += log(s_info_ptr->elapse_time);
          adjusted_g_mean += log(s_info_ptr->adjusted_time);
        }


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

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


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


    /* Calculate the arithmetic and geometric means */

    if (arith_mean != 0) {     /* Don't bother doing any of this if
the
                          elapsed time mean is 0 */
      arith_mean = arith_mean / num_stmt;
      adjusted_a_mean = adjusted_a_mean / num_stmt;
      geo_mean = exp(geo_mean / num_stmt_for_geo_mean);
      adjusted_g_mean = exp(adjusted_g_mean /
num_stmt_for_geo_mean);

    }


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

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

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

      /* Increase the Ts measurement by one second since the
accuracy of our */
      /* timestamps is only to 1 second and if the start was at
1.01 seconds, */
      /* and the end was at 5.99 seconds, we get a free second
... this will */
      /* be made explicit in the upcoming revision of the spec
(after 1.0.1) */
       /* TIME_ACC jen start*/
      /* NOTE this can probably be better coded by changing
get_elapsed_time */
      /* to just calculate the elapsed time give a start and an
end time, and */
      /* to also give a precision for the calculation (sec,
10ths....).  The */
      /* call then will grab a timestamp before calling. THen
we can get rid */
      /* of the if def...and just call get_elapsed_time (whcih
can handle the */
      /* os differences on its own */

#if defined (SQLUNIX) || defined (SQLAIX)
      Ts = g_struct->stream_end_time.tv_sec - g_struct-
>stream_start_time.tv_sec + 1;
#elif (defined (SQLOS2) || defined(SQLWINT) || defined
(SQLWIN) || defined(SQLDOS))
      Ts = g_struct->stream_end_time.time - g_struct-
>stream_start_time.time + 1;
#else
#error Unknown operating system
#endif
      /* TIME_ACC jen stop*/


      QppD = (3600 * g_struct->scale_factor) /
adjusted_g_mean;
      QthD = (num_stmt * 3600 * g_struct->scale_factor) / Ts;
      QphD = sqrt(QppD*QthD);

      /* if the decimal part has some meaningful value then
print the database size
      with decimal part; otherwise just print the integer part */

      if (db_size_frac_part < .01)
        fprintf (outstream,
            "QppD@%-.0f%-5.5s = %10.3f\n\nTs %11 =
%10.0f\n\nQthD@%-.0f%-5.5s = %10.3f\n\nQphD@%-
.0f%-5.5s = %10.3f\n",
            db_size,db_size_qualifier,QppD,
Ts,db_size,db_size_qualifier,
            QthD,db_size,db_size_qualifier, QphD);
      else
        fprintf (outstream,
            "QppD@%-.2f%-5.5s = %10.3f\n\nTs %11 =
%10.0f\n\nQthD@%-.2f%-5.5s = %10.3f\n\nQphD@%-
.2f%-5.5s = %10.3f\n",
            db_size,db_size_qualifier,QppD,
Ts,db_size,db_size_qualifier,
            QthD,db_size,db_size_qualifier, QphD);
    }

}

/*************************************************
**************/
/* free up all the elements of the sqlda after done processing
```

```
*/
/*********************************************
*************/
void free_sqlda (struct sqlda *sqlda, int select_status)    /*
@d30369 tjg */
{
  int loopvar;

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

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


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

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


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

#ifdef SQLWINT
  /* PROCESS_INFORMATION childprocess[100]; */
  char commandline[256];
  HANDLE              su_hSem;
  char               UF1_semfile[256];
#else
  int childpid[100];
  int                su_semid; /* semaphore for controlling split
updates*/
  key_t              su_semkey; /* key to generate semid */
#endif

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

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

  /* Load the orderkeys into the staging table */
  /* In SMP environments one could use a load command
but by using a */
  /* script we can keep the code common */
#ifdef SQLWINT
  sprintf (statement, "perl %s\\tools\\ploaduf1 %d\n",
sourcedir, updatePair);
  printf("load_uf1 statement = %s\n",statement);
#else
  sprintf (statement, "%s/tools/load_uf1 %d 1", sourcedir,
updatePair);
#endif
  if (system(statement))
  {
    fprintf (stderr, "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");

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

#ifndef SQLWINT
  /*    we will use the first flat file to generate the semaphore
key */
  if (getenv("TPCD_FLATFILES") != NULL)
  {
#ifndef TPCD_NONPARTITIONED

    /* this is assuming that you will be running this from 0th
node */
    sprintf(sourcefile, "%s%corder.tbl.u%d.00000.new",
        getenv("TPCD_FLATFILES"),
PATH_DELIM,updatePair);
#else
    sprintf(sourcefile, "%s%corder.tbl.u%d.0.new",
        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, "Cannot get semaphore! semget failed:
errno = %d\n",errno);
    exit (-1);
```

```
  }
#else  /* SQLWINT */
  sprintf (UF1_semfile, "%s.%s.UF1.semfile",
env_tpcd_dbname, env_user);
  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);   aph 981205 */
      runUF1_fn (updatePair, i, dbname, userid, passwd);
    }
    else
    {
      /* This is the parent */
      if (verbose)
        fprintf (stderr, "stream #%d started with pid %d\n", i,
childpid[i]);
    }
#else  /* SQLWINT */
    sprintf (commandline,
          "start /b %s\\auditruns\\tpcdbatch.exe -z -d %s -i
%d -j 1 -k %d",
          env_tpcd_audit_dir, dbname, updatePair, i ); /* aph
082797 */

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

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

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

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


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


/*************************************************/
/* processing to run the delete update function */
/*************************************************/
void runUF2 ( int updatePair )
{
  char statement[3000];
  char sourcedir[256];

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

  int i, j, c;
  char myoutstreamfile[256];
  FILE *myoutstream;
  char *buffer;
#ifdef SQLWINT
  char commandline[256];
  HANDLE              su_hSem;
  char              UF2_semfile[256];
#else
  int childpid[100];
```

```
  char sourcefile[256];
  int              su_semid; /* semaphore for controlling split
updates*/
  key_t            su_semkey; /* key to generate semid */
#endif

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

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

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

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

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

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


#ifdef SQLWINT
  sprintf (statement, "perl %s\\tools\\ploaduf2 %d\n",
sourcedir, updatePair);
  printf("load_uf2 statement = %s\n",statement);
#else
  sprintf (statement, "%s/tools/load_uf2 %d 1", sourcedir,
updatePair);
#endif
  if (system(statement))
  {
    fprintf (stderr, "load_uf2 failed for UF2, examine
UF2.log for cause. Exiting.\n");
    exit (-1);
  }
  fprintf (outstream, "load_uf2 finished for UF2.\n");

  /* Next we need to get ready to launch a bunch of
concurrent processes */
#ifndef SQLWINT
  /*   we will use the first flat file to generate the semaphore
key */
  if (getenv("TPCD_FLATFILES") != NULL)
    sprintf(sourcefile, "%s%cdelete.%d.00000.new",
        getenv("TPCD_FLATFILES"), PATH_DELIM,
updatePair);
  else
    sprintf(sourcefile,".%cdelete.%d.00000.new",
PATH_DELIM,
        updatePair);

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

  for (i=0; i < concurrent_deletes; i++)
  {
#ifndef SQLWINT
    if ((childpid[i] = fork()) == 0)
    {
      fprintf(stderr, "B-Calling runUF2_fn  %d   %d   %d
...\n",
                    updatePair,
i,chunks_per_concurrent_delete);
      /* runUF2_fn (updatePair, i,
chunks_per_concurrent_delete);   aph 981205 */
      runUF2_fn (updatePair, i,
chunks_per_concurrent_delete, dbname, userid, passwd);
    }
    else
    {
      /* This is the parent */
      if (verbose)
        fprintf (stderr, "stream #%d started with pid %d\n", i,
childpid[i]);
    }
#else
    {
      /*  SECURITY_ATTRIBUTES sec_process;
        SECURITY_ATTRIBUTES sec_thread; */
      /* NEED TO FIX THIS UP - KBS 98/10/20 */

      sprintf (commandline,
        "start /b %s\\auditruns\\tpcdbatch.exe -z -d %s -i
```

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

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

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

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

   }


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


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

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

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


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

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

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

   switch (g_struct->c_l_opt->intStreamNum)
   {
```

```
    case -1: /* qualifiying */
      sprintf(file_name,
"%s%sqryqual%02d.%s",run_dir,delim,qnum,time_stamp);
      break;
    case 0: /* power tests */
      if (qnum < 0) /* update functions */
        sprintf(file_name,
"%s%smpuf%d.%02d.%s",run_dir,delim,abs(qnum), \
                currentUpdatePair,time_stamp);
      else
        sprintf(file_name,
"%s%smpqry%02d.%s",run_dir,delim,qnum,time_stamp);
      break;

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

  }

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

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

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

  return;
}


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

  switch (g_struct->c_flags->select_status)
  {
   case TPCDBATCH_NONSQL:
    g_struct->s_info_stop_ptr = g_struct->s_info_ptr;
    /* if we're at the end of the input file, set the stop
      pointer to this structure */
    rc = FALSE;
    break;
   case TPCDBATCH_EOBLOCK:
    rc = FALSE;
    break;
   case TPCDBATCH_INSERT:
    /* we have to check whether or not this is a throughput */
    /* test, and if it is, we have to set up a semaphore to */
    /* control when the update functions are run.  We want
*/
    /* them to be run after all the query streams have
finished. */
    /* What we do is set up the semaphore here, decrement it
*/
    /* in the query streams, and wait for it to get cleared */
    /* before we allow the UFs to run.                      */
    /* Note: we only set up the semaphore if:               */
    /*       1. we are running the throughput test (num of */
    /*          streams > 0)                                */
    /*       2. we are at the first UF1 (i.e. this is the  */
    /*          case where currentUpdatePair = updatePairStart
*/
    /* we also want to check the sem_on element in the
global */
    /* structure to see if we want to use semaphores or let */
    /* the calling script do the synchronization of the update
*/
    /* stream                                               */
    if ( ( g_struct->sem_on == 1 ) &&
```

```
        ( g_struct->c_l_opt->intStreamNum != 0 ) )
    {
      /* yes we are to be using semaphores */
      /* is this the 1st time into update function 1??? */
      if (currentUpdatePair == updatePairStart )
      {
#ifdef SQLWINT
        fprintf(stderr,"numstreams = %d\n",g_struct-
>c_l_opt->intStreamNum);
        fprintf(stderr,"semfile = %s\n",g_struct->sem_file);
        hSem = CreateSemaphore(NULL, 0,
                        g_struct->c_l_opt->intStreamNum,
                        (LPCTSTR)(g_struct->sem_file));
        if (hSem == NULL)
        {
          fprintf(stderr,
                "CreateSemaphore (ready semaphore) failed,
GetLastError: %d, quitting\n",
                GetLastError());
          exit(-1);
        }

        fprintf(stderr,"Semaphore created successfully!\n");
        for (j = 0; j < g_struct->c_l_opt->intStreamNum;
j++)
        {
          if (verbose)
            fprintf(stderr,"About to wait again ...\n");
          if (WaitForSingleObject(hSem, INFINITE) ==
WAIT_FAILED)
          {
            fprintf(stderr,
                 "WaitForSingleObject (hSem) failed on
stream %d, error: %d, quitting\n",
                 j, GetLastError());
            exit(-1);
          }
          if (verbose)
            fprintf(stderr,"Streams to wait for  %d\n", j);
        }
        fprintf(stderr,"Done waiting! Ready to run
updates!\n");
        /* close the semaphore handle */
        if (! CloseHandle(hSem)) {
          fprintf(stderr, "Close Sem failed - Last Error:
%d\n", GetLastError());
          /* no exit here */
        }
#else
        /* create a semaphore key...use the name of a file that
*/
        /* you know exists */

        semkey = ftok(g_struct->update_num_file,'J');
        if ( (semid =
semget(semkey,1,IPC_CREAT|S_IRUSR|S_IWUSR)) < 0)
        {
          fprintf(stderr,
                "Throughput can't get initial semaphore!
semget failed errno = %d\n",
                errno);
          exit(1);
        }
        if (verbose)
        {
          fprintf(stderr,
                "insert: semkey = %ld, semid = %d, file = %s,
value = %d\n",
                semkey,semid,g_struct->update_num_file,
                (g_struct->c_l_opt->intStreamNum * -1));
        }

        /* call the sem_op routine to decrement the
semaphore by */
        /* however many streams .... by calling this function
with*/
        /* a negative number, this stream is forced to wait
until */
        /* the semaphore gets back to 0 */
        if (sem_op(semid, 0, (g_struct->c_l_opt-
>intStreamNum * -1)) != 0)
        {                                       /*jenSEM*/
          fprintf(stderr,
                "Failure to wait on throughput semaphone for
%d streams\n",
                g_struct->c_l_opt->intStreamNum);
          exit(1);
        }                                       /*jenSEM*/
        fprintf(stderr,"finished waiting on stream
semaphore\n");
        semctl(semid,0,IPC_RMID,0);  /* we've finished
waiting, now */
                                /* remove the semaphore */
#endif
      }
      /* otherwise continue to run*/
    }

    if (g_struct->c_l_opt->update != 5)
    {
      runUF1(currentUpdatePair);
    }
    rc = FALSE;
    break;
  case TPCDBATCH_DELETE:
    if (g_struct->c_l_opt->update != 5) {
      runUF2(currentUpdatePair);
    }
    currentUpdatePair += 1;
    /* update the update.pair.num file to reflect the
successfully completed */
    /* update pair */
    if (g_struct->c_l_opt->update != 5)
    {   /*jen*/
#ifndef NO_INCREMENT
      /* don't update the pair, only for my testing - Haider */
      updateFP = fopen(g_struct->update_num_file,"w");
      fprintf(updateFP,"%d\n",currentUpdatePair);
      fclose(updateFP);
#endif
    }  /*jen*/
    rc = FALSE;
    break;

  }
  return(rc);
}


/*****************************************************
*******************/
/* Handles actual processing of SQL statement.  Initializes
the SQLDA
   for returned rows, does PREPARE, DECLARE, and OPEN
statements and
   executed multiple FETCHes as needed.  If not a SELECT
```

```
statement,
   goes into EXECUTE IMMEDIATE section
*/
/************************************************
*******************/
void SQLprocess(struct global_struct *g_struct)
{
  int rc = 0;                          /* 912RETRY */
  int rows_fetch = 0;
  long sqlcode = SQL_RC_E911;              /* Temporary
sqlcode to test
                                       for deadlocks */
  int max_wait = 1;                        /* Maximum number of
retries
                                       for deadlock scenario */

  int col_lengths[TPCDBATCH_MAX_COLS];     /* array
containing widths of
                                       columns in returned set  */
  struct stmt_info *s_info_ptr;

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

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


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

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

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

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

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

        EXEC SQL DECLARE DYNCUR CURSOR FOR
STMT1;

        EXEC SQL OPEN DYNCUR;
        sqlcode = error_check();

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

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

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

          rows_fetch = 0;

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

            EXEC SQL FETCH DYNCUR USING
DESCRIPTOR :*sqlda;
            if (sqlca.sqlcode == 100)
            {
              sqlcode = sqlca.sqlcode;
            }
            else
            {
              sqlcode = error_check();
            }
            if (sqlcode == 0)
            {
              rows_fetch++;
              if ( (rows_fetch <= s_info_ptr->max_rows_out)
||
                   (s_info_ptr->max_rows_out == -1) )
                echo_sqlda(sqlda,col_lengths);
            }
            else if (sqlcode < 0)
            {
              max_wait++;
              fprintf (stderr, "\nAn error has been detected on
fetch...Retrying...\n");
              SleepSome(10);
            }
          }  while ( (sqlcode == 0) && \
```

```
                    ( (s_info_ptr->max_rows_fetch == -1) || \
                      (rows_fetch < s_info_ptr-
>max_rows_fetch) ) );
          } /* end of successful open */
        } /* end of successful prepare */
      } /** End of block for handling SELECT statements **/

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

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

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

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

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

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

} /* end of SQLprocess */

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

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

  s_info_ptr = g_struct->s_info_ptr;

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

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

  s_info_ptr->elapse_time = get_elapsed_time(start_time);

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

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

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


  fflush(outstream);

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

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

    s_info_ptr = s_info_ptr->next;

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

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

  if (sqlda_allocated)
    free_sqlda(sqlda,g_struct->c_flags->select_status);
    /* fix free() problem on NT
      wlc 090597 */
```

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

  return (TRUE);
}

/************************************************
*****************************/
/* Does some cleaning up once all the statements are
processed.  Disconnects
   from the database, cleans up some semaphore stuff from
the update functions,
   prints out the summary table, and closes all file handles.
*/
/************************************************
*****************************/
int cleanup(struct global_struct *g_struct)
{
#ifndef SQLWINT
  int               semid;              /* semaphore for controlling
UFs*/
  key_t             semkey;             /* key to generate semid
*/
#else
  HANDLE            hSem;
#endif

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

  if (g_struct->c_l_opt->intStreamNum > 0)
  {
    /* print out the stream stop time in the stream count
information file*/
    /* for the throughput tests only (intStreamNum>0)*/
    if (g_struct->c_l_opt->update == 2 ||
        g_struct->c_l_opt->update == 5)
    {
      /* update function stream */
      fprintf(g_struct->stream_report_file,
            "Update function stream stopping at %*.*s\n",
            T_STAMP_3LEN,T_STAMP_3LEN,  /*
TIME_ACC jen*/

get_time_stamp(T_STAMP_FORM_3,&(g_struct-
>stream_end_time)));  /* TIME_ACC jen*/
    }
    else
    {
      /* query streams */
      fprintf(g_struct->stream_report_file,
            "Stream number %d stopping at %*.*s\n",
            g_struct->c_l_opt->intStreamNum,
            T_STAMP_3LEN,T_STAMP_3LEN,  /*
TIME_ACC jen*/

get_time_stamp(T_STAMP_FORM_3,&(g_struct-
>stream_end_time)));  /* TIME_ACC jen*/
    }
    fclose(g_struct->stream_report_file);
  }


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

  /* if we are in a query stream AND this is a throughput
test, then need */
  /* do to some semaphore stuff   (0 implies update functions
are off) */
  /* AND we are supposed to be using semaphores */
  if ( ( g_struct->sem_on == 1 ) &&
       ( g_struct->c_l_opt->update == 0) &&
       ( g_struct->c_l_opt->intStreamNum > 0 ) )
  {



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

#ifndef SQLWINT

    semkey = ftok(g_struct->update_num_file,'J');
    /* Okay, now's the time to increment the semaphore by
the update stream */
    while ((semid = semget(semkey,1,0)) < 0)
    {
      if (errno == ENOENT)
      {
        sleep(2);
        fprintf(stderr,"cleanUp: looping for access to
semaphore stream %d  ",
              g_struct->c_l_opt->intStreamNum);
        fprintf(stderr,"semkey=%ld semid = %d
file=%s\n",semkey,semid,
              g_struct->update_num_file);
      }
      else {
        fprintf(stderr,"query stream %d semget failed errno =
%d\n",
             g_struct->c_l_opt->intStreamNum,errno);
        exit(1);
      }
    }
    if (verbose)
    {
      fprintf(stderr,
            "cleanup: semkey = %ld, semid = %d, file = %s,
stream = %d\n",
            semkey,semid,g_struct->update_num_file,
            g_struct->c_l_opt->intStreamNum);
    }
    /* this stream is done...increment by one */
    if (sem_op(semid, 0, 1) != 0)
/*jenSEM*/
    {                                          /*jenSEM*/
      fprintf(stderr,
            "Failed to increment semaphone for throughput
stream %d\n",
            g_struct->c_l_opt->intStreamNum);
      fprintf(stderr,
            "file for generation of semaphore is: %s\n",
            g_struct->update_num_file);
      exit(1);
    }                                           /*jenSEM*/


#else
    while ((hSem =
OpenSemaphore(SEMAPHORE_ALL_ACCESS |
                  SEMAPHORE_MODIFY_STATE |
```

```
                        SYNCHRONIZE,
                        TRUE,
                        g_struct->sem_file))
                == (HANDLE)(NULL)) {
        /*
        ** if cannot open the semaphore, wait for 0.1 second
        */
        fprintf(stderr,"Retry Open semaphore %s\n",g_struct-
>sem_file);

        Sleep(1000);
    }

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


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

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

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

  return (TRUE);
}
```

E 2: tpcduf.sqc

```
/*************************************************
****************************
*
*  TPCDUF.SQC
*
*  Revision History:
*
*  05 dec 98 aph Created tpcdUF.sqc containing
runUF1_fn() and runUF2_fn()
*               so that it can be bound separately with a different
isolation level.
*
*************************************************
****************************/

/* #define UF1DEBUG */

#include "tpcdbatch.h"
EXEC SQL INCLUDE SQLCA;

/*************************************************
********************/
/* Function Prototypes                                  */
/*************************************************
********************/
extern int SleepSome( int amount );
extern long error_check(void);                          /*
@d28763 tjg */
extern void dumpCa(struct sqlca*);      /*kmw*/
extern int sem_op (int semid, int semnum, int value);
extern char *get_time_stamp(int form, Timer_struct
*timer_pointer);     /* TIME_ACC jen */

/*************************************************
*****************/
/* Declare the SQL host variables.                      */
/*************************************************
*****************/
EXEC SQL BEGIN DECLARE SECTION;
  char UF_dbname[9] = "\0";
  char UF_userid[9] = "\0";
  char UF_passwd[9] = "\0";
  long UF_chunk = 0;         /* jenCI counter for within the
set of chunks*/
EXEC SQL END DECLARE SECTION;

/*************************************************
*****************/
/* Declare the global variables.                        */
/*************************************************
*****************/
extern char env_tpcd_tmp_dir[150];
extern FILE *instream, *outstream;  /* File pointers */
extern char sourcefile[256];    /* Used for semaphores and
table functions?*/
extern struct {                         /* jen LONG */
            short len;
            char data[32700];
           } stmt_str;                  /* jen LONG */


/*************************************************
*****************/
/* UF1 child                                            */
/* (i is the application number.)                       */
/*************************************************
*****************/
void runUF1_fn ( int updatePair, int i, char *dbname, char
*userid, char *passwd )
{
  int rc = 0;
  int split_updates = 2;      /* no. of ways update records are
split */
  int concurrent_inserts = 2;  /* jenCI no of concurrent
updates to be */
                        /* jenCI run at once*/
  int loop_updates = 1;       /* jenCI no of updates to be run
in one  */
                        /* jenCI "concurrent" invocation.
should*/
                        /* jenCI be split_updates /
concurrent_inserts*/
  int startChunk = 0;         /* jenCI number of first chunk to
insert for */
                        /* jenCI this child */
  int stopChunk = 0;          /* jenCI number of last chunk to
insert for */
                        /* jenCI this child */
  long insertedLineitem = 0;  /*kmw*/
  long insertedOrders = 0;    /*kmw*/
  long saveInsertedOrders = 0; /*kbs*/

  long sqlcode;
  int maxwait;
```

```
#ifndef SQLWINT
  int               su_semid;
  key_t             su_semkey;
#else
  HANDLE            su_hSem;
  char              UF1_semfile[256];
#endif

  char myoutstreamfile[256];
  FILE *myoutstream;

  strcpy(UF_dbname, dbname);
  strcpy(UF_userid, userid);
  strcpy(UF_passwd, passwd);

  /* Get ready to start logging diagnostic output */
  sprintf (myoutstreamfile, UF1OUTSTREAMPATTERN,
env_tpcd_tmp_dir, PATH_DELIM,
          updatePair, i);
  if ( (myoutstream = fopen (myoutstreamfile,
WRITEMODE)) == NULL)
  {
    fprintf (stderr, "\nThe output file '%s' for update pair %d
set %d could not be opened. runUF1_fn\n",
            myoutstreamfile,updatePair,i);
    rc=-1;
    goto UF1_exit;
  }
  outstream=myoutstream;    /* initialize outstream for
error_check dxxxxhar*/

  fprintf( myoutstream,"\nUF1 for update pair %d set %d
starting at %*.*s\n",
          updatePair, i,
          T_STAMP_1LEN,T_STAMP_1LEN,  /*
TIME_ACC jen*/
          get_time_stamp(T_STAMP_FORM_1,(Timer_struct
*)NULL));    /* TIME_ACC jen*/


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

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


  /* Establish a connection to the database */
  if (!strcmp(userid,"\0"))   /** No authentication provided
**/
    EXEC SQL CONNECT TO :UF_dbname;
  else
    EXEC SQL CONNECT TO :UF_dbname USER
:UF_userid USING :UF_passwd;
  error_check();
  if (sqlca.sqlcode < 0)
  {
    rc=-1;
    goto UF1_exit;
  }

  /* Start processing each chunk in my range */
#ifdef UF1DEBUG
  fprintf (myoutstream,"Before loop_a startChunk = %d,
stopChunk = %d\n", startChunk, stopChunk);
  fflush(myoutstream);
#endif
  for ( UF_chunk = startChunk; UF_chunk <= stopChunk;
UF_chunk++ )           /*jenCI*/
  {                                                  /*jenCI*/
    /* wlc  062797 */
    sqlcode = SQL_RC_E911;
    maxwait = 1;
    rc = 0;

#ifdef UF1DEBUG
  fprintf (myoutstream, "Before While_a Chunk= %d
\n",UF_chunk);
  fflush(myoutstream);
#endif
    /* Loop to handle any deadlocks */
    while (sqlcode == SQL_RC_E911 && maxwait <=
MAXWAIT && rc==0)
    {
      sqlcode = 0;
#ifdef UF1DEBUG
  fprintf (myoutstream, "in loop before orders exec sql\n");
  fflush(myoutstream);
#endif
      EXEC SQL INSERT INTO TPCD.ORDERS
          SELECT
O_ORDERKEY,O_CUSTKEY,O_ORDERSTATUS,O_TO
TALPRICE,

O_ORDERDATE,O_ORDERPRIORITY,O_CLERK,O_SHI
PPRIORITY,O_COMMENT
          FROM TPCDTEMP.ORDERS_NEW
          WHERE APP_ID = :UF_chunk;

      if (sqlca.sqlcode < 0)
        sqlcode = error_check();

      if (sqlcode == SQL_RC_E911)
      {       /* we've hit a deadlock */
        fprintf (myoutstream,
            "\nDeadlock detected inserting from
tpcdtemp.orders_new for chunk %d for pair
%d..Retrying...\n",UF_chunk,updatePair);
        SleepSome(UF_DEADLOCK_SLEEP);
        maxwait++;                    /* jen DEADLOCK
*/
      }
      else if (sqlcode < 0)
      {
        fprintf(myoutstream,
            "Insert into orders pair %d chunk %d failed
sqlcode=%d\n",
            updatePair,UF_chunk,sqlcode);
        dumpCa(&sqlca);
        rc = -1;
      }
```

```
      else
      {
        /* Everything worked with ORDERS, proceed with
LINEITEM */
        saveInsertedOrders = sqlca.sqlerrd[2];

        sqlcode = 0;
#ifdef UF1DEBUG
        fprintf (myoutstream, "in lineitem for update pair
number %d set %d chunk %d\n",
                updatePair, i,UF_chunk);
        fflush(myoutstream);
#endif

        EXEC SQL INSERT INTO TPCD.LINEITEM
          SELECT
L_ORDERKEY,L_PARTKEY,L_SUPPKEY,L_LINENUM
BER,L_QUANTITY,
          L_EXTENDEDPRICE,L_DISCOUNT,L_TAX,

L_RETURNFLAG,L_LINESTATUS,L_SHIPDATE,L_COM
MITDATE,L_RECEIPTDATE,

L_SHIPINSTRUCT,L_SHIPMODE,L_COMMENT
          FROM TPCDTEMP.LINEITEM_NEW WHERE
APP_ID = :UF_chunk;

        if (sqlca.sqlcode < 0)
          sqlcode = error_check();

        if (sqlcode == SQL_RC_E911)
        {       /* we've hit a deadlock */
          fprintf (myoutstream,
                "\nA deadlock has been detected inserting
from tpcdtemp.lineitem%d_%d...Retrying...\n",
                updatePair, UF_chunk);
          SleepSome(UF_DEADLOCK_SLEEP);
          maxwait++;                    /* jen
DEADLOCK */
        }
        else if (sqlcode < 0)
        {
          fprintf(myoutstream,
                "Insert into lineitem pair %d chunk %d failed
sqlcode=%d\n",
                updatePair,UF_chunk,sqlcode);
          dumpCa(&sqlca);
          rc = -1;
        }
        else
        {
#ifdef UF1DEBUG
          fprintf (myoutstream, "lineitem insert
succeeded\n");
          fflush(myoutstream);
#endif
          /* accumulate the number of row inserted */
          /* Order count ONLY updated if both orders and
lineitem */
          /* go through */
          insertedOrders += saveInsertedOrders;         /*
kbs */
          insertedLineitem += sqlca.sqlerrd[2];
          rc=0;
          EXEC SQL COMMIT WORK;
          error_check();

#ifdef UF1DEBUG
          /* report the number of row inserted */
          fprintf(myoutstream, "    interim %ld rows for
chunk %d into TPCD.ORDERS at %*.*s\n",

insertedOrders,UF_chunk,T_STAMP_1LEN,T_STAMP_1L
EN,  /* TIME_ACC jen*/

get_time_stamp(T_STAMP_FORM_1,(Timer_struct
*)NULL));  /* TIME_ACC jen*/
          /* report the number of row deleted *s inserted */
          fprintf(myoutstream,
                "    interim %ld rows for chunk %d into
TPCD.LINEITEM at %*.*s\n",
                insertedLineitem,UF_chunk,
                T_STAMP_1LEN,T_STAMP_1LEN,  /*
TIME_ACC jen*/
                get_time_stamp(T_STAMP_FORM_1,
                        (Timer_struct *)NULL));  /*
TIME_ACC jen*/

          fprintf( myoutstream,
                "    inserts for update pair %d chunk %d
complete at %*.*s\n\n",
                updatePair, UF_chunk,
                T_STAMP_1LEN,T_STAMP_1LEN,  /*
TIME_ACC jen*/
                get_time_stamp(T_STAMP_FORM_1,
                        (Timer_struct *)NULL));  /*
TIME_ACC jen*/

#endif
        }
      } /* process lineitem INSERTs */
    } /* while loop for deadlocks */
  } /* while processing chunks */

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

insertedLineitem,T_STAMP_1LEN,T_STAMP_1LEN,  /*
TIME_ACC jen*/
        get_time_stamp(T_STAMP_FORM_1,(Timer_struct
*)NULL));  /* TIME_ACC jen*/

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

    goto UF1_exit;
  }

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

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

  /* we used the first flat file to generate the semaphore key
*/
#ifndef SQLWINT
  /*    we will use the first flat file to generate the semaphore
key */
  if (getenv("TPCD_FLATFILES") != NULL) {
#ifndef TPCD_NONPARTITIONED
    /* this is assuming that you will be running this from 0th
node */
    sprintf(sourcefile, "%s%corder.tbl.u%d.00000.new",
          getenv("TPCD_FLATFILES"),
PATH_DELIM,updatePair);
#else
    sprintf(sourcefile, "%s%corder.tbl.u%d.0.new",
          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 semaphore UF1 set
%d\n",i);
    fprintf(stderr,"  semaphore sourcefile = %s  su_semid =
su_semid\n",sourcefile);
    exit(1);
  }                                                    /*jenSEM*/

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

    sleep(1);
  }

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



/*************************************************
*****************/
/* UF2 child                                                */
/*************************************************
*****************/
void runUF2_fn ( int updatePair, int thisConcurrentDelete,
int numChunks, char *dbname, char *userid, char *passwd )
{
  int rc = 0;
  long sqlcode;
  int maxwait;
  int startChunk = thisConcurrentDelete*numChunks; /*
where do we start? */
  long deletedLineitem = 0;  /*kmw*/
  long deletedOrders = 0;    /*kmw*/
  long savedDeletedOrders = 0; /*kbs*/

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

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

  strcpy(UF_dbname, dbname);
  strcpy(UF_userid, userid);
  strcpy(UF_passwd, passwd);

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

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

#ifdef UF2DEBUG
  fprintf (myoutstream, "RunUF2 Called %d   %d   %d\n",
                    updatePair, thisConcurrentDelete,
numChunks );
  fflush(myoutstream);
#endif
  fprintf( myoutstream,
        "\nUF2 for update pair %d set %d starting at
%*.*s\n",
        updatePair, thisConcurrentDelete,
T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
        get_time_stamp(T_STAMP_FORM_1,(Timer_struct
*)NULL));  /* TIME_ACC jen*/

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

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

#ifdef UF2DEBUG
  fprintf (myoutstream, "after connect  %d startchunk= %d,
EndChunk = %d\n",
                        startChunk, startChunk+numChunks);
  fflush(myoutstream);
#endif

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

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

#ifdef UF2DEBUG
  fprintf (myoutstream, "Chunk = %d\n", UF_chunk);
  fflush(myoutstream);
#endif
    while (sqlcode == SQL_RC_E911 && maxwait <=
MAXWAIT && rc == 0)
    {

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

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

      if (sqlcode == SQL_RC_E911)
      {       /* we've hit a deadlock */
        fprintf (myoutstream,
          "\nA deadlock detected while deleting from
ORDERS: update pair %d set %d chunk %d.  Retrying.\n",
          updatePair, thisConcurrentDelete, UF_chunk);
        dumpCa(&sqlca);
        SleepSome(UF_DEADLOCK_SLEEP);
        maxwait++;     /* jen DEADLOCK */
      }
      else if (sqlcode < 0)
      {
        fprintf (myoutstream,"\n%s\n", stmt_str.data);
        fprintf (myoutstream,"\nsqlcode %d occurred
deleting from TPCD.ORDERS\n", sqlca.sqlcode);
        dumpCa(&sqlca);
        fprintf (myoutstream,
              "for update pair number %d set %d chunk
%d..Exiting\n",
              updatePair, thisConcurrentDelete,UF_chunk);
        rc=-1;
      }
      else
      {
        /* accumulate the number of row deleted */
        savedDeletedOrders = sqlca.sqlerrd[2];   /*kbs*/

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

        /* delete the lineitems now */

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

        if (sqlcode == SQL_RC_E911)
        {       /* we've hit a deadlock */
#ifdef UF2DEBUG
  fprintf (myoutstream, "lineitem deadlocked\n");
  fflush(myoutstream);
#endif
          fprintf (myoutstream,
          "\nA deadlock detected while deleting from
LINEITEM: update pair %d set %d chunk %d.  Retrying.\n",
          updatePair, thisConcurrentDelete, UF_chunk);
          dumpCa(&sqlca);
          SleepSome(UF_DEADLOCK_SLEEP);
          maxwait++;     /* jen DEADLOCK */
        }
        else if (sqlcode < 0)
        {
#ifdef UF2DEBUG
  fprintf (myoutstream, "lineitem failed\n");
  fflush(myoutstream);
#endif
```

```
            fprintf (myoutstream,"\nAn error %d occurred
deleting from TPCD.LINEITEM\n",sqlca.sqlcode);
            dumpCa(&sqlca);
            fprintf (myoutstream,"for update pair number %d
set %d chunk %d..Exiting\n",
                    updatePair, thisConcurrentDelete,UF_chunk);
            rc=-1;
          }
         else
          {
#ifdef UF2DEBUG
  fprintf (myoutstream, "lineitem succeeded\n");
  fflush(myoutstream);
#endif
            /* accumulate the number of row deleted */
            /* Order count ONLY updated if both orders and
lineitem */
            /* go through */
            deletedOrders += savedDeletedOrders;   /* kbs */
            deletedLineitem += sqlca.sqlerrd[2];
            rc=0;
            EXEC SQL COMMIT WORK;
            error_check();
#ifdef UF2DEBUG
            /* report the number of rows deleted */
            fprintf(myoutstream, "    interim %ld rows for
chunk %d from TPCD.ORDERS at %*.*s\n",

deletedOrders,UF_chunk,T_STAMP_1LEN,T_STAMP_1LE
N,  /* TIME_ACC jen*/

get_time_stamp(T_STAMP_FORM_1,(Timer_struct
*)NULL));  /* TIME_ACC jen*/
            fprintf(myoutstream, "    interim %ld rows for
chunk %d from TPCD.LINEITEM at %*.*s\n",

deletedLineitem,UF_chunk,T_STAMP_1LEN,T_STAMP_1
LEN,  /* TIME_ACC jen*/

get_time_stamp(T_STAMP_FORM_1,(Timer_struct
*)NULL));  /* TIME_ACC jen*/
            fprintf( myoutstream,
                    "    deletes for update pair %d chunk %d
complete at %*.*s\n\n",
                    updatePair, UF_chunk,
                    T_STAMP_1LEN,T_STAMP_1LEN,  /*
TIME_ACC jen*/
                    get_time_stamp(T_STAMP_FORM_1,
                            (Timer_struct *)NULL));  /*
TIME_ACC jen*/
#endif
          }
        } /* process lineitem deletes */
      } /* while trying to delete one chunk loop */
    } /* while there are more chunks */

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

deletedLineitem,T_STAMP_1LEN,T_STAMP_1LEN,  /*
TIME_ACC jen*/
        get_time_stamp(T_STAMP_FORM_1,(Timer_struct
*)NULL));  /* TIME_ACC jen*/

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

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

UF2_exit:
  fclose (myoutstream);

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

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

```
    fprintf(stderr,"Retry Open semaphore %s\n",
UF2_semfile);

    SleepSome(1);
  }

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

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

E 3: runpower

```
:   # -*-Perl-*-
eval 'exec perl5 -S $0 ${1+"$@"}' # Horrible kludge to
convert this
   if 0;                     # into a "portable" perl script

# usage  runpower [UF]
# where UF is the optional parameter that says to run the
power test
# with the update functions.  By default, the update functions
are not
# run

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

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

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

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

if (@ARGV > 0)
{
   $runUF=$ARGV[0];
}
else
{
   $runUF="no";
}

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

#set up local variables
$runNum=$ENV{"TPCD_RUNNUMBER"};
$runDir=$ENV{"TPCD_RUN_DIR"};
$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$dbname=$ENV{"TPCD_DBNAME"};
$sf=$ENV{"TPCD_SF"};
$platform=$ENV{"TPCD_PLATFORM"};
$delim=$ENV{"TPCD_PATH_DELIM"};
$gatherstats=$ENV{"TPCD_GATHER_STATS"};
$product=$ENV{"TPCD_PRODUCT"};
$RealAudit=$ENV{"TPCD_AUDIT"};
$inlistmax=$ENV{"TPCD_INLISTMAX"};
$pn=$ENV{"TPCD_PHYS_NODE"};

if ($inlistmax eq "default")
{
 $inlistmax = 400;
}

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

# append isolation level information about tpcdbatch to the
miso file
# the miso file is created here but appended to for power and
throughput
#information

$misofile="$runDir${delim}miso$runNum";
if ( -e $misofile )
{
   &rm("$misofile");
}
```

```
# if we are in real audit mode then we must start the db
manager now since
# there must be no activity on the database between the time
the build script
# has finished and the time the power test is started
if ( $RealAudit eq "yes" )
{
   system("db2start");
}

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

open(MISO, ">$misofile") || die "Can't open $misofile: $!\n";
$curTs = `perl gettimestamp "long"`;
print MISO "Timestamp and isolation level of tpcdbatch
before power run at : $curTs\n";
close(MISO);
if ( $product eq "pe" )
{
   system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_id,isolation from sysibm.sysplan
where name='TPCDBATC'\"; db2 connect reset; db2
terminate  >> $runDir${delim}miso$runNum ");
}
else
{
   &verifyTPCDbatch("$misofile","$dbname");
}

if ($platform eq "aix")
{

  # Create the sysunused file.  This reports what disks are
attached, and which
  # ones are being used.  Its use spans both the runpower and
runthroughtput tests
  system("echo \"The following disks are assigned to the
indicated volume groups\" > $runDir/sysunused$runNum")
&& die "cannot create $runDir/sysunused$runNum";

  system("lspv >> $runDir/sysunused$runNum");
  system("echo \"The following volume groups are currently
online\" >> $runDir/sysunused$runNum");
  $curTs = `perl gettimestamp "long"`;
  system("echo \"$curTs\" >> $runDir/sysunused$runNum");
  system("lsvg -o >> $runDir/sysunused$runNum");
  # show the disks that are used/unused
  system("getdisks \"Before the start of the Power Test\"");

}
else
{
  # for all other platforms
  system("echo Assume that all portions of the system are
used >> $runDir${delim}sysunused$runNum");
}

&getConfig("p");
if ($gatherstats eq "on")
{
   # gather vm io and net stats
   if ($platform eq "aix")
   {
      # gather vmstats and iostats (and net stats if in mpp
mode)
      system("perl getstats p &");
   }
   else
   {
      print "Stats gather not set up for current platform
$platform\n";
   }
}
if ( $runUF ne "UF" )
{
  print "Beginning power stream....no update functions\n";
  # run the 17 queries for the powertest (stream = 0) with NO
update functions

$ret=system("$auditDir${delim}auditruns${delim}tpcdbatch
-d $dbname -f $runDir${delim}qtextpow.sql -r on -b on -s
$sf -u p -m $inlistmax -n 0 -l
$auditDir${delim}auditruns${delim}querytext${delim}strea
mpow.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}strea
m0.list");
}
if ($ret == 0)
{
   print "Power stream completed succesfully.\n";
}
else
{
   print "Power stream failed. ret=$ret\n";
}

if ($platform eq "aix")
{
   # show that the same disks are still used or unused
   system("getdisks \"After completion of the Power Test\"");

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


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

```
close(MISO);

if ( $product eq "pe" )
{
   system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_id,isolation from sysibm.sysplan
where name='TPCDBATC'\";db2 connect reset;db2
terminate  >> $runDir${delim}miso$runNum");
}
else
{
   &verifyTPCDbatch("$misofile","$dbname");
}
if ( $RealAudit ne "yes" )
{
   $curTs = `perl gettimestamp "short"`;
   # grab the db and dbm snapshot before we deactivate
   system("db2 get snapshot for all on $dbname >
$runDir${delim}dbrun$runNum.snap.$curTs");
   system("db2 get snapshot for database manager >>
$runDir${delim}dbrun$runNum.snap.$curTs");
}

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

if ( $RealAudit eq "yes" )
{
   system("db2stop");
}

1;

sub getConfig
{
  $testtype=$_[0];
  print "Getting database configuration.\n";

$dbtunefile="$runDir${delim}m${testtype}dbtune${runNu
m}";
  open(DBTUNE, ">$dbtunefile") || die "Can't open
$dbtunefile: $!\n";
  $timestamp=`perl gettimestamp "long"`;
  print DBTUNE "Database and Database manager
configuration taken at : $timestamp";
  close(DBTUNE);
  system("db2 get database configuration for $dbname >>
$dbtunefile");
  system("db2 get database manager configuration >>
$dbtunefile");
  system("db2set >> $dbtunefile");

}

sub verifyTPCDbatch
{
  $logfile=$_[0];
  $dbname=$_[1];
  $file="verifytpcdbatch.clp";
  open(VERTBL, ">$file") || die "Can't open $file: $!\n";
  print VERTBL "connect to $dbname;\n";
  print VERTBL "select
name,creator,valid,last_bind_time,isolation from
sysibm.sysplan where name='TPCDBATC';\n";
  print VERTBL "connect reset;\n";
  print VERTBL "terminate;\n";
  close(VERTBL);
  system("db2 -vtf $file >> $logfile");
}
```

Appendix F: ACID Test Source Code

F 1: acid.sqc

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

/*   changes:
 *
 *  961109  jel    add EXEC SQL CLOSE for each cursor in
acidT
 *              to avoid bug in db2pe v1r2
 *  980225  gav    port to NT
 *  981103  kal    added ast_acidQ  for isolation test 7
 *  981103  kal    changed ast query to be the same as that
used in
 *              consistency tests.  Fixed so the long lEprice is
 *              cast to a double.  Changed so uses 3 decimal
points of
 *              precision.
 *  981218  sas             Moved acidT function to acidT.sqc,
so the updates and
 *                          and queries can be bound under
different isolation level,
 *                          if necessary.
 *
 */

#include "acid.h"



void sqlerror(char * , struct sqlca *);

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
char dbname[8];   /* = "tpcd"; */
EXEC SQL END DECLARE SECTION;

#ifdef SQLWINT

 /*
 ** redefine gettimeofday so I don't have to
 ** change too much aix-specific code
 */
/* typedef struct timeval { unsigned tv_sec; unsigned
tv_usec; }; */

struct timeb timer;

 void gettimeofday( struct timeval *tv, struct timezone *tz)
 {
   ftime(&timer);
   tv->tv_sec = timer.time;
   tv->tv_usec = timer.millitm  * 1000;
   tz->dummy = 0;
 }
#endif

/*---------------------------------------------------------------*/
/*       acidQ                                                   */
/*---------------------------------------------------------------*/
int acidQ (struct acidQ_struct *acid)
{
  time_t timeT;
  FILE *out;
  char out_fn[50];
  struct timeval tv;
  struct timezone tz;
  int mypid;
  int rc = 0;


  EXEC SQL BEGIN DECLARE SECTION;
  long    okey;
  long    lEprice;
  double  eprice;
  EXEC SQL END DECLARE SECTION;

  okey = acid->o_key;

  /* mypid = getpid(); */
  mypid = acid->tag;

  sprintf(out_fn,
"%s%cacidQ.out.%d",getenv("TPCD_TMP_DIR"),del(),myp
id);
  out=fopen(out_fn,"a");
  if (out == NULL)
  {
    fprintf(stderr, "ERROR input file %s could not be
appended to!!\n",out_fn);
   }

  gettimeofday(&tv, &tz);
  time(&timeT);
  fprintf(out,"\n---------- START of acidQ tag: %d ----------
\n\n",mypid);
  fprintf(out, "acidQ tag: %d, begin transaction time: (%us
%06uu) %s",
         mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
  fprintf(out, "okey: %d\n", okey);

  gettimeofday(&tv, &tz);
  time(&timeT);
  fprintf(out,"acidQ tag: %d, before read of LINEITEM:
(%us %06uu) %s",
         mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));

  /*
  ** use the same sql code as used in the consistsql.pl to
  ** run the consistency acid queries.  Note we assign an
long int
  ** to lEprice (we make it 10s of pennies by * 1000).  Then
divide
  ** by 1000.0 and cast it to a double (eprice) for printing
  */

  EXEC SQL
    SELECT

INTEGER(DECIMAL(SUM(DECIMAL(INTEGER(INTEG
ER(DECIMAL

(INTEGER(100*DECIMAL(L_EXTENDEDPRICE,20,3)),
20,3) *
        (1-L_DISCOUNT)) * (1+L_TAX)),20,3)/100.0),20,3)
* 1000)
      into :lEprice
    FROM
      TPCD.LINEITEM
```

```
    WHERE
      L_ORDERKEY = :okey;

  if (sqlca.sqlcode != 0) {
    rc = sqlca.sqlcode;
    fprintf(out,"acidQ **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    sqlerror("acidQ: select sum(l_extendedprice)", &sqlca);
    goto Qerror;
  }
  eprice = (double)lEprice / 1000.0;  /* translate to double
for printout*/

  gettimeofday(&tv, &tz);
  time(&timeT);
  fprintf(out,"ACID tag: %d, after read of LINEITEM: (%us
%06uu) %s",
        mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
  fprintf(out, "okey: %d \t sum(l_extendedprice): %0.3f\n",
        okey, eprice);

  EXEC SQL COMMIT;
  if (sqlca.sqlcode != 0) {
    rc = sqlca.sqlcode;
    fprintf(out,"acidQ **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    sqlerror("acidQ: COMMIT", &sqlca);
    goto Qerror;
  }
  acid->l_extendedprice = eprice;

  rc = 0;
  goto Qexit;

 Qerror:
  EXEC SQL rollback work;
  if (sqlca.sqlcode != 0) sqlerror("acidQ: ROLLBACK
FAILED", &sqlca);

 Qexit:
  fprintf(out,"\n---------- END of acidQ tag: %d ----------
\n\n",mypid);
  fflush(out);fclose(out);
  return(rc);
}

/*-----------------------------------------------------------------*/
/*       ast_acidQ                                                 */
/*-----------------------------------------------------------------*/
int ast_acidQ (struct acidQ_struct *acid)
{
  time_t timeT;
  FILE *out;
  char out_fn[50];
  struct timeval tv;
  struct timezone tz;
  int mypid;
  int rc = 0;


  EXEC SQL BEGIN DECLARE SECTION;
  double    ast_lEprice;
  double    ast_eprice;
  EXEC SQL END DECLARE SECTION;


  /* mypid = getpid(); */
  mypid = acid->tag;

  sprintf(out_fn,
"%s%cast_acidQ.out.%d",getenv("TPCD_TMP_DIR"),del(),
mypid);
  out=fopen(out_fn,"a");
  gettimeofday(&tv, &tz);
  time(&timeT);
  fprintf(out,"\n---------- START of ast_acidQ tag: %d --------
--\n\n",mypid);
  fprintf(out, "ast_acidQ tag: %d, begin transaction time:
(%us %06uu) %s",
        mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));

  gettimeofday(&tv, &tz);
  time(&timeT);
  fprintf(out,"ast_acidQ tag: %d, before read of LINEITEM:
(%us %06uu) %s",
        mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));

  /*
  ** use the same query acidQ except don't select for specfic
okey.
  ** this ensures that the ast will be used instead of the base
table
  ** Have to use ast_lEprice as double since this sum is so
big
  */
  EXEC SQL
    SELECT
      SUM ( L_EXTENDEDPRICE*(1-L_DISCOUNT)*(1 +
L_TAX))
      into :ast_lEprice
    FROM
      TPCD.LINEITEM;

  if (sqlca.sqlcode != 0) {
    rc = sqlca.sqlcode;
    fprintf(out,"ast_acidQ **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    sqlerror("ast_acidQ: select sum(l_extendedprice)",
&sqlca);
    goto Qerror;
  }
  ast_eprice = ast_lEprice;  /* use ast_eprice  for printout to
be consistent*/

  gettimeofday(&tv, &tz);
  time(&timeT);
  fprintf(out,"AST_ACID tag: %d, after read of LINEITEM:
(%us %06uu) %s",
        mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
  fprintf(out, "sum(l_extendedprice): %0.3f\n",
        ast_eprice);

  EXEC SQL COMMIT;
  if (sqlca.sqlcode != 0) {
    rc = sqlca.sqlcode;
    fprintf(out,"ast_acidQ **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    sqlerror("ast_acidQ: COMMIT", &sqlca);
    goto Qerror;
  }
  acid->l_extendedprice = ast_eprice;

  rc = 0;
  goto Qexit;

 Qerror:
```

```
  EXEC SQL rollback work;
  if (sqlca.sqlcode != 0) sqlerror("ast_acidQ: ROLLBACK
FAILED", &sqlca);

 Qexit:
  fprintf(out,"\n---------- END of ast_acidQ tag: %d ----------
\n\n",mypid);
  fflush(out);fclose(out);
  return(rc);
}




/*-----------------------------------------------------------------*/
/*       connect_to_TM                                             */
/*-----------------------------------------------------------------*/
void connect_to_TM( void )
{
  char *dbname_ptr;
  if ((dbname_ptr = getenv("TPCD_QUAL_DBNAME")) !=
NULL) {
     fprintf(stderr,"********** %s
**********\n",dbname_ptr);
     strcpy (dbname, dbname_ptr);
  }

  EXEC SQL CONNECT TO :dbname IN SHARE MODE;
  if (sqlca.sqlcode <  0) {
     fprintf(stderr, "CONNECT TO %s failed SQLCODE =
%d\n", dbname, sqlca.sqlcode);
     exit(-1);
  }
  return;
}

/*-----------------------------------------------------------------*/
/*       disconnect_from_TM                                        */
/*-----------------------------------------------------------------*/
void disconnect_from_TM ( void )
{
  EXEC SQL CONNECT RESET;
  if (sqlca.sqlcode <  0) {
     fprintf(stderr, "DISCONNECT failed SQLCODE =
%d\n", sqlca.sqlcode);
     exit(-1);
  }
  return;
}

/*-----------------------------------------------------------------*/
/*       sqlerror                                                  */
/*-----------------------------------------------------------------*/
void sqlerror(char *msg, struct sqlca *psqlca)
{
  FILE *err_fp;


  char  err_fn[256];


  int j,k;

  sprintf(err_fn,
"%s%cacid.sqlerrors",getenv("TPCD_TMP_DIR"),del());
  err_fp=fopen(err_fn,"a");
  fprintf(err_fp,"acid: sqlcode: %4d %s\n", psqlca->sqlcode,
msg);
  fprintf(stderr,"acid: sqlcode: %4d %s\n", psqlca->sqlcode,
msg);
  fflush(stderr);
  if (psqlca->sqlerrmc[0] != ' ' || psqlca->sqlerrmc[1] != ' ') {
    fprintf(err_fp,"acid: slerrmc:  ");
    for(j = 0; j < 5; j++)
    {
      for(k = 0; k < 14; k++) fprintf(err_fp,"%x ", psqlca-
>sqlerrmc[j*10+k]);
      fprintf(err_fp,"    ");
      for(k = 0; k < 14; k++) fprintf(err_fp,"%c",  psqlca-
>sqlerrmc[j*10+k]);
      fprintf(err_fp,"\n");
      if (j < 4) fprintf(err_fp,"          ");
    }
  }

  fprintf(err_fp,"acid: sqlerrp:  ");
  for(j = 0; j < 8; j++)    fprintf(err_fp,"%c", psqlca-
>sqlerrp[j]);
  fprintf(err_fp,"\n");

  fprintf(err_fp,"acid: sqlerrd: ");
  for(j = 0; j < 6; j++)    fprintf(err_fp," %d", psqlca-
>sqlerrd[j]);
  fprintf(err_fp,"\n");

  if (psqlca->sqlwarn[0] != ' ') {
    fprintf(err_fp,"acid: sqlwarn: ");
    for(j = 0; j < 8; j++)    fprintf(err_fp,"%c ", psqlca-
>sqlwarn[j]);
    fprintf(err_fp,"\n");
  }

  fprintf(err_fp,"\n");
  fflush(err_fp);fclose(err_fp);
}

#ifdef SQLWINT
void sleep(int sec)
{
 Sleep(sec * 1000);
}
#endif


char del(void)
{
#ifdef SQLWINT
 return '\\';
#else
 return '/';
#endif
}
```

F 2: acidT.sqc

```
/************************************************
**********************/
/*     File: acidT.sqc                                       */
/************************************************
**********************/

/*  changes:
 *
 *  981218  sas  Created. Moved function acidT from
acid.sqc to this
```

```
 *                              file.
 *              Commented out updateQ funtion which is no
longer being
 *              used by the package.
 */

#include "acid.h"

#define DEADLOCK -911

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

extern void sqlerror(char * , struct sqlca *);

EXEC SQL INCLUDE SQLCA;

#ifdef SQLWINT
  extern void gettimeofday( struct timeval * , struct timezone
*);
#endif

/*---------------------------------------------------------------*/
/*       acidT                                                   */
/*---------------------------------------------------------------*/
int acidT (struct acidT_struct *acid)
{

  time_t timeT;
  FILE *out;
  char out_fn[50];
  struct timeval tv;
  struct timezone tz;
  int mypid;
  int rc = 0;

  EXEC SQL BEGIN DECLARE SECTION;
  long     o_key, l_key, delta;
  long     l_partkey, l_suppkey;
  double   l_quantity, l_tax, l_discount, l_extendedprice;
  double   o_totalprice;
  double   new_quantity, rprice, cost, new_extprice,
new_ototal, ototal;
  EXEC SQL END DECLARE SECTION;

  EXEC SQL DECLARE l_cursor CURSOR FOR
    SELECT l_partkey, l_suppkey, l_quantity,
    l_tax, l_discount,
    l_extendedprice
      FROM tpcd.lineitem
      WHERE l_orderkey = :o_key
      AND l_linenumber = :l_key
      FOR UPDATE OF l_extendedprice, l_quantity;

  EXEC SQL DECLARE o_cursor CURSOR FOR
    SELECT o_totalprice
      FROM tpcd.orders
      WHERE o_orderkey = :o_key
      FOR UPDATE OF o_totalprice;

  if (acid->termination < 0 || acid->termination > 3) acid-
>termination = 0;
  o_key = acid->o_key;
  l_key = acid->l_key;
  delta = acid->delta;

  if (acid->logging) {
    /* mypid = getpid(); */
    mypid = acid->tag;
    sprintf(out_fn,
"%s%cacidT.out.%d",getenv("TPCD_TMP_DIR"),del(),myp
id);
    out=fopen(out_fn,"a");
    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"\n---------- START of acidT tag: %d ----------
\n\n",mypid);
    fprintf(out, "acidT tag: %d, begin transaction time: (%us
%06uu) %s",
          mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    fprintf(out, "o_key: %d\tl_key: %d\tdelta: %d\n", o_key,
l_key, delta);
  }
#ifdef DEBUG
  printf("o_key: %d\tl_key: %d\tdelta: %d\n", o_key, l_key,
delta);
#endif

 retry_tran:

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

  EXEC SQL OPEN l_cursor;
  if (sqlca.sqlcode != 0) {
    if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
      fprintf(out,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    } else {
      fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: OPEN l_cursor", &sqlca);
    goto Terror;
  }

  EXEC SQL FETCH l_cursor INTO
    :l_partkey, :l_suppkey, :l_quantity, :l_tax,
    :l_discount, :l_extendedprice;
  if (sqlca.sqlcode != 0) {
    if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
      fprintf(out,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    } else {
      fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: FETCH l_cursor", &sqlca);
    goto Terror;
  }

#ifdef DEBUG
  printf("l_quantity = %0.3f\n",l_quantity);
  printf("l_tax = %0.3f  \n",l_tax);
  printf("l_discount = %0.3f \n",l_discount);
  printf("l_extendedprice = %0.3f \n", l_extendedprice);
#endif
```

```
  if (acid->logging) {
    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"acidT tag: %d, after read of LINEITEM:
(%us %06uu) %s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    fprintf(out, "l_partkey: %d  l_suppkey: %d  l_quantity:
%0.3f\nl_tax: %0.3f  l_discount: %0.3f  l_extendedprice:
%0.3f\n",
            l_partkey, l_suppkey, l_quantity, l_tax, l_discount,
l_extendedprice);
  }

  rprice = TRUNC2( l_extendedprice/l_quantity );
  cost = TRUNC2( rprice * delta );
  new_extprice = l_extendedprice + cost;
  new_quantity = l_quantity + delta;

#ifdef DEBUG
  printf("rprice = %0.3f\n", rprice );
  printf("cost = %0.3f\n", cost );
  printf("new_extprice = %0.3f\n", new_extprice );
  printf("new_quantity = %0.3f\n", new_quantity );
#endif

  EXEC SQL UPDATE tpcd.lineitem
    SET l_extendedprice = :new_extprice,
       l_quantity = :new_quantity
    WHERE CURRENT OF l_cursor;

  if (sqlca.sqlcode != 0) {
    if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
      fprintf(out,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    } else {
      fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: UPDATE l_cursor", &sqlca);
    goto Terror;
  }

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

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

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

  EXEC SQL OPEN o_cursor;
  if (sqlca.sqlcode != 0) {
    if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
      fprintf(out,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    } else {
      fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: OPEN o_cursor", &sqlca);
    goto Terror;
  }

  EXEC SQL FETCH o_cursor INTO :o_totalprice;
  if (sqlca.sqlcode != 0) {
    if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging)
    {
      fprintf(out,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    }
    else
    {
      fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    }
    sqlerror("acidT: FETCH o_cursor", &sqlca);
    goto Terror;
  }

#ifdef DEBUG
  printf("o_totalprice = %0.3f\n",o_totalprice);
#endif

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

  ototal = o_totalprice -
             TRUNC2( TRUNC2( l_extendedprice * (1-
l_discount) ) * (1+l_tax) );
  new_ototal = TRUNC2( new_extprice * (1.0-l_discount) );
  new_ototal = TRUNC2( new_ototal * (1.0+l_tax) );
  new_ototal = ototal + new_ototal;
```

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

  EXEC SQL UPDATE tpcd.orders
    SET o_totalprice = :new_ototal
    WHERE CURRENT OF o_cursor;
  if (sqlca.sqlcode != 0) {
    if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
      fprintf(out,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    } else {
      fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: UPDATE o_cursor", &sqlca);
    goto Terror;
  }

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

/*
** why is this code in here? we don't want to
** commit until the history table has been updated as well
 if (acid->termination == 0) {
    EXEC SQL CLOSE L_CURSOR;
    EXEC SQL CLOSE O_CURSOR;
    EXEC SQL COMMIT;
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
      }
      sqlerror("acidT: COMMIT", &sqlca);
      goto Terror;
    }
  }
*/

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

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

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

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

  switch (acid->termination) {
   case 1:
    {
     if (acid->logging)
     {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, wait before COMMIT:
(%us %06uu) %s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
     }
    }
    sleep(60);
   case 0:
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, immediately before
COMMIT: (%us %06uu) %s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    EXEC SQL CLOSE L_CURSOR;
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: CLOSE L_CURSOR", &sqlca);
      goto Terror;
    }
    EXEC SQL CLOSE O_CURSOR;
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode =
```

```
%d\n",sqlca.sqlcode);
        } else {
          fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
        } /* endif */
        sqlerror("acidT: CLOSE O_CURSOR", &sqlca);
        goto Terror;
      }
      EXEC SQL COMMIT;
      if (sqlca.sqlcode != 0) {
        if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
        rc = sqlca.sqlcode;
        if (acid->logging) {
          fprintf(out,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
        } else {
          fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
        } /* endif */
        sqlerror("acidT: COMMIT", &sqlca);
        goto Terror;
      }
      if (acid->logging) {
        gettimeofday(&tv, &tz);
        time(&timeT);
        fprintf(out,"acidT tag: %d, after COMMIT: (%us
%06uu) %s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
      }
      break;
    case 3:
      if (acid->logging) {
        gettimeofday(&tv, &tz);
        time(&timeT);
        fprintf(out,"acidT tag: %d, wait before ROLLBACK:
(%us %06uu) %s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
      }
      sleep(60);
    case 2:
      if (acid->logging) {
        gettimeofday(&tv, &tz);
        time(&timeT);
        fprintf(out,"acidT tag: %d, immediately before
ROLLBACK: (%us %06uu) %s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
      }
      EXEC SQL CLOSE L_CURSOR;
      EXEC SQL CLOSE O_CURSOR;
      EXEC SQL rollback work;
      if (sqlca.sqlcode != 0) {
        if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
        rc = sqlca.sqlcode;
        if (acid->logging) {
          fprintf(out,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
        } else {
          fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
        } /* endif */
        sqlerror("acidT: ROLLBACK", &sqlca);
        goto Terror;
      }
      if (acid->logging) {
        gettimeofday(&tv, &tz);
        time(&timeT);
        fprintf(out,"acidT tag: %d, after ROLLBACK: (%us
%06uu) %s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
      }
      break;
    }

    acid->l_partkey = l_partkey;
    acid->l_suppkey = l_suppkey;
    acid->l_quantity = l_quantity;
    acid->l_tax = l_tax;
    acid->l_discount = l_discount;
    acid->l_extendedprice = l_extendedprice;
    acid->o_totalprice = o_totalprice;

    rc = 0;
    goto Texit;

   Terror:
      EXEC SQL CLOSE L_CURSOR;
      EXEC SQL CLOSE O_CURSOR;
    EXEC SQL rollback work;
    if (sqlca.sqlcode != 0) sqlerror("acidT: ROLLBACK
FAILED", &sqlca);

   Texit:
    if (acid->logging) {
      fprintf(out,"\n---------- END of acidT tag: %d ----------
\n\n",mypid);
      fflush(out);fclose(out);
    }
    return(rc);
}



/*---------------------------------------------------------------*/
/*      updateQ                                                  */
/*---------------------------------------------------------------*/


int updateQ (struct update_struct *us)
{
    FILE *out;
    time_t timeT;
    struct timeval tv;
    struct timezone tz;
    int qnum;
    int rc = 0;
    int i;
    int secs2sleep;
    char buff[256];
    struct acidtype {int logging;} a, *acid;

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

    qnum = us->qnum;

    acid = &a;
    acid->logging= 1;

    sprintf(buff,
"%s%cupdate.out",getenv("TPCD_TMP_DIR"),del());
    out=fopen(buff,"a");
```

```
  gettimeofday(&tv, &tz);
  time(&timeT);
  fprintf(out,"\n---------- START of update ----------\n\n");
  fprintf(out, "update query number: %d, begin transaction
time: (%us %06uu) %s",
        qnum, tv.tv_sec, tv.tv_usec, ctime(&timeT));

  sqlca.sqlcode = 0;
  discount = 0.25;
  price = 5000.50;
  acctbal = 1000.00;
  availqty = 10;
  size = 5;

  for (i=1; i <= 2; i++) {
    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"update query number: %d, pass %d,
immediately before UPDATE: (%us %06uu) %s",
          qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));

    switch (qnum)
    {
     case 1:
     {
       EXEC SQL
         UPDATE TPCD.LINEITEM set L_DISCOUNT =
L_DISCOUNT + :discount
         WHERE L_ORDERKEY IN (326,512,928,995);
       if (sqlca.sqlcode != 0) {
         rc = sqlca.sqlcode;
         if (acid->logging)
         {
          fprintf(out,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
         }
         else
         {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
         }
         sqlerror("update query number 1", &sqlca);
         goto Uerror;
       }
       discount = discount * (-1);
       secs2sleep = 300;
       break;
     }
     case 2:
     {
       EXEC SQL
         UPDATE TPCD.SUPPLIER set S_ACCTBAL =
S_ACCTBAL + :acctbal
         WHERE S_NAME in
('Supplier#000000647','Supplier#000000070','Supplier#0000
00802');
       if (sqlca.sqlcode != 0) {
         rc = sqlca.sqlcode;
         if (acid->logging)
         {
          fprintf(out,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
         }
         else
         {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
         }
         sqlerror("update query number 2", &sqlca);
         goto Uerror;
       }
       acctbal = acctbal * (-1);
       secs2sleep = 90;
       break;
     }
     case 3:
     {
       EXEC SQL
         UPDATE TPCD.LINEITEM set L_DISCOUNT =
L_DISCOUNT + :discount
         WHERE L_ORDERKEY IN (260930, 402497,
457859, 509889, 58117,
                    538311, 588421, 416167, 97830,
90276);
       if (sqlca.sqlcode != 0) {
         rc = sqlca.sqlcode;
         if (acid->logging)
         {
          fprintf(out,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
         }
         else
         {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
         }
         sqlerror("update query number 3", &sqlca);
         goto Uerror;
       }
       discount = discount * (-1);
       secs2sleep = 300;
       break;
     }
     case 4:
     {
       if ( i ==1 ) {
         EXEC SQL
           UPDATE TPCD.ORDERS set O_ORDERDATE =
O_ORDERDATE - 6 MONTHS
           WHERE O_ORDERKEY =  67461;
         /*  WHERE O_ORDERKEY IN
(22400,28515,34338,46596,67461,92644,98307);*/
       } else {
         EXEC SQL
           UPDATE TPCD.ORDERS set O_ORDERDATE =
O_ORDERDATE + 6 MONTHS
           WHERE O_ORDERKEY =  67461;
       }
       if (sqlca.sqlcode != 0) {
         rc = sqlca.sqlcode;
         if (acid->logging)
         {
          fprintf(out,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
         }
         else
         {
          fprintf(stderr,"update query number: %d, pass %d,
```

```
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 4", &sqlca);
        goto Uerror;
      }
      secs2sleep = 300;
      break;
    }
    case 5:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT =
L_DISCOUNT + :discount
        WHERE L_ORDERKEY IN
(70976,566279,152897,84226,232483);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
         fprintf(out,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
         fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 5", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 300;
      break;
    }
    case 6:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT =
L_DISCOUNT + :discount
        WHERE L_ORDERKEY in
(33,131,161,195,229,230,231,323,353,356);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
         fprintf(out,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
         fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 6", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 300;
      break;
    }
    case 7:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT =
L_DISCOUNT + :discount
        WHERE L_ORDERKEY IN
(562917,410659,16550,398401,157634,429920,45411);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
         fprintf(out,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
         fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 7", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 300;
      break;
    }
    case 8:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT =
L_DISCOUNT + :discount
        WHERE L_ORDERKEY IN
(129569,343591,270242,254983,98500,28963);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
         fprintf(out,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
         fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 8", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 300;
      break;
    }
    case 9:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT =
L_DISCOUNT + :discount
        WHERE L_ORDERKEY IN
(113509,232997,246691,379233,448162,32134);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
         fprintf(out,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
```

```
          else
          {
           fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                   qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 9", &sqlca);
          goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 300;
        break;
      }
      case 10:
      {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_DISCOUNT =
L_DISCOUNT + :discount
          WHERE L_ORDERKEY IN
(516487,245411,265799,253025,6914,562020);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
           fprintf(out,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                   qnum, i, sqlca.sqlcode);
          }
          else
          {
           fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                   qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 10", &sqlca);
          goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 300;
        break;
      }
      case 11:
      {
        EXEC SQL
          UPDATE TPCD.PARTSUPP set PS_AVAILQTY =
PS_AVAILQTY + :availqty
          WHERE PS_PARTKEY IN
(12098,5134,13334,17052,3452,12552,1084,5797);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
           fprintf(out,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                   qnum, i, sqlca.sqlcode);
          }
          else
          {
           fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                   qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 11", &sqlca);
          goto Uerror;
        }
        availqty = availqty * (-1);
        secs2sleep = 180;
        break;
      }
      case 12:
      {
        if ( i ==1 ) {
          EXEC SQL
            UPDATE TPCD.LINEITEM set
L_RECEIPTDATE = L_RECEIPTDATE - 3 YEARS
            WHERE L_ORDERKEY IN
(33,70,195,355,677,837,960,962,1028);
        } else {
          EXEC SQL
            UPDATE TPCD.LINEITEM set
L_RECEIPTDATE = L_RECEIPTDATE + 3 YEARS
            WHERE L_ORDERKEY IN
(33,70,195,355,677,837,960,962,1028);
        }
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
           fprintf(out,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                   qnum, i, sqlca.sqlcode);
          }
          else
          {
           fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                   qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 12", &sqlca);
          goto Uerror;
        }
        secs2sleep = 300;
        break;
      }
      case 13:
      {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_DISCOUNT =
L_DISCOUNT + :discount
          WHERE L_ORDERKEY IN
(263,9476,32355,34854,53445,56901);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
           fprintf(out,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                   qnum, i, sqlca.sqlcode);
          }
          else
          {
           fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                   qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 13", &sqlca);
          goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 90;
        break;
      }
      case 14:
      {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_DISCOUNT =
```

```
L_DISCOUNT + :discount
          WHERE L_ORDERKEY IN
(32,225,326,448,449,483,512);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
           fprintf(out,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          else
          {
           fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 14", &sqlca);
          goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 180;
        break;
      }
      case 15:
      {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_DISCOUNT =
L_DISCOUNT + :discount
          WHERE L_ORDERKEY IN (1,4,7,35,135,131300);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
           fprintf(out,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          else
          {
           fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 15", &sqlca);
          goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 180;
        break;
      }
      case 16:
      {
        EXEC SQL
          UPDATE TPCD.PART set P_SIZE = P_SIZE + :size
          WHERE P_PARTKEY IN (4,7,15,1313);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
           fprintf(out,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          else
          {
           fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 16", &sqlca);
          goto Uerror;
        }
        size = size * (-1);
        secs2sleep = 180;
        break;
      }
      case 17:
      {
        EXEC SQL
          UPDATE TPCD.LINEITEM set
L_EXTENDEDPRICE = L_EXTENDEDPRICE + :price
          WHERE L_ORDERKEY IN
(4065,110372,165061,265702,87138);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
           fprintf(out,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          else
          {
           fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 17", &sqlca);
          goto Uerror;
        }
        price = price * (-1);
        secs2sleep = 90;
        break;
      }
      default:
      {
        fprintf(out,"ERROR: Invalid query number specified
%d\n", qnum);
        rc = 1;
        goto Uexit;
      }
      }

      gettimeofday(&tv, &tz);
      time(&timeT);

      if (acid->logging)
        fprintf(out,"update query number: %d, pass %d, after
UPDATE: (%us %06uu) %s",
              qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));
      else
        fprintf(stderr,"update query number: %d, pass %d,
after UPDATE: (%us %06uu) %s",
              qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));

      if ( i == 2 ) {
        gettimeofday(&tv, &tz);
        time(&timeT);
        fprintf(out,"update query number: %d, pass %d,
sleeping for %d seconds: (%us %06uu) %s",
              qnum, i, secs2sleep, tv.tv_sec, tv.tv_usec,
ctime(&timeT));
        fflush(out);
        system("touch /tmp/tpcd/update.sync.sleep");
        sleep(secs2sleep);
```

```
    }

    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"update query number: %d, pass %d,
immediately before COMMIT: (%us %06uu) %s",
          qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));

    EXEC SQL COMMIT;
    if (sqlca.sqlcode != 0) {
      rc = sqlca.sqlcode;
      fprintf(out,"update pass %d, **ERROR** sqlcode =
%d\n", i, sqlca.sqlcode);
      sqlerror("update: COMMIT", &sqlca);
      goto Uerror;
    }
    gettimeofday(&tv, &tz);
    time(&timeT);
    if (acid->logging)
      fprintf(out,"update query number: %d, pass %d, after
COMMIT: (%us %06uu) %s",
            qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    else
      fprintf(stderr,"update query number: %d, pass %d,
after COMMIT: (%us %06uu) %s",
            qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));
  }

  rc = 0;
  goto Uexit;

 Uerror:
  EXEC SQL rollback work;
  if (sqlca.sqlcode != 0) sqlerror("update: ROLLBACK
FAILED", &sqlca);
  system("touch /tmp/tpcd/update.sync.sleep");

 Uexit:
  fprintf(out,"\n---------- END of update ----------\n\n");
  fflush(out);fclose(out);
  return(rc);
}
```

F 3: acid.h

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

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

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

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

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

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



struct update_struct {
  int     qnum;
};

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

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

/*
** in acid.sqc
*/

int updateQ (struct update_struct *us);

char del(void);

#ifdef SQLWINT
void sleep (int sec);
typedef struct timezone { int dummy; };
#endif
```

Appendix G: Database Contents

Database Connection Information

Database server = DB2/NT 5.2.0

SQL authorization ID = DB2ADMIN

Local database alias = TPCD

SELECT * FROM TPCD.REGION FETCH FIRST 10 ROWS ONLY

R_REGIONKEY R_NAME R_COMMENT

---

0 AFRICA xSx31zz31Cl1z4OAnmm05AjiOxC3AMMNOgC0kACgwn gg3glP7LLLywlQy7R

1 AMERICA kgyh3LSnC72k6zlAz0LP3k2L4QB1QL1O673OjO1SPj0ng Q7CO100SBgmgRQ4lgPCMk21A425iklyAR4yBRAwR4C m5miNw 4jl13mMnxw17B

2 ASIA NSg6xlMlA1lzm6mOR0Ajx nhRA77NgRxBwL1M6Py RjySB3RLwkyPkwMM2R1BQ xAzkOgkjmll0gAghinP5inmNmR76MlijMS3S2zxONR15

3 EUROPE zlSL7Qwg12hMBL5lhlz0M45QkjShwSyiO04MLOh7wn1A RLQPyPAyAiil576l1Li7AlnR1S RQ4SLny7B2Ryj5P66MLhn NxhwB4C3ig0SO

4 MIDDLE EAST RllxmhPLz3Cy2mNlg4QMBnNASM ACki MPki7Oi

5 record(s) selected.

SELECT * FROM TPCD.NATION FETCH FIRST 10 ROWS ONLY

N_NATIONKEY N_NAME N_REGIONKEY N_COMMENT

---

0 ALGERIA 0 2Cxhl7 L1iwk6hMh300izngN32CPwCikyLk6khMzSRA

5 ETHIOPIA 0 NS7n LSOP Oz5n1AlB2S02nN0lMh4SBxP iRhBO 047R26 2BlM

12 JAPAN 2 y PiRn6O4Rny0P6Q0NhM5N26AhghiClRQlP QNQx205B3im7zPBOSSm5MAl3BjMLiM3P

6 FRANCE 3 3mjmizl S 3L3k2hNNhNlP4w370xRxyN15wn

22 RUSSIA 3 63w6MRN 2yANOMC0hjAn4RC7Qy6jhln37 2RggAM3x5S3i1On57

10 IRAN 4 h532g43BgShyO50OgSB2hO6jxQn3Q1w4NmPL4hg0mmP OC5SSLg6miR7m6B317 lz4jQMwPRyRn1jiRPmi mk0 yj0C5M5xi

20 SAUDI ARABIA 4 kC7 nNNz3ON73Az03O S13CklRh47jlxw5 5k6LClx52LQCQS6P1B3k1xPgMnk3PmMy7 l1nlBBBwBk0x6MACxOiigLO5mm5mO7N

14 KENYA 0 yA0jNSxmS25NRiCy1B6yhSL4NLMROQ3kLyAPx43wil27 kN1QmPLmPw7ywh2kyj2iSBwSjx26zClx

15 MOROCCO 0 7mC2hnARnj21nwllAOhR2nwRmgz22AkP6xj3iQiPNB0Pj xBMzS3g 4y0L4Alx1gNCAP2kjQMBR2xwiiSNLgCMQx74RmhMy4 OONhPCNQQ 3y

16 MOZAMBIQUE 0 LB6SkSOmkznRLS4z5P2BOmC23RnA6h7mn0gPAAN7nk xy6jO0k3w3R32R6A1ASk0LMyngO36jCL1gl5wQw4AMP C

10 record(s) selected.

SELECT * FROM TPCD.PART FETCH FIRST 10 ROWS ONLY

P_PARTKEY P_NAME P_MFGR P_BRAND P_TYPE P_SIZE P_CONTAINER P_RETAILPRICE P_COMMENT

---

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 FETCH FIRST 10 ROWS ONLY

S_SUPPKEY S_NAME S_ADDRESS S_NATIONKEY S_PHONE S_ACCTBAL S_COMMENT

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

151 Supplier#000000151 ny7AkOBAjL3L4iwM7NRnOSxN4i1SQmxmB3iC2N1g 22 32-960-568-5148 +8.56172000000000E+003 MCy0lOkSOQ0N1nPOmx1h0jwjS25MCj2CNyw7OPMQjg AR01M3iRgjjgx 7gl02B6xy 40 g2M 6myBOBB041M57xQ2 NQCPkL

152 Supplier#000000152 2nC1NRM5wRz2LBiN127n yj6L5LMQ7i 24 34-659-493-1274 +3.84660000000000E+003 2khmM3LOhLi04yQ5mz14Qx42x4BO00

153 Supplier#000000153 6L47l7h7MP7zRw 9 19-346-843-5260 +8.50550000000000E+002 jNALLC4O71x1jmCjO7SgA32OL4P4NmkSjO5wnS

154 Supplier#000000154 mi453jk2m0PLPN7OPxj555jA 4 6 13 23-471-808-2661 +4.15567000000000E+003 C367mPiwlNl2hxOC7NQQm6xlSBB6nwChRRSCg2kN 2ARzLiO

155 Supplier#000000155 lOR2 A3RmnRB 5lSzhgy4QQOnizLM7w5 22 32-417-987-3690 +3.40143000000000E+003 5P6MxS0P jLl27l65751lmmCAgwnjxm7j1APn53LQMO5jjBM5L5n Mi1NPhLN1jP5SB472ROn372ywNOz

156 Supplier#000000156 3hyN02mRAn1h0 5 15-106-692-4998 +9.78074000000000E+003 RixOni2M 7lm2RwRhCO3gQCwkNABxg210llOxL4n51LgLLPwMRN 7jgA

157 Supplier#000000157 1RCS6Ng6A6wM1l6ChNi2mn56 1j04N24in7lN 3 13-776-259-5994 -9.63190000000000E+002 nANLgP335NRxPm03jLxRxnAwwOnnm5LhwjPhP1n7AL2 mwR0l7yM A0zg6L3Cji7MnizzlL7 NL2Bx B7 jzyi66AACh

158 Supplier#000000158 MxyQCzh0zN0j yk6hzS7n5O65l7gnik 7 17-873-902-6175 +1.59644000000000E+003 S4g3hlk3zxBgO1iO1Sw n0AwS17RR4imnknL2h NQkRg7NA20OO4z10

159 Supplier#000000159 OR iAz4mi77 4 14-606-224-3002 +3.26360000000000E+002 n0nj0BllzMNghLyB33N2PSlhiA4hSPQOkQ24w3h7j32Plim 2w0h3A0lA2hjLMnhM0iL3y0m1LPgl16C1 miPyxQQwhw2gi7Sml

160 Supplier#000000160 S74w05 QOhi32 k75R gz4xmP 4 14-471-505-8811 +5.06764000000000E+003 AjgR7mB3PLz42lBN03xzl5nSOSxyC4ALkN435RxOxQj6j

10 record(s) selected.

SELECT * FROM TPCD.PARTSUPP FETCH FIRST 10 ROWS ONLY

```
PS_PARTKEY  PS_SUPPKEY  PS_AVAILQTY
PS_SUPPLYCOST           PS_COMMENT

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

        420      225421        1074   +5.90900000000000E+002
xiz1zRxz75P xLg4Lx1CkPRjj4l7Bh5gRk6 7x32
xiPShmS5NBi5w3CBCBmx73y mCnl2MQB2j5
37imLkMNP34Nz43MwA

        421        422        8121   +4.02290000000000E+002
4x34yLRSglgBylAgm2iQ2gzBzx
SjwQCBP71mQzLjxRz5ARhzhRi1ORkjiNNk6h
iiLj6kOM27M gh360z7m

        421       75422        1190   +2.34570000000000E+002
lRi
mPk5wz5zCSR5g5wwz2kkRj3PO55OLS24B4l7BAOSPw4z
Az3C02CSm3j6g1NCyw1O6j7g MikOnk
PwzRhkxPC50QxCQP2g5gOj1Azl4ACkN7m5SSh5

        421      150422        6215   +5.39500000000000E+002
gm6L3PgO2M MLM zm5001LQm62x7m5Ag
LOk3Q1n1i5x2C6PxNR12Cz730O

        421      225422        9804   +6.68600000000000E+002
gxMRnhwwO11j623NOnLxi3yiShnh2Q3w5j0BySRP04Pzx
7zmy

        422        423        6976   +5.37530000000000E+002
OBPxnLPxjkC7nLkkN5Sk471mljh73nQLNS05Qn1l1h36M
COmL4N2n357k4MQ7njhRjwnOM2yQ2xQ0BgwzygjOzO4
ykywM060ni1y4QOPSmAO6Nil6A60xCNAn3mQlwLmj
gC7NSNR yz57giBLBQi

        422       75423        1054   +5.03650000000000E+002
kBQC 1j22OANiSPg45QNCnSNMMl53iAix51nMjxgkn
7gQ7NSO 6ihk3w32PAz1OwnO3m A
gxlR20lN043Slz2Mi4z
liN2M643mSwAwyjLCP3S5nxSQMLnOlkxS3LiQOzjPkRS
5Sz1gzngAmMPB6NkC y5

        422      150423        1741   +4.28150000000000E+002
lwhP0mBnmLiQxLN6SBMRiOzA2y6Bk2MQ6L7QL54nAi
gN0iMMP5kzQiCMm06Mh45wgmLNSAxl3nP4niOC4yiO2
B7zP34mjy1RPM

        422      225423        8386   +3.23230000000000E+002
3SwlR mARhgBMz5iMS34LnN0Rxm02yOOLnlMz6
05k37R2hN44hR2xlwggwMBi7B24wCi0

        423        424        9892   +1.25610000000000E+002
7Mxgx2jhxhCSLBLn0M44knNh SkPhwSOxMPg67PjmOn
6RgnSBBBN3zCOimAwgz5NgCR606lh1QRQ



  10 record(s) selected.



SELECT * FROM TPCD.CUSTOMER FETCH FIRST 10
ROWS ONLY

C_CUSTKEY   C_NAME                 C_ADDRESS
C_NATIONKEY C_PHONE        C_ACCTBAL
C_MKTSEGMENT C_COMMENT

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

        400 Customer#000000400        CkyMl1S0x 7
7mNwjhkOmgzyx50BMLQR                14 24-522-746-
1247   -9.84600000000000E+001 BUILDING
5ylkyC3Rnz2R3yzM1kLNnwjji y5k R4

        401 Customer#000000401        0jN7gN2ix
mA5RACLk2LABRC06MQk                 19 29-667-766-
5291   +4.14643000000000E+003 BUILDING
2Rym2L0hgBP56M71khnBkBj
xl1gn5wM2Lk4N0RAl0LwL1Lzmwx

        402 Customer#000000402        3662m k7kmzhQ
6 16-950-729-1638   +2.10667000000000E+003
AUTOMOBILE
CSzzhm5Rw1wh4gMg4PCjByR6li31z0mCg2h3C061NLxz6
whLSj5lQ67LCLm35S7jOSxCQiS7SQnQmyS16M3LkM26i
Bmn

        403 Customer#000000403
gxPkB6xk5jw41AOB2ChnNNBMyLBS                14 24-
753-433-1769   +6.69336000000000E+003 HOUSEHOLD
wMC3R6Lm2xgQx5yw4hgiC6xn3xCM3zChmg57y72036A
0xQCxz015wiM0lg7NxmN5LmjRxSAyNO66A1R

        404 Customer#000000404        NmQ1C0B1gO
xLS2NMil6hg0nNOkC3C1SO0SBA          22 32-840-785-
1776   +7.40873000000000E+003 BUILDING
53ASBwnzi2237OzNgxwOQAORg0mnPR3PnPi1zj6gMRC
Q3iNS 0nmAy52204gMOzQAg0lkw0mg5Sig
7wyCO1nzBQim2y0OMP13hw4Mx0hNgljhL

        405 Customer#000000405
CzmLMgzSwBmSmQ10OlPL1z 0MiNPmNRN3m
10 20-509-301-7901   +7.51914000000000E+003
MACHINERY    BiwNn5Cmjxnjk
N1nQlAimx0RCPxASm6iBlgxPiAQOlj21yxiR4B

        406 Customer#000000406        Pi06ymg3
S4Azzh7j61BRlLN3zxnQQMm3Q1CBN          9 19-426-
693-4043   +4.28694000000000E+003 FURNITURE    gm
mM5zMLz71lMR3P OxL3RQ73PMS1wQMNjP37NklP

        407 Customer#000000407        7SRk7
7OORRgkmlhSlMLhCP                   1 11-975-454-
8499   +9.53708000000000E+003 MACHINERY
1OwgOBz0ByzQ5y10
NLQhC3jQn0NhklRCLy3mxwOLhQ1CkxPzz16Am47LgQ
m4 nC2LlwyR6nOwA SyPmB

        408 Customer#000000408
j61C6QR5C1BlwOL6kM                  10 20-177-
807-5661   +6.82537000000000E+003 BUILDING    5y4B
Q6Q5zil0jxhLPPQ2PB1Ll5S5n0NmPLiPMzNRLz5Oj7gOL
xPmAjnLAL4LBxg2QlOl7ny0j7h6OMg4zPSBBLgQARiOh

        409 Customer#000000409
```

AANBCxi2CL7ynwP7lR3g24mAOSkCA 11
21-466-412-4731 +3.96986000000000E+003
FURNITURE
jkSP7lg544OPRS0hL3PA4jCNymLwOClA4AmN5gCmxA7
0Pwk1x36 xN1SgR LlAgPQ2jhCj2hC572x63jhL2k053N
lg4OSOjyBhgzR4Qnl

10 record(s) selected.

SELECT * FROM TPCD.ORDERS FETCH FIRST 10
ROWS ONLY

O_ORDERKEY O_CUSTKEY O_ORDERSTATUS
O_TOTALPRICE O_ORDERDATE
O_ORDERPRIORITY O_CLERK O_SHIPPRIORITY
O_COMMENT

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

1 2340049 O +2.06270270000000E+005
01/02/1996 5-LOW Clerk#000028517 0
A0xCm5ARNL mxjChn2kC64xA4L6zBg2O5jhg M42izyPO
QlymN1ky5kmSiSgBAQA

2 3699418 O +3.97439100000000E+004
12/01/1996 1-URGENT Clerk#000026375 0
5PRxL1nM7xhQNzP2hnjhy1zz ykhg4P2A MMg5Px3OCN
0B0iyCRgiC2

3 4103281 F +2.52918760000000E+005
10/14/1993 5-LOW Clerk#000028628 0
nm0kygQBnw7RS3AAA4k

4 1668661 O +7.11905000000000E+003
10/11/1995 5-LOW Clerk#000003702 0
CP42CySQlz64n3mCyjm17 4B0CL
L5772m4k2Ai4h1nPySwSmNyCl4jOAOx5y4Rjx36nhO1x2
x4Qw

5 3901709 F +1.70010350000000E+005
07/30/1994 5-LOW Clerk#000027744 0
3PNC7zMP534MSizgy34Bxj62l0C7n6PBk7

6 1830002 F +2.87606000000000E+003
02/21/1992 4-NOT SPECIFIED Clerk#000001740
0 1CN00NA0z75SwwCxMNB0MLNL

7 3827629 O +2.01161380000000E+005
01/10/1996 2-HIGH Clerk#000014089 0
gmiC6hj5L4 0ixCAQkmB6giC16l4L16g

32 3457553 O +2.58512050000000E+005
07/16/1995 2-HIGH Clerk#000018469 0
7ihNSz00NCxA31PPx6RM4ih
BPPlz417SLk3SRA1zx0nlikRgjkx

33 2583433 F +7.82025000000000E+004
10/27/1993 3-MEDIUM Clerk#000012258 0
jkACLh 0igMiy72n Sky0h0B6NB70j7Q

34 3744836 O +1.12299560000000E+005
07/21/1998 3-MEDIUM Clerk#000006684 0
05k 2x242klm jyA wB0CBzzQnz5P11nAml5AL5jC lg5

10 record(s) selected.

SELECT * FROM TPCD.LINEITEM FETCH FIRST 10
ROWS ONLY

L_ORDERKEY L_PARTKEY L_SUPPKEY
L_LINENUMBER L_QUANTITY
L_EXTENDEDPRICE L_DISCOUNT
L_TAX L_RETURNFLAG L_LINESTATUS
L_SHIPDATE L_COMMITDATE L_RECEIPTDATE
L_SHIPINSTRUCT L_SHIPMODE L_COMMENT

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

1702 3216191 291202 4
+2.00000000000000E+001 +2.21406000000000E+004
+4.00000000000000E-002 +5.00000000000000E-002 N
F 06/08/1995 07/12/1995 06/28/1995 COLLECT
COD MAIL 5
g06CRwyOLCQ6NjyhgCnRAA7ABhnhSwS4g5x437w

1702 510428 60431 5
+2.30000000000000E+001 +3.30832000000000E+004
+6.00000000000000E-002 +8.00000000000000E-002 N
O 07/04/1995 06/30/1995 08/03/1995 COLLECT
COD MAIL B4hPMPBA
Q3M0MjgAkm5k1652yMRk

1702 4759116 184162 6
+1.90000000000000E+001 +2.23227200000000E+004
+3.00000000000000E-002 +3.00000000000000E-002 N
O 06/24/1995 07/05/1995 07/19/1995 TAKE
BACK RETURN MAIL
OnjAL2kxzkNP4yPnC0xg1Lj

1703 449228 299231 1
+4.80000000000000E+001 +5.65056000000000E+004
+9.00000000000000E-002 +2.00000000000000E-002 R
F 04/30/1993 04/07/1993 05/08/1993 DELIVER
IN PERSON MAIL gRlCLk6L7Myhwwy3AMwj7
zkMCQQPAkh S41iMB

1703 2167109 292131 2
+4.00000000000000E+000 +4.70400000000000E+003
+5.00000000000000E-002 +1.00000000000000E-002 A
F 02/07/1993 04/02/1993 03/04/1993 COLLECT
COD AIR 5n1g
iBOlz3Om2Pj0CAS2xPimCOSQlA0177Q1k7

1703 2597342 47359 3
+1.50000000000000E+001 +2.15883000000000E+004

```
+1.00000000000000E-002   +2.00000000000000E-002 R
F           03/22/1993 04/27/1993   03/31/1993    COLLECT
COD             REG AIR    2BPiwx0nS1

      1728     3513857      213858           1
+1.00000000000000E+001   +1.87068000000000E+004
+8.00000000000000E-002   +3.00000000000000E-002 N
O           06/26/1996 07/15/1996   07/26/1996    NONE
FOB       lC0xM63A11LgjRmwxS1kR

      1728     5380110      205162           2
+1.90000000000000E+001   +2.26071500000000E+004
+7.00000000000000E-002   +1.00000000000000E-002 N
O           06/17/1996 08/06/1996   06/20/1996    COLLECT
COD             FOB        Bh1jyxAyw1l210m

      1728     3496796       46819           3
+4.20000000000000E+001   +7.52900400000000E+004
+2.00000000000000E-002   +6.00000000000000E-002 N
O           07/20/1996 07/12/1996   07/27/1996    NONE
TRUCK      Q0Nz5njwO B2BQn3SBMOM

      1728      636528      186533           4
+4.20000000000000E+001   +6.15085800000000E+004
+6.00000000000000E-002   +8.00000000000000E-002 N
O           07/07/1996 06/30/1996   07/27/1996    NONE
RAIL       1Lhiiyyy1kgxM


  10 record(s) selected.



connect reset

DB20000I  The SQL command completed successfully.
```

Appendix H: Price Quotations

FEB 15 '99 14:33 FROM IBM SWMFG SOLUTIONS TO 916505569385 PAGE.001/001

P.O. Box 100
Somers, NY 10589
914 766 1900

February 15, 1999

Mr. Dave Davenport
Vice President of Storage and Systems Architecture
Hitachi Data Systems
750 Central Expressway
Santa Clara, CA 95050

Via Facsimile: (650) 556-9385

Dear Mr. Davenport:

Below is a price quote for DB2 Universal Database Enterprise Edition for NT 5.2.0.

DB2 Universal Database Enterprise Edition for NT 5.2.0 is currently available. The specific level of DB2 Universal Database Enterprise Edition for NT 5.2.0 which was used for your 30GB TPC-D benchmark will be available on June 30, 1999.

The price of DB2 Universal Database Enterprise Edition for NT, including support for up to 51 users, and 24 by 7, 4 hour response time defect support for the life of the product, is as follows:

- $ 7,999 for the Base Program Package (Part Number 31L0112)
- $16,999 for the upgrade from 1 CPU to 4 CPU licence (Part Number 31L0130)

Therefore the total price is $24,998. A Passport Advantage discount of 13% ($3,250) applies, bringing the net price to $21,748. This quote is valid until Friday, April 16, 1999.

Paul Rivot

Paul Rivot
WW DB2 Marketing Manager

** TOTAL PAGE.001 **