
Hewlett Packard Company

**HP ProLiant DL760 X900
Using IBM DB2 UDB 7.2**

**TPC BenchmarkTMH
Full Disclosure Report**

**Second Edition
March 2003**

Second Edition – March, 2003

Hewlett Packard Company, the Sponsor of this benchmark test, believes that the information in this document is accurate as of the publication date. The information in this document is subject to change without notice. The Sponsor assumes no responsibility for any errors that 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 upon workload, specific application requirements, and system design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, the TPC Benchmark H should not be used as a substitute for a specific customer application benchmark when critical capacity planning and/or product evaluation decisions are contemplated.

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

© Hewlett Packard Company 2003.

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

Printed in the United States March, 2003

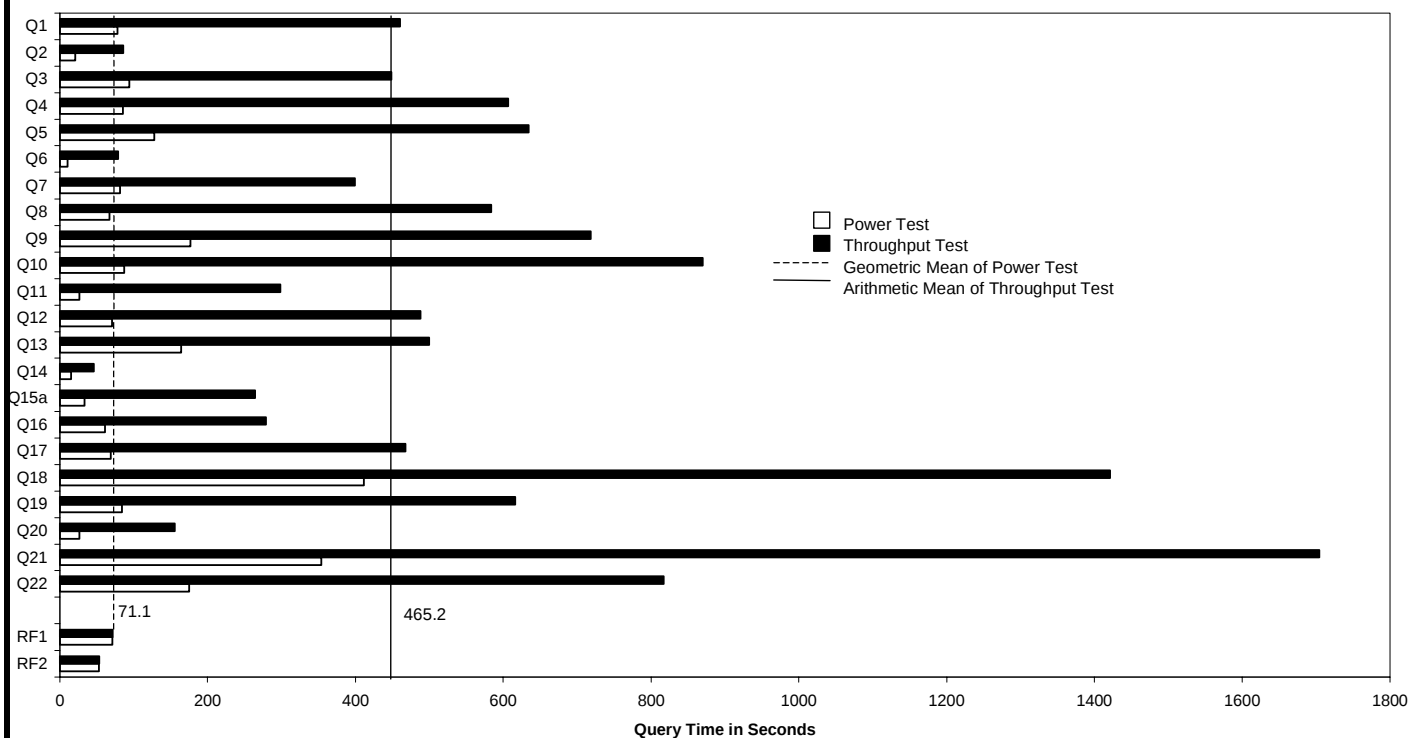
HP ProLiant is a registered trademark of Hewlett Packard Company.

IBM and DB2 UDB are registered trademarks of International Business Machines Corporation.

TPC Benchmark, TPC-H, QppH, QthH and QphH are trademarks of the Transaction Processing Performance Council.

All other brand or product names mentioned herein must be considered trademarks or registered trademarks of their respective owners.

Hewlett Packard Company		HP ProLiant DL760 X900-64P		TPC-H Rev. 2.0	
				Report Date: April 9, 2002	
Total System Cost		Composite Query per Hour Metric		Price / Performance	
\$2,636,750		12995.4 QphH@300GB		\$203 \$ / QphH@300GB	
Database Size	Database Manager	Operating System	Other Software	Availability Date	
300GB	IBM DB2 UDB 7.2	Microsoft Windows 2000 Advanced Server	Microsoft Visual C++ 6.0	June 20, 2002	



Database Load Time = 1:18:04	Load Included Backup: N	Total Data Storage / Database Size = 22.61
RAID (Base tables only): Y	RAID (Base tables and auxiliary data structures): Y	RAID (All): Y
System Configuration: 16 nodes		
Processors (per node) :	4 x 900MHz Intel Pentium III Xeon Processors w/ 2MB cache	
Memory (per node) :	4 GB memory	
Disks (per node) :	1 x 9GB 10K RPM drive 24 x 18GB 15K RPM drives	
Servers interconnect :	32 ports Gigabit Ethernet Switch	
Total Disk Storage:	6783.36 GB	

Hewlett Packard Company		ProLiant DL760 X900		TPC-H Rev. 2.0		
				Report Date:		9-Apr-02
Description	Part Number	Third Party	Unit Price	Qty	Extended Price	3 yr. Maint. Price
Server Hardware						
Brand Pricing						
Proliant DL760 X900-2M 4P 2G	212692-001	1	44,400	16	710,400	
- 4 Pentium III Xeon/900MHz 2MB Cache						
- 2 GB SD RAM Memory						
- CD-ROM 24X						
- 10/100 Ethernet Controller						
1G-Memory Kit SDRAM (2 x 512MB)	328808-B21	1	1,436	32	45,952	
NC6136 Gigabit Server Adapter, 64-bit/66MHz, PCI, 1000 SX	203539-B21	1	621	16	9,936	
\$5500 15" Monitor	261602-169	1	169	16	2,704	
Scroll Mouse Carbon	231947-B21	1	5	16	80	
PS/2 Easy Access Internet Keyboard	265977-001	1	12	16	192	
Rack Model 9142 (42U - Opal) - Flat Pallet	120663-B21	1	1,321	6	7,926	
Rack Coupling Kit	120669-B21	1	83	5	415	
Sidewall Panel Kit	120670-B21	1	207	1	207	
CarePaq 3YR 24X7 4HR 700 SERIES SVR (FM-HE724-36)	401784-002	1	3,390	16		54,24
Subtotal					777,812	54,24
Storage						
Compaq SMART Array Controller 5304 - 4SCSI channels	158939-B21	1	2,052	32	65,664	
StorageWorks Enclosure Model 4354R - Rack-mountable	190211-001	1	3,523	32	112,736	
18.2GB Pluggable Ultra3 SCSI 10K 1" Universal HDD	142673-B22	1	311	16	4,976	
18.2 GB Hot-Plug Wide U3 15K 1"	188122-B22	1	390	384	149,760	
18.2 GB Hot-Plug Wide U3 15K 1" (10% spare)	188122-B22	1	390	39	15,210	
CarePaq 3YR 24X7/4HR EMPTY DISK ENCL (FM-4E724-36)	171242-002	1	157	32		5,02
Subtotal					348,346	5,02
Hardware and Maintenance Discount						
Large Purchase and Net 30 discount	16.0%	1			(\$180,185)	(\$9,48
Hardware Subtotal					945,973	49,78
Software						
DB2 EE Edi. Lic. with 1Y support per processor	D5B5RLL IBM	3	22,153	64	1,417,792	include
DB2 EE Edi. Support per processor per year (Y2 and Y3)	E1B5ULL IBM	3	1,081	128		138,36
Microsoft Windows 2000 Advanced Server	C10-00475 Microsoft	2	2,399	16	38,384	include
Microsoft Visual Studio Professional 6.0 Win32	659-00390 Microsoft	2	1,079	1	1,079	include
Subtotal					1,457,255	138,36
Connectivity						
Extreme Summit 7i	Extreme	4	15,124	3	45,372	include
Connectivity Subtotal					45,372	
Total					\$2,448,600	\$188,15
Three-Year Cost of Ownership:						\$2,636,750
OphH Rating:						12995.4
\$ / OphH:						\$203

**Hewlett Packard
Company**

**HP
ProLiant DL760 X900-64P**

TPC-H Rev. 2.0

Report Date:
April 9, 2002

Numerical Quantities

Measurement Results:

Database Scale Factor	= 300
Total Data Storage / Database Size	= 22.61
Start of Database Load	= 4/3/2002 19:02:01
End of Database Load	= 4/3/2002 20:20:05
Database Load Time	= 1:18:04
Query Streams for Throughput Test	= 6
TPC-H Power	= 15372.8
TPC-H Throughput	= 10985.6
TPC-H Composite Query-per-Hour Metric (QphH@300GB)	= 12995.4
Total System Price Over 5 Years	= \$2,636,750
TPC-H Price/ Performance Metric (\$/QphH@300GB)	= \$203

Measurement Intervals:

Measurement Interval in Throughput Test (Ts)	= 12977 seconds
--	-----------------

Duration of Stream Execution:

Stream ID	Seed	Start Date	Start time	Stop Date	Stop Time	Duration
Stream00	403202005	4/3/2002	21:46:03	4/3/2002	22:26:47	0:40:44
Stream01	403202006	4/3/2002	22:26:53	4/4/2002	1:46:30	3:19:37
Stream02	403202007	4/3/2002	22:26:53	4/4/2002	1:47:35	3:20:42
Stream03	403202008	4/3/2002	22:26:53	4/4/2002	1:50:06	3:23:13
Stream04	403202009	4/3/2002	22:26:53	4/4/2002	1:42:20	3:15:27
Stream05	403202010	4/3/2002	22:26:56	4/4/2002	1:50:40	3:23:44
Stream06	403202011	4/3/2002	22:26:57	4/4/2002	1:38:28	3:11:31
Refresh		4/4/2002	1:50:40	4/4/2002	2:03:09	0:12:29

**Hewlett Packard
Company**

**HP
ProLiant DL760 X900-64P**

TPC-H Rev. 2.0

Report Date:
April 9, 2002

TPC-H Timing Intervals (in seconds)

Query	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Stream 00	78.3	20.9	94.3	85.4	127.6	10.7	81.5	67.4
Stream 01	365.7	149.3	467.5	389.9	714.5	33.8	361.5	418.8
Stream 02	631.1	80.3	517.6	625.6	634.8	91.4	276.4	652.6
Stream 03	405.1	92.5	139.9	716.3	632.2	71.5	564.6	680.3
Stream 04	649.1	57.3	438.1	571.9	665.6	83.8	404.3	404.6
Stream 05	390.0	58.0	625.3	653.7	536.0	123.8	372.0	732.7
Stream 06	322.2	79.0	501.8	683.1	621.9	69.4	415.1	611.9
Min Qi	322.2	57.3	139.9	389.9	536.0	33.8	276.4	404.6
Max Qi	649.1	149.3	625.3	716.3	714.5	123.8	564.6	732.7
Avg Qi	460.5	86.1	448.4	606.8	634.2	79.0	399.0	583.5
Query	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16
Stream 00	176.7	87.1	26.6	70.6	164.1	14.9	33.5	61.3
Stream 01	948.8	835.9	257.6	382.0	597.7	42.9	152.5	380.0
Stream 02	1079.7	734.1	209.6	335.0	503.7	42.0	349.8	171.8
Stream 03	506.9	927.3	385.8	430.5	401.2	54.4	316.2	302.1
Stream 04	450.2	959.2	542.8	696.0	467.0	40.8	318.3	239.4
Stream 05	493.4	846.6	149.7	611.6	518.2	51.0	275.5	321.9
Stream 06	831.4	916.6	244.7	474.8	510.7	45.6	171.7	258.2
Min Qi	450.2	734.1	149.7	335.0	401.2	40.8	152.5	171.8
Max Qi	1079.7	959.2	542.8	696.0	597.7	54.4	349.8	380.0
Avg Qi	718.4	870.0	298.4	488.3	499.8	46.1	264.0	278.9
Query	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream 00	69.2	411.5	84.1	26.2	353.9	174.7	71.0	52.9
Stream 01	340.4	1538.1	661.2	105.5	1953.0	880.3	66.0	54.1
Stream 02	622.5	1256.1	616.2	182.3	1335.2	1094.0	72.2	52.9
Stream 03	446.1	1545.0	485.4	139.7	1848.8	1101.3	72.8	53.9
Stream 04	402.5	1112.9	607.9	132.5	1592.4	890.0	72.9	53.6
Stream 05	468.8	1812.2	671.0	180.5	1948.0	384.1	71.7	53.3
Stream 06	525.4	1260.3	655.4	191.5	1547.5	553.2	73.0	52.7
Min Qi	340.4	1112.9	485.4	105.5	1335.2	384.1	66.0	52.7
Max Qi	622.5	1812.2	671.0	191.5	1953.0	1101.3	73.0	54.1
Avg Qi	467.6	1420.8	616.2	155.3	1704.2	817.2	71.4	53.4

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark™ H test conducted on the HP ProLiant DL760 X900 using IBM DB2 UDB 7.2, in conformance with the requirements of the TPC Benchmark™ H Standard Specification, Revision 2.0. The operating system used for the benchmark was Microsoft Windows 2000 Advanced Server with Service Pack 2. The application was written in Microsoft Visual C++ v6.0.

The benchmark results are summarized in the following table.

Hardware	Software	Total System Cost	QppH @ 300GB	QthH @ 300GB	QphH @ 300GB	\$ / QphH @ 300GB
HP ProLiant DL760 X900	IBM DB2 UDB 7.2 Microsoft Windows 2000 Advanced Server	\$2,636,75	15372.8	10985.6	12995.4	\$ 203

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

Copies of this full disclosure report can be obtained from either the Transaction Processing Performance Council or Hewlett Packard Company at the following address:

Transaction Processing Performance Council (TPC)
c/o Shanley Public Relations
777 North First Street, Suite 600
San Jose, CA 95112 USA
Telephone: (408) 295-8894
Fax: (408) 295-9768

or

Hewlett Packard Company
Mail Stop 150402
20555 SH-249
Houston, TX 77070-20555 USA

Standard and Executive Summary Statements

Pages ii-iv contains the Executive Summary and Numerical Quantities Summary of the benchmark results for the HP ProLiant DL760 X900.

Auditor

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

Table Of Contents

ABSTRACT	I
OVERVIEW	I
STANDARD AND EXECUTIVE SUMMARY STATEMENTS.....	I
AUDITOR	I
TABLE OF CONTENTS.....	II
1.0 GENERAL ITEMS.....	5
1.1 TEST SPONSOR.....	5
1.2 PARAMETER SETTINGS	5
1.3 CONFIGURATION ITEMS.....	5
2.0 CLAUSE 1: LOGICAL DATABASE DESIGN	7
2.1 TABLE DEFINITIONS	7
2.2 PHYSICAL ORGANIZATION OF DATABASE	7
2.3 HORIZONTAL PARTITIONING	7
2.4 REPLICATION	7
3.0 CLAUSE 2: QUERIES AND REFRESH FUNCTIONS RELATED ITEMS	8
3.1 QUERY LANGUAGE.....	8
3.2 RANDOM NUMBER GENERATION	8
3.3 SUBSTITUTION PARAMETERS GENERATION.....	8
3.4 QUERY TEXT AND OUTPUT DATA FROM DATABASE	8
3.5 QUERY SUBSTITUTION PARAMETERS AND SEEDS USED.....	8
3.6 ISOLATION LEVEL.....	9
3.7 REFRESH FUNCTIONS	9
4.0 CLAUSE 3: DATABASE SYSTEM PROPERTIES	10
4.1 ATOMICITY REQUIREMENTS	10
4.2 CONSISTENCY REQUIREMENTS	10
4.3 ISOLATION REQUIREMENTS.....	11
4.4 DURABILITY REQUIREMENTS.....	13
5.0 CLAUSE 4: SCALING AND DATABASE POPULATION.....	15
5.1 INITIAL CARDINALITY OF TABLES	15
5.2 DISTRIBUTION OF TABLES AND LOGS ACROSS MEDIA	15
5.3 MAPPING OF DATABASE PARTITIONS/REPLICATIONS.....	16
5.4 IMPLEMENTATION OF RAID.....	17
5.5 DBGEN MODIFICATIONS.....	17
5.6 DATABASE LOAD TIME.....	17
5.7 DATA STORAGE RATIO.....	17
5.8 DATABASE LOAD MECHANISM DETAILS AND ILLUSTRATION.....	17
6.0 CLAUSE 5: PERFORMANCE METRICS AND EXECUTION RULES RELATED ITEMS.....	19
6.1 STEPS IN THE POWER TEST.....	19
6.2 TIMING INTERVALS FOR EACH QUERY AND REFRESH FUNCTION	19
6.3 NUMBER OF STREAMS FOR THE THROUGHPUT TEST	19
6.4 START AND END DATE/TIMES FOR EACH QUERY STREAM	19
6.5 TOTAL ELAPSED TIME FOR THE MEASUREMENT INTERVAL	19
6.6 REFRESH FUNCTION START DATE/TIME AND FINISH DATE/TIME	19
6.7 TIMING INTERVALS FOR EACH QUERY AND EACH REFRESH FUNCTION FOR EACH STREAM	20
6.8 PERFORMANCE METRICS.....	20

6.9 THE PERFORMANCE METRIC AND NUMERICAL QUANTITIES FROM BOTH RUNS	20
6.11 SYSTEM ACTIVITY BETWEEN TESTS	20
7.0 CLAUSE 6: SUT AND DRIVER IMPLEMENTATION RELATED ITEMS	21
7.1 DRIVER	21
7.2 IMPLEMENTATION SPECIFIC LAYER (ISL)	21
7.3 PROFILE-DIRECTED OPTIMIZATION	21
8.0 CLAUSE 7: PRICING RELATED ITEMS	22
8.1 HARDWARE AND SOFTWARE USED	22
8.2 TOTAL 3 YEAR PRICE	22
8.3 AVAILABILITY DATE	22
8.4 COUNTRY-SPECIFIC PRICING	22
9.0 CLAUSE 9: RELATED ITEMS	23
9.1 AUDITORS' REPORT	23
APPENDIX A: TUNABLE PARAMETERS	26
A.1 DB2 DATABASE TPCD CONFIGURATION	26
A.2 DB2 DATABASE MANAGER CONFIGURATION	26
A.3 DB2 ENVIRONMENT VARIABLES	27
A.4 WINDOWS 2000 ENVIRONMENT VARIABLES	27
A.5 WINDOWS 2000 CONFIGURATION	28
A.6 HP SMART ARRAY CONTROLLER CACHE SETTINGS AND REGISTRY ENTRIES	28
A.7 COMPILER OPTIONS FOR TPCD BATCH DRIVER	28
A.8 COMPILER OPTIONS FOR ACID DRIVER	28
A.9 CPQ_SCATTERED-READ	28
APPENDIX B: DATABASE BUILD SCRIPTS	30
B.1 BUILDTPCD	30
B.2 TPCD.SETUP	39
B.3 CPQ.DBMCFG.LOAD	42
B.4 CPQ.DBCFG.LOAD	42
B.5 CPQ_DB2SET_BUILD.BAT	43
B.6 CPQ_CREATE_TABLESPACES	43
B.7 CPQ_CREATE_TABLES	43
B.8 LOAD	44
B.9 CPQ_TPCD_INDEXES	45
B.10 CPQ_RUNSTATS	46
B.11 CPQ.DBMCFG.RUN	46
B.12 CPQ.DBCFG.RUN	46
B.13 CPQ_DB2SET_RUN.BAT	47
B.14 CPQ_BUFFER_POOLS	47
B.15 CREATEUFTBLS	47
B.16 PREPDRVR.BAT	47
B.17 BUILD.C	48
B.18 DRIVER.SQC	52
B.19 DSS.H	60
B.20 DSSTYPES.H	64
APPENDIX C: QUERY TEXT AND OUTPUT	134
C.1 QUALIFICATION QUERIES AND OUTPUT	136
C.2 FIRST 10 ROWS OF TEST DATABASE TABLES	215
C.3 QUERY SUBSTITUTION PARAMETERS	218

APPENDIX D: IMPLEMENTATION SPECIFIC LAYER AND DRIVER SOURCE CODE.....	223
D.1 TPCDBATCH.H	223
D.2 TPCDBATCH.SQC	224
D.3 TPCDUF.SQC	252
D.4 RUNPOWER.....	258
D.5 RUNTHROUGHPUT	261
D.6 PLOADUF1	265
D.7 PLOADUF2	266
D.8 DOUFLOAD.BAT	267
D.9 UFLoad.BAT	267
D.10 LOAD_UF1_DATA.....	267
D.11 LOAD_UF2_DATA.....	268
APPENDIX E: ACID TRANSACTION SOURCE CODE.....	269
E.1 ACID.SQC	269
E.2 ACID.H	279
APPENDIX F: PRICE QUOTATIONS	281

1.0 General Items

1.1 Test Sponsor

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

HP is the sponsor of this TPC-H Benchmark.

1.2 Parameter Settings

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

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

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

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

1.3 Configuration Items

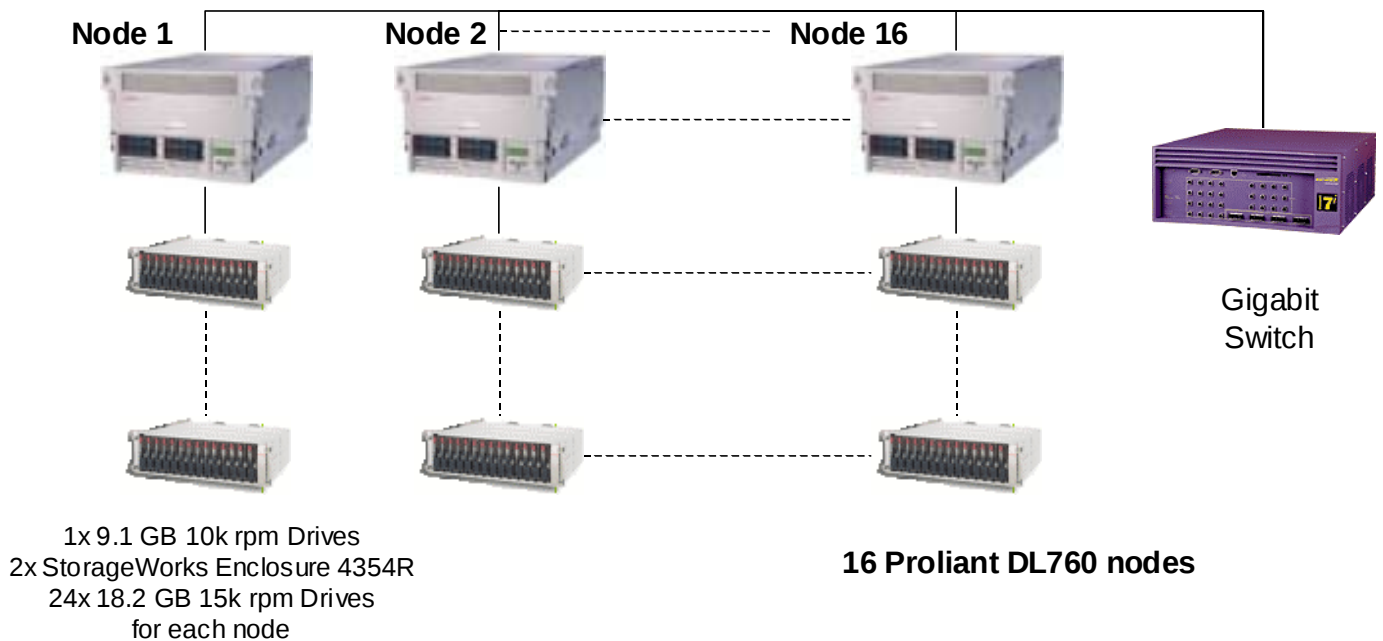
Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:

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

The server System Under Test (SUT), a 16 nodes cluster HP ProLiant DL760 X900, depicted in Figure 1.1, consisted of (for each node):

- Four 900 MHz Pentium III Xeon® Processors with 2M L2 cache
- 4096 MB of memory
- 2 x HP 5300 Array Controllers
- 2 x StorageWorks Enclosure 4354R
- 1 x 9.1GB Pluggable SCSI-3 10K rpm 1" height drives
- 24 x 18.2GB Pluggable SCSI-3 15K rpm 1" height drives
- One NC7770 Gigabit PCI Network Card

Figure 1.1 Benchmarked and priced configuration



As the 9.1GB 10K rpm disk drives used in the benchmarked configuration are no longer available they are substituted with 18.2 GB 10K rpm disk drives in the priced configuration.

2.0 Clause 1: Logical Database Design

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

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

2.1 Table Definitions

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

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

2.2 Physical Organization of Database

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

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

2.3 Horizontal Partitioning

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

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

2.4 Replication

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

No replication was used.

3.0 Clause 2: Queries and Refresh Functions Related Items

3.1 Query Language

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

SQL was the query language used.

3.2 Random Number Generation

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

The supplied QGEN version 1.3.0 was used.. DBGEN 1.3.0 was modified to allow the scale factor of 3000. See Appendix B “Database Build Scripts” for details.

3.3 Substitution Parameters Generation

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

The supplied QGEN version 1.3.0 was used to generate the substitution parameters.

3.4 Query Text and Output Data from Database

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

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

- Table names are fully qualified. For example, the “NATION” table is referred to as “TPCD.NATION.”
- The standard IBM SQL date syntax is used for the date arithmetic. For example, DATE(‘1996-01-01’) + 3 MONTHS.
- The semicolon (;) is used as a command delimiter.
- Count_big was used instead of count where aggregates exceed the range of values supported by an integer.

3.5 Query Substitution Parameters and Seeds Used

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

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

3.6 Isolation Level

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

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

3.7 Refresh Functions

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

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

4.0 Clause 3: Database System Properties

4.1 Atomicity Requirements

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

All ACID tests were conducted according to specification. The Atomicity, Isolation, Consistency and Durability tests were performed on the HP ProLiant DL760 X900. Appendix E, “ACID Transaction Source Code,” contains the source code for the ACID transaction and query.

4.1.1 Atomicity of the Completed Transactions

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

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

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

4.1.2 Atomicity of Aborted Transactions

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

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

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

4.2 Consistency Requirements

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

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

$O_TOTALPRICE = SUM(L_EXTENDEDPRICE - L_DISCOUNT) * (1 + L_TAX)$
For each ORDER and LINEITEM defined by $(O_ORDERKEY = L_ORDERKEY)$

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

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

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

4.2.1 Consistency Tests

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

The queries defined in section 3.3 “Consistency Condition” of TPC-H specification were run after the initial database build and prior to executing the ACID transaction. The queries showed that the database was in a consistent state.

After executing 7 streams of 100 ACID transactions, the queries defined in 3.3 “Consistency Condition” section were run again. The queries showed that the database was still in a consistent state.

4.3 Isolation Requirements

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

4.3.1 Isolation Test 1 - Read-Write Conflict with Commit

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

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

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

4.3.2 Isolation Test 2 - Read-Write Conflict with Rollback

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

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

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

4.3.3 Isolation Test 3 - Write-Write Conflict with Commit

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

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

1. First session: Start an ACID transaction T1 for a randomly selected O_KEY, L_KEY and DELTA. The transaction is delayed for 60 seconds just prior to the Commit.
2. Second session: Start a second ACID transaction T2 for the same O_KEY, L_KEY and for a randomly selected DELTA2. This transaction is forced to wait while the First Session holds a lock on the LINEITEM record requested by the Second Session.
3. First session: The ACID transaction T1 is released and the Commit is executed, releasing the record. With the LINEITEM record now released, the ACID transaction T2 can now complete.
4. Verify that: $T2.L_EXTENDEDPRICE = T1.EXTENDEDPRICE + (DELTA * (T1.L_EXTENDEDPRICE) / T1.L_QUANTITY)$

4.3.4 Isolation Test 4 - Write-Write Conflict with Rollback

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

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

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

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

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

The following steps were performed to successfully conduct this test:

1. First session: Start an ACID transaction T1 for a randomly selected O_KEY, L_KEY and DELTA. The ACID transaction was suspended prior to Commit.
2. Second session: Start a second ACID transaction T2, which selects random values of PS_PARTKEY and PS_SUPPKEY and returns all columns of the PARTSUPP table for which PS_PARTKEY and PS_SUPPKEY are equal to the selected values.
3. T2 completes.
4. T1 is allowed to complete.
5. Verify that the appropriate rows in the ORDERS, LINEITEM and HISTORY tables are changed.

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

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

Isolation Test 6 specifies that Q1 (from TPC-H specification Clause 2.4) must be used as T1. 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_EXTENDEDPRICE*(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') - 0 DAYS
  AND L_QUANTITY != (SELECT COUNT(*) FROM TPCD.LINEITEM AS L2 WHERE L2.L_DISCOUNT =
L_DISCOUNT
  AND L_EXTENDEDPRICE != (SELECT COUNT(*) FROM TPCD.LINEITEM AS L2 WHERE
L2.L_QUANTITY = L_QUANTITY)
GROUP BY L_RETURNFLAG, L_LINESTATUS
ORDER BY L_RETURNFLAG, L_LINESTATUS;

```

The following steps were performed to successfully conduct this test:

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

4.4 Durability Requirements

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

4.4.1 Permanent Unrecoverable Failure of Any Durable Medium and Loss of System Power

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

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

1. The consistency test was verified.
2. The current count of the total number of records in the HISTORY table was determined giving hist1.
3. A test to run 200 ACID transactions on each execution was started.
4. One of the disks containing the DB2 TPC-H database transaction log data was removed after at least 25 ACID transactions had completed.
5. The database consistency was not affected because of log mirroring, and ACID transactions continued to execute successfully.
6. One of the disks containing the DB2 TPC-H database data tables was removed after at least 25 additional transactions had executed.
7. The database consistency was not affected because of data mirroring, and ACID transactions continued to execute successfully.
8. The system was shut down by Powering OFF the system after at least 25 additional transactions had completed.
9. The system was powered back on and rebooted.
10. The mirrored disks removed in Steps 4 and 6 were reinserted and reestablished.
11. Step 2 was performed, giving hist2. It was verified that hist2 – hist1 was equal to or greater than the number of records in the success file.
12. Consistency test was verified.

4.4.2 System Crash

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

See section 4.4.1.

4.4.3 Memory Failure

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

See section 4.4.1

5.0 Clause 4: Scaling and Database Population

5.1 Initial Cardinality of Tables

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

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

Table 5. 1: Initial Number of Rows

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

5.2 Distribution of Tables and Logs Across Media

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

DB2 was configured on 16 HP ProLiant DL760 X900 nodes on the test system.

Each node had:

- 2 x Smart Array 5300 disk controllers
- 2 x StorageWorks Enclosure 4354R
- 24 x 18.2GB 15k rpm external disk drives
- 1 x 9.1GB 10k rpm internal disk drives

For each node, all 24 disks were used for the table data tablespaces (LINEITEM_TABLE, OTHER_STUFF), the index tablespaces (LINEITEM_INDEXES, OTHER_INDEXES) and the temporary tablespace (TEMP_TABLE1).

The database log was distributed on 6 disks.

The benchmark execution programs resided on a shared home directory NFS (Z:) mounted from the coordinator node. The tablespace for NATION and REGION tables resided also on that drive.

All tablespaces above were mirrored. For each mirror, the primary and secondary copies were on separate disks.

The operating system resided on 1 internal disks on each node.

A detailed description of distribution of tablespaces and log can be found in Table 5.2.1 and Table 5.2.2

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

SMART Array Controller Number	SMART Logical Drive Array Letter	Number of Physical Drives in SMART LDA	SMART Logical Drive Number	SMART Fault Tolerance	Disk Format	Size (MB)	Contents
0	A	6	1	RAID0+1	RAW	4500	Lineitem table
			2	RAID0+1	RAW	500	Lineitem indexes
			3	RAID0+1	RAW	8000	Temp tables
			4	RAID0+1	RAW	2000	Other Tables
			5	RAID0+1	RAW	384	Other Indexes
0	B	6	1	RAID0+1	RAW	4500	Lineitem table
			2	RAID0+1	RAW	500	Lineitem indexes
			3	RAID0+1	RAW	8000	Temp tables
			4	RAID0+1	RAW	2000	Other Tables
			5	RAID0+1	RAW	384	Other Indexes
1	A	6	1	RAID0+1	RAW	4500	Lineitem table
			2	RAID0+1	RAW	500	Lineitem indexes
			3	RAID0+1	RAW	8000	Temp tables
			4	RAID0+1	RAW	2000	Other Tables
			5	RAID0+1	RAW	384	Other Indexes
1	B	6	1	RAID0+1	RAW	4500	Lineitem table
			2	RAID0+1	RAW	500	Lineitem indexes
			3	RAID0+1	RAW	8000	Temp tables
			4	RAID0+1	RAW	2000	Other Tables
			5	RAID0+1	RAW	384	Other Indexes
			6	RAID0+1	RAW	5000	Log

Table 5.2.2: Distribution of TPC-H Database Tables Across Media

Ctrl	LINEITEM TABLE	LINEITEM INDEXES	TEMP TABLE	OTHER TABLES	OTHER INDEXES
0 -A	4,500 MB 25%	512 MB 25%	8,000 MB 25%	2,000 MB 25%	384 MB 25%
0 -B	4,500 MB 25%	512 MB 25%	8,000 MB 25%	2,000 MB 25%	384 MB 25%
1 -A	4,500 MB 25%	512 MB 25%	8,000 MB 25%	2,000 MB 25%	384 MB 25%
1 -B	4,500 MB 25%	512 MB 25%	8,000 MB 25%	2,000 MB 25%	384 MB 25%
Total	18,000 MB 100%	2,048 MB 100%	32,000 MB 100%	8,000 MB 100%	1,536 MB 100%

Note: An additional 28 disks/node (28 x 18.2 GB) were used to store the flat data files and were not priced. The disk controller for these drives was physically removed from the system after the initial loading of the 300 GB database. The disk controller and corresponding 28 drives/node remained physically detached from the system during the execution of the power, throughput, qualification and ACID tests.

5.3 Mapping of Database Partitions/Replications

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

The database was not replicated. The database was physically partitioned into 16 logical nodes, each residing on one physical node.

5.4 Implementation of RAID

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

RAID 0+1 was used for the entire database and database recovery logs.

5.5 DBGEN Modifications

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

We used a modified version of the standard distribution of DBGEN 1.3.0 so that data could be generated in parallel for each of the nodes, and to allow for a scale factor of 3000. Also DBGEN was modified to facilitate the generation of data for a specific node in a clustered environment. See Appendix B for details.

5.6 Database Load time

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

The database load time was 1 hour 18 minutes 4 seconds.

5.7 Data Storage Ratio

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

Disk Type	Number of Disks	Space per Disk	Total Disk Space	Data Storage Ratio
9.1 GB 10k rpm Ultra3 SCSI	1 /node	8.46 GB	135.36	
18.2 GB 15k rpm Ultra3 SCSI	24 /node	16.92 GB	6512.64	22.61

5.8 Database Load Mechanism Details and Illustration

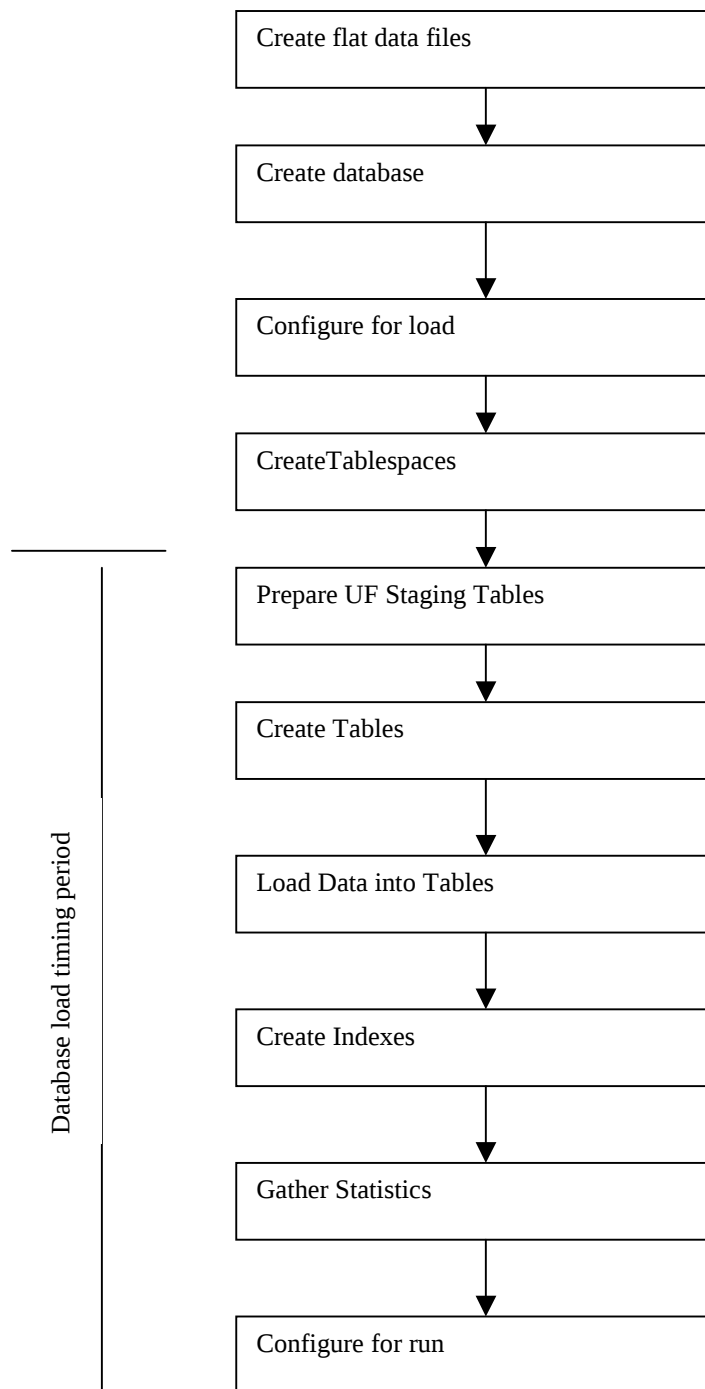
The details of the database load must be disclosed, including a block diagram illustrating the overall process. Disclosure of the load procedure includes all steps, scripts, input and configuration files required to completely reproduce the test and qualification databases.

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

The NATION and REGION tables were created on node 1 and then loaded from dbgen output. The other tables were loaded on all of the nodes.

The tables were loaded as depicted in Figure 5.8.

Figure 5.8: Block Diagram of Database Load Process



6.0 Clause 5: Performance Metrics and Execution Rules Related Items

6.1 Steps in the Power Test

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

The following steps were used to implement the power test:

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

6.2 Timing Intervals for Each Query and Refresh Function

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

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

6.3 Number of Streams for The Throughput Test

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

Seven streams were used for the Throughput Test.

6.4 Start and End Date/Times for Each Query Stream

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

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

6.5 Total Elapsed Time for the Measurement Interval

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

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

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

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

Stream ID	RF	Start Date	Start time	Stop Date	Stop Time
Stream00	RF1	4/3/2002	21:46:02	4/3/2002	21:47:13
Stream00	RF2	4/3/2002	22:25:54	4/3/2002	22:26:47
Stream01	RF1	4/4/2002	1:50:40	4/4/2002	1:51:46
Stream01	RF2	4/4/2002	1:51:46	4/4/2002	1:52:40
Stream02	RF1	4/4/2002	1:52:40	4/4/2002	1:53:53
Stream02	RF2	4/4/2002	1:53:53	4/4/2002	1:54:45
Stream03	RF1	4/4/2002	1:54:45	4/4/2002	1:55:58
Stream03	RF2	4/4/2002	1:55:58	4/4/2002	1:56:52
Stream04	RF1	4/4/2002	1:56:52	4/4/2002	1:58:05
Stream04	RF2	4/4/2002	1:58:05	4/4/2002	1:58:59
Stream05	RF1	4/4/2002	1:58:59	4/4/2002	2:00:10
Stream05	RF2	4/4/2002	2:00:10	4/4/2002	2:01:04
Stream06	RF1	4/4/2002	2:01:04	4/4/2002	2:02:17
Stream06	RF2	4/4/2002	2:02:17	4/4/2002	2:03:09

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

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

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

6.8 Performance Metrics

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

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

6.9 The Performance Metric and Numerical Quantities from Both Runs

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

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

Run	QppH @ 300GB	QthH @ 300GB	QphH @ 300GB
Run 1	15372.8	10985.6	12995.4
Run 2	15622.5	11177.6	13214.5

6.11 System Activity Between Tests

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

The system was not restarted between runs.

7.0 Clause 6: SUT and Driver Implementation Related Items

7.1 Driver

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

Appendix D "Implementation specific layer and Driver Source Code" contains the source code used for the driver and all scripts used in connection with it.

The power test is invoked by calling tpdbatch with the stream number 0 specified, an indication that the refresh functions must be run, and the SQL file that contains the power stream queries.

The Throughput test is invoked by initiating a call to tpdbatch for every query stream that will be run. tpdbatch gets the stream number for each of the streams, and the SQL file specific to that stream number as the queries to execute. The refresh function is initiated as a separate call to tpdbatch with the SQL script for the refresh functions and the total number of query streams specified.

7.2 Implementation Specific Layer (ISL)

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

The implementation specific layer is a single executable SQL application that uses embedded dynamic SQL to process the EQT generated by QGEN. The application is called tpdbatch to indicate that it processes a batch of TPC-H queries, although it is completely capable of processing any arbitrary SQL statement (both DML and DDL). A separate instance of tpdbatch 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 tpdbatch is invoked, it is provided with a context of whether it is running a power test, query stream or refresh stream, as well as an input file containing the 22 queries and/or refresh functions. tpdbatch then connects to the database, performs any session initialization as well as preparing output files required by the auditor. Then it proceeds to read from the input file and processes each query or refresh function in turn.

For queries, each query is prepared, described, and a cursor is opened and used to fetch the required number of rows. After the last row has been retrieved a commit is issued. For the refresh functions, during the database build all data is first split for each node. For RF1, the data for each node is further split into n equal portions for both the lineitem and orders tables taking care that the records for the same orderkey remain in the same set. For RF2, the data for each node is further split into m equal portions. During the run, when tpdbatch encounters a call to execute RF1, it first calls a shell script which loads these n sets of data into the temporary tables (one each for lineitem and orders), containing a column to hold the chunk number. Then tpdbatch forks off n children to do an insert with subselect into the original lineitem and orders tables. When tpdbatch encounters a call to execute RF2, it calls a shell script that loads these data into a single staging table. Then tpdbatch forks off p children (where $p * x = m$) to do x sets of deletes from the orders and lineitem tables with a subselect from the staging table.

7.3 Profile-Directed Optimization

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

Profile-directed optimization was not used.

8.0 Clause 7: Pricing Related Items

8.1 Hardware and Software Used

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

A detailed list of all hardware and software, including the 5-year price, is provided in the Executive Summary at the front of this report. The price quotations are included in Appendix F, at the end of this document.

8.2 Total 3 Year Price

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

A detailed list of all hardware and software, including the 3-year price, is provided in the Executive Summary at the front of this report. The price quotations are included in Appendix F, at the end of this document. For a large purchase and cash discount, this purchase qualifies for a 16% discount from Hewlett Packard Company.

8.3 Availability Date

The committed delivery date for general availability of products used in the price calculations must be reported. When the priced system includes products with different availability dates, the availability date reported on the executive summary must be the date by which all components are committed to being available. The full disclosure report must report availability dates individually for at least each of the categories for which a pricing subtotal must be provided.

The HP ProLiant DL760 X900, system memory, additional processors, disk controllers and hard drives are available at the time of publication. All other hardware is generally available at the time of publication.

The system software, Microsoft Windows 2000 Advanced Server, used in this test is generally available at the time of publication. Service Pack 2 for Microsoft Windows 2000 Advanced Server is generally available at the time of publication. The database software, IBM DB2 UDB 7.2.0 for Windows 2000 is generally available at the time of publication. Fix Pack 4 for DB2 is generally available at the time of publication.

Any modifications made to DB2 that were not in FP-4 will be available by June 20, 2002

8.4 Country-Specific Pricing

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

The configuration is priced for the United States of America

.

9.0 Clause 9: Related Items

9.1 Auditors' Report

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

This implementation of the TPC Benchmark H was audited by Bradley J. Askins of InfoSizing (www.sizing.com). Further information regarding the audit process may be obtained from:

InfoSizing
1373 North Franklin Street
Colorado Springs, CO 80903-2527
Telephone: (719) 473-7555
Fax: (719) 473-7554

Requests for this TPC Benchmark H Full Disclosure Report should be sent to:

Transaction Processing Performance Council
c/o Shanley Public Relations
777 North First Street, Suite 6000
San Jose, CA 95112-6311 USA
Telephone: (408) 295-8894
Fax: (408) 295-9768

INFO SIZING



Benchmark Sponsors: Daniel Pol
Compaq Computer Corporation
20555 SH-249, MS 150402
Houston, TX 77070

April 5, 2002

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

Platform: **COMPAQ ProLiant DL760 X900-64P**
Database Manager: **IBM DB2 UDB 7.2**
Operating System: **Microsoft Windows 2000 Advanced Server**

The results were:

CPU (Speed)	Memory	Disks	QphH@100GB
16 nodes: COMPAQ ProLiant DL760 X900-64P			
4 x Intel Pentium III Xeon (900 MHz)	2MB L2 Cache/cpu 4 GB Main	24 x 18 GB 1 x 9 GB	12995.4

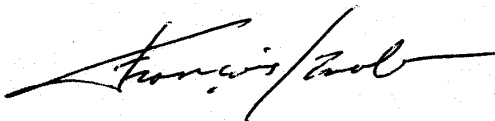
In my opinion, this performance result was produced in compliance with the TPC's requirements for the benchmark. The following verification items were given special attention:

- The database records were defined with the proper layout and size
- The database population was generated using DBGEN
- The database was properly scaled to 300GB and populated accordingly
- The compliance of the database auxiliary data structures was verified
- The database load time was correctly measured and reported

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

- The required ACID properties were verified and met
- The query input variables were generated by QGEN
- The query text was produced using minor modifications
- The execution of the queries against the SF1 database produced compliant answers
- A compliant implementation specific layer was used to drive the tests
- The throughput tests involved 6 query streams
- The ratio between the longest and the shortest query was such that no query timing was adjusted
- The execution times for queries and refresh functions were correctly measured and reported
- The repeatability of the measured results was verified
- The required amount of database log was configured
- The system pricing was verified for major components and maintenance
- The major pages from the FDR were verified for accuracy

Respectfully Yours,



François Raab, President



Bradley J. Askins, Auditor

Appendix A: Tunable Parameters

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

A.1 DB2 Database TPCB Configuration

Database Configuration for Database TPCB

Database configuration release level = 0x0900
 Database release level = 0x0900
 Database territory = US
 Database code page = 1252
 Database code set = IBM-1252
 Database country code = 1
 Dynamic SQL Query management (DYN_QUERY_MGMT) = DISABLE
 Directory object name (DIR_OBJ_NAME) =
 Discovery support for this database (DISCOVER_DB) = ENABLE
 Default query optimization class (DFT_QUERYOPT) = 7
 Degree of parallelism (DFT_DEGREE) = 4
 Continue upon arithmetic exceptions (DFT_SQLMATHWARN) = NO
 Default refresh age (DFT_REFRESH_AGE) = 0
 Number of frequent values retained (NUM_FREQVALUES) = 0
 Number of quantiles retained (NUM_QUANTILES) = 600
 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
 Data Links Token Expiry Interval (sec) (DL_EXPINT) = 60
 Data Links Number of Copies (DL_NUM_COPIES) = 1
 Data Links Time after Drop (days) (DL_TIME_DROP) = 1
 Data Links Token in Uppercase (DL_UPPER) = NO
 Data Links Token Algorithm (DL_TOKEN) = MAC0
 Database heap (4KB) (DBHEAP) = 20000
 Catalog cache size (4KB) (CATALOGCACHE_SZ) = 386
 Log buffer size (4KB) (LOGBUFSZ) = 2048
 Utilities heap size (4KB) (UTIL_HEAP_SZ) = 20000
 Buffer pool size (pages) (BUFFPAGE) = 55000
 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) = 384
 Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 1024
 Sort list heap (4KB) (SORTHEAP) = 22500
 SQL statement heap (4KB) (STMTHEAP) = 2048
 Default application heap (4KB) (APPLHEAPSZ) = 768
 Package cache size (4KB) (PCKCACHESZ) = 640
 Statistics heap size (4KB) (STAT_HEAP_SZ) = 2048
 Interval for checking deadlock (ms) (DLCHKTIME) = 5000
 Percent. of lock lists per application (MAXLOCKS) = 15
 Lock timeout (sec) (LOCKTIMEOUT) = -1
 Changed pages threshold (CHNGPGS_THRESH) = 80
 Number of asynchronous page cleaners (NUM_IOCLEANERS) = 16
 Number of I/O servers (NUM_IOSERVERS) = 16
 Index sort flag (INDEXSORT) = YES
 Sequential detect flag (SEQDETECT) = YES
 Default prefetch size (pages) (DFT_PREFETCH_SZ) = 16
 Track modified pages (TRACKMOD) = OFF
 Default number of containers = 1

Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32
 Max number of active applications (MAXAPPLS) = 40
 Average number of active applications (AVG_APPLS) = 1
 Max DB files open per application (MAXFILOP) = 1024
 Log file size (4KB) (LOGFILSZ) = 8192
 Number of primary log files (LOGPRIMARY) = 100
 Number of secondary log files (LOGSECOND) = 25
 Changed path to log files (NEWLOGPATH) =
 Path to log files = \\.\PhysicalDrive20
 First active log file =
 Group commit count (MINCOMMIT) = 1
 Percent log file reclaimed before soft ckckpt (SOFTMAX) = 800
 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
 Number of database backups to retain (NUM_DB_BACKUPS) = 12
 Recovery history retention (days) (REC_HIS_RETENTN) = 366
 TSM management class (TSM_MGMTCLASS) =
 TSM node name (TSM_NODENAME) =
 TSM owner (TSM_OWNER) =
 TSM password (TSM_PASSWORD) =

A.2 DB2 Database Manager Configuration

Database Manager Configuration

Node type = Partitioned Database Server with local and remote clients
 Database manager configuration release level = 0x0900
 Maximum total of files open (MAXTOTFILOP) = 16000
 CPU speed (millisec/instruction) (CPUSPEED) = 6.297923e-007
 Communications bandwidth (MB/sec) (COMM_BANDWIDTH) = 1.000000e+001
 Max number of concurrently active databases (NUMDB) = 1
 Data Links support (DATALINKS) = NO
 Federated Database System Support (FEDERATED) = NO
 Transaction processor monitor name (TP_MON_NAME) =
 Default charge-back account (DFT_ACCOUNT_STR) =
 Java Development Kit 1.1 installation path (JDK11_PATH) =
 Diagnostic error capture level (DIAGLEVEL) = 3
 Notify Level (NOTIFYLEVEL) = 2
 Diagnostic data directory path (DIAGPATH) = z:\bigblue\blue
 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) = 12

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) = 430000
 Directory cache support (DIR_CACHE) = YES
 Application support layer heap size (4KB) (ASLHEAPSZ) = 15
 Max requester I/O block size (bytes) (RQRIOLBK) = 32767
 DOS requester I/O block size (bytes) (DOS_RQRIOLBK) = 4096
 Query heap size (4KB) (QUERY_HEAP_SZ) = 1000
 DRDA services heap size (4KB) (DRDA_HEAP_SZ) = 128
 Priority of agents (AGENTPRI) = 2
 Max number of existing agents (MAXAGENTS) = 160
 Agent pool size (NUM_POOLAGENTS) = 64
 Initial number of agents in pool (NUM_INITAGENTS) = 4
 Max number of coordinating agents (MAX_COORDAGENTS) = (MAXAGENTS - NUM_INITAGENTS)
 Max no. of concurrent coordinating agents (MAXCAGENTS) = MAX_COORDAGENTS
 Max number of logical agents (MAX_LOGICAGENTS) = MAX_COORDAGENTS
 Keep DARI process (KEEPDARI) = YES
 Max number of DARI processes (MAXDARI) = MAX_COORDAGENTS
 Initialize DARI process with JVM (INITDARI_JVM) = NO
 Initial number of fenced DARI process (NUM_INITDARIS) = 0
 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) = DB2cBLUE
 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) = 768
 Number of FCM request blocks (FCM_NUM_RQB) = 1536
 Number of FCM connection entries (FCM_NUM_CONNECT) = 1280
 Number of FCM message anchors (FCM_NUM_ANCHORS) = 128
 Node connection elapse time (sec) (CONN_ELAPSE) = 10
 Max number of node connection retries (MAX_CONNRETRIES) = 5
 Max time difference between nodes (min) (MAX_TIME_DIFF) = 60
 db2start/db2stop timeout (min) (START_STOP_TIME) = 10

A.3 DB2 Environment Variables

DB2_ANTIJOIN=Y
 DB2_STRIPED_CONTAINERS=ON
 DB2_LIKE_VARCHAR=y,y

DB2BPVARS=z:\tpch\cpq_ddl\cpq_scattered-read
 DB2_HASH_JOIN=Y
 DB2_BINSORT=N
 DB2ACCOUNTNAME=db2\tpch
 DB2INSTOWNER=G1
 DB2PORTRANGE=50101:50132
 DB2MEMMAXFREE=137438953472,20
 DB2_RR_TO_RS=ON
 DB2OPTIONS=-t -v +c
 DB2NTPRICLASS=R
 DB2NTNOCACHE=ON
 DB2INSTPROF=z:\bigblue
 DB2_PARALLEL_IO=*

A.4 Windows 2000 Environment Variables

ALLUSERSPROFILE=C:\Documents and Settings\All Users
 APPDATA=C:\Documents and Settings\Administrator\Application Data
 CLASSPATH=.;C:\Program Files\SQLLIB\java\db2java.zip;C:\Program Files\SQLLIB\java\runtime.zip;C:\Program Files\SQLLIB\java\sqlj.zip;C:\Program Files\SQLLIB\bin
 CommonProgramFiles=C:\Program Files\Common Files
 COMPUTERNAME=G14
 ComSpec=C:\WINNT\system32\cmd.exe
 DB2INSTANCE=BLUE
 DB2TEMPDIR=C:\Program Files\sqllib
 DSS_NODE=13
 DSS_NUMNODES=32
 HOMEDRIVE=C:
 HOMEPATH=\\Documents and Settings\Administrator
 ICM_FOLDER=Information Catalog Manager
 IMNINST=help
 IMNINSTSRV=C:\PROGRA~1\IBM\IMNMQ
 INCLUDE=C:\Program Files\SQLLIB\INCLUDE;C:\Program Files\SQLLIB\LIB;C:\Program Files\SQLLIB\TEMPLATES\INCLUDE
 LIB=C:\Program Files\SQLLIB\LIB
 LOGONSERVER=\\G14
 MDIS_PROFILE=C:\Program Files\SQLLIB\METADATA\PROFILES
 NUMBER_OF_PROCESSORS=4
 OS=Windows_NT
 Os2LibPath=C:\WINNT\system32\os2\dll;
 Path=z:\perl\bin;z:\tools;z:\tpch\tools;C:\WINNT\system32;C:\WINNT;C:\WINNT\System32\Wbem;C:\PROGRA~1\IBM\IMNMQ;C:\Program Files\SQLLIB\BIN;C:\Program Files\SQLLIB\FUNCTION;C:\Program Files\SQLLIB\SAMPLES\REPL;C:\Program Files\SQLLIB\HELP
 PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
 PROCESSOR_ARCHITECTURE=x86
 PROCESSOR_IDENTIFIER=x86 Family 6 Model 10 Stepping 4, GenuineIntel
 PROCESSOR_LEVEL=6
 PROCESSOR_REVISION=0a04
 ProgramFiles=C:\Program Files
 PROMPT=\$P\$G
 SESSIONNAME=Console
 SystemDrive=C:
 SystemRoot=C:\WINNT
 TEMP=C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp
 TMP=C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp
 TPCD_DBNAME=tpcd
 USERDOMAIN=G14
 USERNAME=Administrator
 USERPROFILE=C:\Documents and Settings\Administrator
 VWSPATH=C:\Program Files\SQLLIB
 VWS_FOLDER=IBM DB2
 VWS_LOGGING=C:\Program Files\SQLLIB\LOGGING
 VWS_TEMPLATES=C:\Program Files\SQLLIB\TEMPLATES
 windir=C:\WINNT

A.5 Windows 2000 Configuration

All services were running in their default state.

From the Control Panel the Network, Services, Server, Properties, Optimization was changed from the default of “Maximize Throughput for File Sharing” to “Maximize Throughput for Network Applications”. This setting remained in effect for the duration of the benchmark.

From the Control Panel the System, Performance, Optimize performance for was set to “Applications”. This setting remained in effect for the duration of the benchmark.

The Gigabit network card and switch were set to use 4088 bytes frames.

A.6 HP SMART Array Controller Cache Settings and registry entries

The HP SMART Array Controller cache settings for all the database data drives was reconfigured for the default ratio of 50% read / 50% write . The HP SMART Array Controller cache for the database log drives was disabled.

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\cpqciiss
b]
"Type"=dword:00000001
"Start"=dword:00000000
"ErrorControl"=dword:00000001
"Tag"=dword:00000102
"ImagePath"=hex(2):53,00,79,00,73,00,74,00,65,00,6d,00,33,00,32,00,5c,00,
44,00,\
52,00,49,00,56,00,45,00,52,00,53,00,5c,00,63,00,70,00,71,00,63,00,69,00,73
,\
00,73,00,62,00,2e,00,73,00,79,00,73,00,00,00
"DisplayName"="Compaq CISS Controllers Device Driver"
"Group"="port"
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\cpqciiss
b\Parameters]
"CompletionMode"=dword:00000001
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\cpqciiss
b\Security]
"Security"=hex:01,00,14,80,a0,00,00,00,ac,00,00,00,14,00,00,00,30,00,00,00,
02,\
```

```
00,1c,00,01,00,00,00,02,80,14,00,ff,01,0f,00,01,01,00,00,00,00,01,00,00,\
```

```
00,00,02,00,70,00,04,00,00,00,00,00,18,00,fd,01,02,00,01,01,00,00,00,00,
,\
```

```
05,12,00,00,00,74,00,69,00,00,00,1c,00,ff,01,0f,00,01,02,00,00,00,00,05,\
```

```
20,00,00,00,20,02,00,00,76,00,65,00,00,00,18,00,8d,01,02,00,01,01,00,00,
,\
```

```
00,00,05,0b,00,00,00,20,02,00,00,00,00,1c,00,fd,01,02,00,01,02,00,00,00,
,\
```

```
00,05,20,00,00,23,02,00,00,76,00,65,00,01,01,00,00,00,00,05,12,00,00,
,\
00,01,01,00,00,00,00,05,12,00,00,00
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\cpqciiss
b\Enum]
"0"="PCI\VEN_0E11&DEV_B060&SUBSYS_40700E11&REV_02\3&26
7a616a&0&08"
```

```
"Count"=dword:00000004
"NextInstance"=dword:00000004
"1"="PCI\VEN_0E11&DEV_B060&SUBSYS_40700E11&REV_02\3&26
7a616a&0&10"
"2"="PCI\VEN_0E11&DEV_B060&SUBSYS_40700E11&REV_02\3&13
c0b0c5&0&28"
"3"="PCI\VEN_0E11&DEV_B060&SUBSYS_40700E11&REV_02\3&10
70020&0&10"
```

A.7 Compiler Options for TPCD Batch Driver

```
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 %1
db2 prep tpcdbatch.sqc bindfile package isolation rr blocking all datatype iso
db2 prep tpcdUF.sqc bindfile package isolation rs blocking all optlevel 1
degree 1 datatype iso
db2 connect reset
db2 terminate
REM make sure LIBPATH is set to include the compiler libraries and db2
libraries
cl -c -Z7 -G6 -DWIN32 -DSQLWINT -D_X86_=1 -W3 -J tpcdbatch.C
cl -c -Z7 -G6 -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.8 Compiler Options for ACID Driver

```
db2start
db2 connect to %1
db2 prep acid.sqc bindfile package isolation rr
db2 connect reset
db2 terminate
```

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

```
REM !!!! The compilation of the mainacid program absolutely requires
REM !!!! including the COMMODE.OBJ file.
REM !!!! This is used to avoid NT file buffering -gav
REM Attempting to link with COMMODE.OBJ ...
link -debug -out:mainacid.exe commode.obj mainacid.obj acid.obj -
align:0x1000 "user32.lib" "kernel32.lib" db2api.lib -subsystem:console
```

A.9 Cpq_scattered-read

```
NUMPREFETCHQUEUES=4
PREFETCHQUEUESIZE=200
```

```
# 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
LOG_PERF_HINTS=5
```


Appendix B: Database Build Scripts

B.1 buildtpcd

```
#!/usr/bin/perl
# usage buildtpcd [QUAL]

# ASSUMPTIONS: all ddl files have commits in them!
($myName = $0) =~ s@.*\/@@; $usage="
Usage: buildtpcd [QUAL]
    where QUAL is the optional parameter saying to build the qualification
        database (sf = .1 = 100MB)\n";

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

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

# Use the macros in here so that they can handle the platform differences.
# macro.pl should be sourced from cmvc, other people wrote and maintain it.
require "macro.pl";
require "tpcdmacro.pl";

# Make output unbuffered.
select(STDOUT);
$| = 1 ;

# verify that necessary environment variables for building the database
# are present. Default those that aren't necessary
require "version";

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" || $platform eq "sun" || $platform eq "ptx" ||
$platform eq "hp" || $platform eq "linux")
    {
        $ENV{"TPCD_DBPATH"} = $ENV{"HOME"};
    }
    elsif ( $platform eq "nt" )
    {
        $ENV{"TPCD_DBPATH"} = $ENV{"HOMEDRIVE"};
    }
    else
    {
        die "platform $platform not supported yet\n";
    }
}
if (length($ENV{"TPCD_DDLPATH"}) <= 0)
{
    # if no db pathname specified, use default
    $ENV{"TPCD_DDLPATH"} =
"/afs/tor/groups/dbp/perf/benchmark/tpcd/ddl/vanilla";
}
if (length($ENV{"TPCD_GENERATE_SEED_FILE"}) <= 0)
{
    # if no db pathname specified, use default
    $ENV{"TPCD_GENERATE_SEED_FILE"} = "no";
}
if (length($ENV{"TPCD_DDL"}) <= 0)
{
    $ENV{"TPCD_DDL"} = "dss.ddl";
}
if (length($ENV{"TPCD_STAGING_TABLE_DDL"}) <= 0)
{
    $ENV{"TPCD_STAGING_TABLE_DDL"} = "NULL";
}
if (length($ENV{"TPCD_PRELOAD_STAGING_TABLE_SCRIPT"}) <=
0)
{
    $ENV{"TPCD_PRELOAD_STAGING_TABLE_DDL"} = "NULL";
}
if (length($ENV{"TPCD_DELETE_STAGING_TABLE_SQL"}) <= 0)
{
    $ENV{"TPCD_DELETE_STAGING_TABLE_DDL"} = "NULL";
}
if (length($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/sql/lib/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_LOG_DIR_SETUP_SCRIPT"}) <= 0)
{
    $ENV{"TPCD_LOG_DIR_SETUP_SCRIPT"} = "NULL";
}
if (length($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"} eq "NULL")
{
    $ENV{"TPCD_SMPDEGREE"} = 1;
}
if (length($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)
{
    $ENV{"TPCD_BUFFERPOOL_DEF"}="NULL"
}
if (length($ENV{"TPCD_EXPLAIN_DDL"}) <= 0)
{
    $ENV{"TPCD_EXPLAIN_DDL"}="NULL"
}
if (length($ENV{"TPCD_NODEGROUP_DEF"}) <= 0)
{
    $ENV{"TPCD_NODEGROUP_DEF"}="NULL"
}
if (length($ENV{"TPCD_PHYS_NODE"}) <= 0)
{
    $ENV{"TPCD_NODEGROUP_DEF"}="NULL"
}
if (length($ENV{"TPCD_APPEND_ON"}) <= 0)
{
    $ENV{"TPCD_APPEND_ON"}="yes"
}
}
#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"};
$stagingTbl=$ENV{"TPCD_STAGING_TABLE_DDL"};
$preloadSampleUF=$ENV{"TPCD_PRELOAD_STAGING_TABLE_SCRIPT"};
$deleteSampleUF=$ENV{"TPCD_DELETE_STAGING_TABLE_SQL"};
$buffpooldef=$ENV{"TPCD_BUFFERPOOL_DEF"};
$nodegroupdef=$ENV{"TPCD_NODEGROUP_DEF"};
$tbspdddl=$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"};
$idtemp=$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"};
$genSeed=$ENV{"TPCD_GENERATE_SEED_FILE"};
$appendOn=$ENV{"TPCD_APPEND_ON"};

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"};
$logDirScript=$ENV{"TPCD_LOG_DIR_SETUP_SCRIPT"};
$machine=$ENV{"TPCD_MACHINE"};
$configfile=$ENV{"TPCD_CONFIGFILE"};
$dbmconfig=$ENV{"TPCD_DBM_CONFIG"};
$loadconfigfile=$ENV{"TPCD_LOAD_CONFIGFILE"};
$loadDBMconfig=$ENV{"TPCD_LOAD_DBM_CONFIGFILE"};
$smpdegree=$ENV{"TPCD_SMPDEGREE"};
$agentpri=$ENV{"TPCD_AGENTPRI"};
$activate=$ENV{"TPCD_ACTIVATE"};
$RealAudit=$ENV{"TPCD_AUDIT"};
$loadscript=$ENV{"TPCD_LOAD_SCRIPT"};
$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$loadsetScript=$ENV{"TPCD_LOAD_DB2SET_SCRIPT"};
$setScript=$ENV{"TPCD_DB2SET_SCRIPT"};
$explainDDL=$ENV{"TPCD_EXPLAIN_DDL"};
$user=$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"};

print "DBNAME= $dbname\n\n";

$sf=$ENV{"TPCD_QUAL_SF"};
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";
}
$tbspdddl=$ENV{"TPCD_QUAL_TBSP_DDL"};
$input=$ENV{"TPCD_QUAL_INPUT"};
if ( length($ENV{"TPCD_QUALCONFIGFILE"}) <= 0 )
{
    die "TPCD_QUALCONFIGFILE environment variable not set\n";
}
$configfile=$ENV{"TPCD_QUALCONFIGFILE"};
if ( length($ENV{"TPCD_DBM_QUALCONFIG"}) <= 0 )
{
    die "TPCD_DBM_QUALCONFIG environment variable not set\n";
}
$dbmconfig=$ENV{"TPCD_DBM_QUALCONFIG"};
if ( length($ENV{"TPCD_LOAD_QUALCONFIGFILE"}) <= 0 )
{
    die "TPCD_LOAD_QUALCONFIGFILE environment variable not set\n";
}
$loadconfigfile=$ENV{"TPCD_LOAD_QUALCONFIGFILE"};
if ( length($ENV{"TPCD_LOAD_DBM_QUALCONFIGFILE"}) <= 0 )
{
    die "TPCD_LOAD_DBM_QUALCONFIGFILE environment variable not set\n";
}
$loadDBMconfig=$ENV{"TPCD_LOAD_DBM_QUALCONFIGFILE"};
if (length($ENV{"TPCD_LOG_DIR"}) <= 0)
{
    $ENV{"TPCD_LOG_DIR"} = "NULL";
}
$logDir=$ENV{"TPCD_LOG_QUAL_DIR"};
}

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

##### set up the config parms for the load, indexes and stats #####
if ($loadconfigfile eq "NULL")
{
    if ( ( $machine eq "NULL" ) )
    {
        die "Neither a LOAD config file name nor a machine size has been specified!\n";
    }
}
```

```

}
$ioclnrs=1;
$chngpgs=60;

if ( $machine eq "small" )
{
    $buffpage = 5000;
    $sorheap = 3000;
    $sheapthres = 8000;
    $util_heap_sz = 5000;
    $ioservers = 6;
}
elseif ( $machine eq "medium" )
{
    $buffpage = 10000;
    $sorheap = 8000;
    $sheapthres = 20000;
    $util_heap_sz = 10000;
    $ioservers = 10;
}
elseif ( $machine eq "big" )
{
    $buffpage = 30000;
    $sorheap = 20000;
    $sheapthres = 50000;
    $util_heap_sz = 30000;
    $ioservers = 20;
}
elseif ( $machine eq "sunsmp" )
{
    $buffpage = 60000;
    $sorheap = 20000;
    $sheapthres = 80000;
    $util_heap_sz = 30000;
    $ioservers = 80;
}
elseif ( $machine eq "eastwood" )
{
    $buffpage = 80000;
    $sorheap = 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 ( $nodegroupdef ne "NULL" )
{
    print " Nodegroup ddl in $ddlpath${delim}$nodegroupdef \n";
}
if ( $buffpooldef ne "NULL" )
{
    print " Bufferpool ddl in $ddlpath${delim}$buffpooldef \n";
}
print " Table ddl in $ddlpath${delim}$ddl \n";
print " Index ddl in $ddlpath${delim}$indexddl \n";
if ( $extraindex ne "no" )
{
    print " Indices to create after the load $ddlpath${delim}$extraindex \n";
}

```

```

}
if ( $loadscript eq "NULL" )
{
    if ( $input eq "NULL" )
    {
        print " Data generated by DBGEN in $dbgen\n";
    }
    else
    {
        print " Data loaded from flat files in $input\n";
    }
}
if ( $earlyindex eq "yes" )
{
    print " Indexes created before loading\n";
}
else
{
    print " Indexes created after loading\n";
}
if ( $addRI ne "NULL" )
{
    print " RI being used from $ddlpath${delim}$addRI\n";
}
if ( $astFile ne "NULL" )
{
    print " AST being used from $ddlpath${delim}$astFile\n";
}
if ( $loadstats eq "yes" )
{
    if ( $earlyindex eq "yes" )
    {
        print " Statistics for tables and indexes gathered during load\n";
    }
    else
    {
        if ( $runstatShort eq "NULL" )
        {
            print " Statistics for tables and indexes gathered after load using
            $ddlpath${delim}$runstats \n";
        }
        else
        {
            print " Statistics for tables and indexes gathered after load using
            $ddlpath${delim}$runstats and $ddlpath${delim}$runstatShort\n";
        }
    }
}
else
{
    if ( $runstatShort eq "NULL" )
    {
        print " Statistics for tables and indexes gathered after load using
        $ddlpath${delim}$runstats \n";
    }
    else
    {
        print " Statistics for tables and indexes gathered after load using
        $ddlpath${delim}$runstats and $ddlpath${delim}$runstatShort\n";
    }
}
if ( $loadconfigfile ne "NULL" )
{
    print " Database Configuration parameters for LOAD taken from
    $ddlpath${delim}$loadconfigfile\n";
}
if ( $loadDBMconfig ne "NULL" )
{
    print " Database manager Configuration parameters for LOAD taken from
    $ddlpath${delim}$loadDBMconfig\n";
}

```

```

}
if ( $configfile ne "NULL" )
{
    print " Database Configuration parameters taken from
$ddlpath${delim}$configfile\n";
}
else
{
    print " Database Configuration paramters taken from
$ddlpath${delim}dss.dbmconfig${sfReal}GB\n";
    $configfile="dss.dbmconfig${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 "NULL")
{
    print " Machine size set to $machine so the following configuration\n";
    print " parameters are used for load, create index and runstats: \n";

```

```

print " BUFFPAGE = $buffpage \n";
print " SORTHEAP = $sortheap \n";
print " SHEAPTHRES = $sheapthres\n";
print " NUM_IOSERVERS = $ioservers\n";
print " NUM_IOCLEANERS = $ioclnrs\n";
print " CHNGPGS_THRESH = $chngpgs\n";
print " UTIL_HEAP_SZ = $util_heap_sz\n";
print " Degree of parallelism (dft_degree and max_querydegree) set to
$smpdegree\n";
print " Parameters for load are: temp file = $ldtemp\n";
print " sort buf = $sortbuf\n";
print " ld parallelism = $load_parallelism\n";
if ( $fparse eq "yes" )
{
    print " FASTPARSE used on load\n";
}
}
if ( $loadscript ne "NULL" )
{
    print " Load commands in $ddlpath${delim}$loadscript\n";
}

print " Degree of parallelism (dft_degree and max_querydegree) set to
$smpdegree\n";
if ( $agentpri ne "NULL" )
{
    print " AGENTPRI set to $agentpri\n";
}
if ( $activate eq "yes" )
{
    print " Database will be activated when build is complete\n";
}
if ( $explainDDL ne "NULL" )
{
    print " EXPLAIN DDL being used from
$ddlpath${delim}$explainDDL\n";
}
else
{
    print " EXPLAIN DDL being used from default sqllib directory\n";
}

print "Sleeping for 15 seconds to give you a chance to reconsider...\n";
sleep 15;
##### end echo parameters
#####
# don't need separate calls for mln/mpp vs uni since the $all_in will be
# defined appropriately
if ( $platform eq "nt" )
{
    if (($mode eq "uni") || ($mode eq "smp"))
    {
        #spaces required for NT
        $rc=&doddb_noconn("db2set DB2OPTIONS=\ -t -v +c\\", $all_in);
    }
    else
    {
        $rc=&doddb_noconn("db2set DB2OPTIONS=\\ \ -t -v +c\\", $all_in);
    }
}
else
{
    if (($mode eq "uni") || ($mode eq "smp"))
    {
        $rc=&doddb_noconn("db2set DB2OPTIONS=\ -t -v +c\\", $all_in);
    }
    else
    {

```

```

    $src=&dodb_noconn("db2set DB2OPTIONS=\\\"-t -v +c\\\"",$all_ln);
}
}

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

if ( $platform eq "nt" )
{
    $src=&dodb_noconn("db2set DB2NTNOCACHE=ON",$all_ln);
}

if ( $src != 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" )
{
    if ( $platform eq "nt" )
    {
        ##### Mike , th, the- for some reason rah on NT doesn't like running
        ##### a fully qualified file name. Switch to dir, then run
        ## check CHANGE THIS to have single node and multinode execution
        if ( ( $mode eq "uni" ) || ( $mode eq "smp" ) )
        {
            $ret=system("${ddlpath}${delim}$loadsetScript");
            # $ret=system("cd ${ddlpath} $sep $loadsetScript");
        }
        else
        {
            $ret=system("rah \" call ${ddlpath}${delim}$loadsetScript \"");
        }
    }
    else
    {
        $ret=system("${ddlpath}${delim}$loadsetScript");
    }
    if ( $ret != 0 )
    {
        die "failure setting load time db2set parms from $loadsetScript \n";
    }
}

# stopping and starting db2 before we continue
print "Stopping DB2 ... \n";
$src=system("db2stop");
if ( $src < 0 )
{
    die "failure during db2stop rc = $src \n";
}

print "Starting DB2 ... \n";
$src=system("db2start");
if ( $src != 0 )
{
    die "failure during db2start rc = $src \n";
}

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

```

```

$src = &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

if($src != 0){
    die "Failure during create database. rc = $rc\n";
}

&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 ($logDirScript ne "NULL")
{
    system ("perl $ddlpath${delim}$logDirScript");
}
elseif ( $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);

# setup the load configuration
&loadConfig;
} # end of "CREATE DATABASE" phase

```

```

if (( $buildStage eq "ALL" ) || ( $buildStage eq "CRTTBSP" ) ||
    ( $buildStage eq "LOAD" ) ||
    ( ( $buildStage eq "INDEX" ) && ( $earlyindex eq "yes" ) ) )
{
    # create the nodegroups is there is a file specified
    if ( $nodegroupdef ne "NULL" )
    {
        #run the create bufferpool ddl
        &outtime("*** Creating the nodegroups");
        &dodb2file($dbname,"$ddlpath${delim}$nodegroupdef", $once);
    }
    # create the explain tables - these should go into userspace1 since no
    # other tablespaces exist
    if ( $explainDDL ne "NULL" )
    {
        $explnPathFile="$ddlpath${delim}$explainDDL";
    }
    else
    {
        if ( $platform eq "ptx" )
        {
            $home=$ENV{"HOME"};
            $sqlpath="$home${delim}sqllib";
        }
        if ( $platform ne "nt" )
        {
            $sqlpath="$~${delim}sqllib";
        }
        else
        {
            $sqlpath=$ENV{"DB2PATH"};
            $sqlpath="$~$sqlpath";
        }
        $explnPathFile="$sqlpath${delim}misc${delim}EXPLAIN.DDL";
    }
    &dodb_conn($dbname,
        "db2 -tvf $explnPathFile $sep \
        db2 alter table explain_instance locksize table append on $sep \
        db2 alter table explain_statement locksize table append on $sep \
        db2 alter table explain_argument locksize table append on $sep \
        db2 alter table explain_object locksize table append on $sep \
        db2 alter table explain_operator locksize table append on $sep \
        db2 alter table explain_predicate locksize table append on $sep \
        db2 alter table explain_stream locksize table append on",
        $once);

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

    # create the tablespaces
    &outtime("*** Ready to start creating the tablespaces");
    &dodb2file($dbname,"$ddlpath${delim}$tblspddl", $once);
    # if we are in audit mode, then we must create the the tablespaces and
    # tables for the update functions and we must generate the data for the
    # update functions before we start timing the load. (All activity
    # on the database after the table creation is started and before the
    performance
    # tests are run must be included in load time
    # NOTE: we do not have to do this if we are building the qualification
    database
    &outtime("*** Start of audited Load Time - starting to create tables");

    # create/populate the staging tables
    if ( $stagingTbl ne "NULL" )
    {

```

```

        # staging tables must be created for both test and qualification database
        # but they do not need to be populated for the qualification database
        &dodb2file($dbname,"$ddlpath${delim}$stagingTbl", $once);
        if ( $qual ne "QUAL" )
        {
            if ( $preloadSampleUF ne "NULL" )
            {
                # preload the sample UF data for statistics gathering
                $src = system ("perl $ddlpath${delim}$preloadSampleUF");
            }
            if ( $deleteSampleUF ne "NULL" )
            {
                # delete the sample rows now that stats have been gathered
            }
            &dodb2file($dbname,"$ddlpath${delim}$deleteSampleUF", $once);
        }
    }
    &dodb2file($dbname,"$ddlpath${delim}$ddl", $once);

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

    if ($appendOn eq "yes"){
        &dodb_conn($dbname,
            "db2 alter table tpcd.nation locksize table $sep \
            db2 alter table tpcd.region locksize table $sep \
            db2 alter table tpcd.customer locksize table $sep \
            db2 alter table tpcd.supplier locksize table $sep \
            db2 alter table tpcd.part locksize table $sep \
            db2 alter table tpcd.partsupp locksize table $sep \
            db2 alter table tpcd.lineitem append on $sep \
            db2 alter table tpcd.orders append on",
            $once);
    }
    else{
        &dodb_conn($dbname,
            "db2 alter table tpcd.nation locksize table $sep \
            db2 alter table tpcd.region locksize table $sep \
            db2 alter table tpcd.customer locksize table $sep \
            db2 alter table tpcd.supplier locksize table $sep \
            db2 alter table tpcd.part locksize table $sep \
            db2 alter table tpcd.partsupp locksize table $sep \
            db2 alter table tpcd.lineitem pctfree 4 $sep \
            db2 alter table tpcd.orders pctfree 4",
            $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
        &loadConfig;
    }

    # 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 eq "NULL") || ( $loadscript eq "" ))
  {
    $src = system("perl genloaduni $qual");
    if ( $src != 0 )
    {
      die "genloaduni failed rc = $src\n";
    }
  }
  else
  {
    &dodb2file_noconn("$ddlpath${delim}$loadscript",$once);
  }
}
elseif (( $mode eq "mln" ) || ( $mode eq "mpp" ))
{
  &outtime("**** Starting the load");
  # call the appropriate dbgen/split/(sort)/load for mln/mpp

  if (( $loadscript eq "NULL") || ( $loadscript eq "" ))
  {
    $src = system("perl genloadmpp $qual");
    if ( $src != 0 )
    {
      die "genloadmpp failed rc = $src\n";
    }
  }
  else
  {
    $src = &dodb2file_noconn("$ddlpath${delim}$loadscript $sf");
    $src = system("call $ddlpath${delim}$loadscript");
  }

  if ( $src != 0 )
  {
    die "doload for $dbname failed rc = $src. Script is
    $ddlpath${delim}$loadscript\n";
  }
}
else
{
  die "TPCD_MODE not set to one of uni, smp, mln or mpp\n";
}

&outtime("**** Load complete");

# verify that the audit directory exists
$filename="$auditDir";
if (-e $filename)
{
  # set up the $auditDir/$dbname.$user.update.pair.num file
  # to start at update pair 1
  $filename="$auditDir${delim}$dbname.$user.update.pair.num";
}
else
{

```

```

  mkdir ("$auditDir", 0775) || die "cannot mkdir $auditDir";
}
print "setting update pair num to 1\n";
system("echo 1 > $filename");
} # 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
    &loadConfig;
  }
  # 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
    &loadConfig;
  }
  # 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 specified 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" || $platform eq "sun" || $platform eq "ptx" )
      {
        print "runstats from $ddlpath${delim}$runstatShort running
        now\n";
        $src = system("db2 -tvf \"$ddlpath${delim}$runstatShort\" >
        \"$auditDir${delim}tools${delim}runstatShort.out\" & ");
        print "rc from runstatshort=$src\n";
      }
      elseif ( $platform eq "nt" )
      {
        system("start db2 -tvf $ddlpath${delim}$runstatShort");

```

```

    }
    else
    {
        print "Don't know how to start in background on $platform
platform\n";
        print "therefore running runstats serially\n";
        &dodb2file($dbname,"$ddlpath${delim}$runstatShort",$once);
    }
    # run the full runstats, or the remainder of what wasn't put into the short
    # runstats file. You should be sure that this runstats will take longer
    # than the short runstats that is running in the background, otherwise
    # setting the config will happen before this is done.
    &dodb2file($dbname,"$ddlpath${delim}$runstats",$once);
    &outtime("*** Runstats completed");
}
} # end build phaose all/load/index/runstats

# add the RI
if ( $addRI ne "NULL" )
{
    &outtime("*** Adding RI constraints started");
    &dodb2file($dbname,"$ddlpath${delim}$addRI",$once);
    &outtime("*** Adding RI constraints completed");
}

#add the AST if it has been requested
if ( $astFile ne "NULL" )
{
    &outtime("*** Adding AST started");
    &dodb2file($dbname,"$ddlpath${delim}$astFile",$once);
    &outtime("*** Adding AST completed");
}

# check tbsp info
&dodb_conn($dbname,"db2 list tablespaces show detail",$once);

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

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

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

# set the db2 environment variables for running the benchmark
if ( $setScript ne "NULL" )
{
    if ( $platform eq "aix" || $platform eq "sun" || $platform eq "ptx")
    {
        $ret=system("$ddlpath${delim}$setScript");
    }
    elsif ( $platform eq "nt" )
    {
        if (($mode eq "uni" ) || ($mode eq "smp" ))
        {
            $ret = system("perl $ddlpath${delim}$setScript");
        }
        else
        {
            $ret = system("rah \"call $ddlpath${delim}$setScript\" ");

```

```

    }
    }
    if ( $ret != 0 )
    {
        die "failure setting runtime db2set parms from $setScript \n";
    }
}
# if logging is enabled, we must take a backup of the database
if ( $log eq "yes" )
{
    &dodb_noconn("db2 update database configuration for $dbname using
LOGRETAIN yes",$all_ln);
    print "\n NOTE: DO NOT RESET THE DATABASE
CONFIGURATION or you will lose logretain\n";
    # force a connection to the database on all nodes to ensure LOGRETAIN
    is
    # set in effect.
    # An error message will print to screen if the logretain is set properly
    # i.e. SQL116N A connection to or activation of database <database
    name>
    # cannot be made.
    # This is expected and the lack of this error message should be seen as an
    # error in the database build.
    &dodb_conn($dbname,"db2 \"select count(*) from tpcd.region\"",$all_ln);

    if ( $qual eq "QUAL" )
    {
        &outtime("*** Starting the backup");

        if (( $mode eq "mln" ) || ( $mode eq "mpp"))
        {
            # must back up catalog node first...assume node 00
            $src=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
            $src=system("db2_all \'}<<-000< db2 backup database $dbname to
$backupdir without prompting\" ");
            if ( $rc != 0 )
            {
                die "backup of remaining nodes failed rc = $rc\n";
            }
        }
        else
        {
            &dodb_noconn("db2 backup database $dbname to
$backupdir",$once);
        }
        &outtime("*** Finished the backup");
    }
    else
    {
        # This is the test database. Clause 3.1.4 states that "the test sponsor is
        # not required to make or have backup copies of the test database;
        however
        # all other mechanisms that guarantee durability of the qualification
        # database must be enabled in the same way for the test database".
        # According to this clause we do need to keep the backup of the
        database.
        &dodb_noconn("db2dart $dbname /CHST /WHAT DBBP
OFF",$all_ln);
    }
}

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

```

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

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

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

$rc = system("perl buildtpcdbatch $qual");
if ( $rc != 0 )
{
    die "buildtpcdbatch failed rc=$rc\n";
}

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${delim}tools${delim}tpcd.runsetup");
    system("perl setupRun");
    # before we stop the database for the final time
    # if we are in the real audit mode then run dbtables and dbcheck before
    # we print out the 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
    if ( $qual eq "QUAL" )
    {
        $verifyType="q";
    }
    else
    {
        $verifyType="t";
    }
    system("perl tablesdb $verifyType");

    &dodb2file($dbname, "$auditDir${delim}tools${delim}first10rows.sql", $once);
}
# stop, restart and activate the database, if necessary

#Create Catalog info
$rc = system("perl catinfo.pl b");

if ( $rc != 0 )
{
    warn "catinfo failed!!! rc = $rc\n";
}

$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");

##### subroutine for setting the load
configuration #####
sub loadConfig{

    &outtime("**** Setting LOAD configuration.");
    if ($loadconfigfile eq "NULL" || $loadDBMconfig 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_in);
        &dodb_noconn("db2 update dbm cfg using sheapthres
$sheapthres", $once);
    }
    else
    {
        &dodb2file_noconn("$ddlpath${delim}$loadconfigfile", $all_in);
        &dodb2file_noconn("$ddlpath${delim}$loadDBMconfig", $once);
    }
    &dodb_noconn("db2 terminate", $once);
    $rc=system("db2stop");
    if ( $rc != 0 )
    {
        die "failure during db2stop after resetting for load config rc = $rc \n";
    }
    $rc=system("db2start");
    if ( $rc != 0 )
    {
        die "failure during db2start rc = $rc \n";
    }
}

1;
```

B.2 tpcd.setup

NOTE: ALL variable definitions must have a comment at the end - haven't got

```
# the getvars script recognizing the uncommented line yet
TPCD_PLATFORM=nt # aix, nt, sun ....
TPCD_VERSION=2 # 1 or 2 (Version of tpcd). Default 1
TPCD_DBNAME=TPCD # name to create database under
TPCD_WORKLOAD=H # TPC version (R for TPCR, H for TPCB)
TPCD_AUDIT_DIR=Z:\tpch # 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=mpp # uni/smp/mln/mpp
TPCD_PHYS_NODE=16 # number of physical nodes
TPCD_LN_PER_PN=1 # number of logical nodes per physical
node
TPCD_SF=300 # size of the database (1=1GB,...) to
# get test size databases use:
# 0.012 = 12MB
# 0.1 = 100MB
TPCD_BUILD_STAGE=ALL # where to start the build -
currently the
# following is possible:
# ALL - do everything (create,load,
# index,stats,config) (Default)
# CRTTBSP - start after create db and
# config setting. Start right at
# create tbps
# LOAD - start from the load of the tables
# INDEX - start from the index creation
# (NOTE if earlyindex is specified,
# then this will do the create index
# followed by the load...)
# RUNSTATS - start from the runstats
# (NOTE Do not use this option if
# distribution stats are gathered
# as part of the load, this will
# start after the load and indices
# have been created.
# CONFIG - start from the setting up of
# the benchmark runs config setup
#
TPCD_DBPATH=C: # Path for database (defaults to home)
TPCD_DDLPATH=Z:\tpch\cpq_ddl # Path for all ddl files and
customized
# scripts (load script), config files,etc
TPCD_BUFFERPOOL_DEF=cpq_buffer_pools # Name of file with
bufferpool definitions
# and sizes
TPCD_NODEGROUP_DEF=NULL # Name of file in ddlpath with
nodegroup
# definitions
TPCD_EXPLAIN_DDL=EXPLAIN.DDL # File with DDL for
explains statments
# if this is NULL then uses the default
# and puts it in USERSPACE1 across all
# nodes...nt 1TB found it was faster if
# just in a single node nodegroup
TPCD_TBSP_DDL=cpq_create_tablespace # ddl file for tablespaces
TPCD_DDL=cpq_create_tables # ddl file for tables
TPCD_QUAL_TBSP_DDL=dss.ddl12MB.tbps.nt.qual # ddl file for
tablespaces for qual
TPCD_QUAL_DDL=dss.ddl12MB.nt.qual # ddl file for qualification
database
# tablespaces and tables should be identical
# to regular ddl except container names
TPCD_INDEXDDL=cpq_tpch_indexes # ddl file for indexes
TPCD_EXTRAINDEX=no # no = no extra indexes
# filename = If you want to create some
# indices before
# the load, and some indices after, then
```

```
# use this additional file to specify the
TPCD_ADD_RI=NULL # file name that contains any RI
# constraints to add after index creation
# set to NULL (default) if unused
# indices to create after the load.
TPCD_AST=NULL # file name that contains complete AST
TPCD_RUNSTATS=cpq_runstats # ddl file for runstats. If you have
# created indices before the load (ie
# TPCD_EARLYINDEX=yes), have specified to
# gather stats on the load command (either
# through your own load script or by using
# TPCD_LOADSTATS=yes, AND you have
# specified a file for TPCD_EXTRAINDEX
# then this runstats file should include
# the runstats commands specifically for
# the extra indices.
TPCD_RUNSTATSHORT=NULL # ddl file for short runstats that
are
# run in the background while the
# TPCD_RUNSTATS are run in the foreground
# of the build. If this is used, then
# TPCD_RUNSTATS should have the runstats
# command for lineitem and
# TPCD_RUNSTATSHORT should have runstats
# commands for all other tables.
TPCD_DBGEN=Z:\tpch\appendix.v2\dbgen # path name to data generation
code
# Parameters used to specify source of
# data for load scripts
TPCD_INPUT=flatfiles # NULL - use dbgen generated data OR
# path name - to the pre-generated
# flat files
# /gwl/dss/12MB - path for pregen'd 12MB
# /gwl/dss/100MB - path for pregen'd 100MB
#
TPCD_QUAL_INPUT=NULL # NULL - use dbgen generated
data OR
# path name - to the pre-generated
# flat files
TPCD_TAILOR_DIR=C:\tailor # path name for the directory used
to
# generate split specific config files
# only used for partitioned environment
TPCD_EARLYINDEX=no # create indexes before the load
# LOAD specific parameters follow:
TPCD_LOAD_DB2SET_SCRIPT=cpq_db2set_build.bat # Script that
contains the db2set commands
# for the load process Use NULL if not
# specified
TPCD_LOAD_CONFIGFILE=cpq.dbcfg.load # 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=cpq.dbmcfg.load # 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_QUAL_CONFIGFILE=loadcfg.sql # config file with specific
database config
# parms for the load/index/runstats part
# of the build for qualification db.
# set to NULL if use defaults
TPCD_LOAD_DBM_QUAL_CONFIGFILE=loaddbmcfg.sql # config file
with specific
```

```

# database manager config parts for the
# load/index/runstats part of the build.
# set to NULL if use defaults

TPCD_LOADSTATS=no          # gather statistics during load
# ignored if EARLYINDEX is not set
# due to runstats limitation
TPCD_TEMP=C:\temp          # path for LOAD temp files
# defaults to /u/<instance>/sqllib/tmp
# used in load script only
TPCD_SORTBUF=4096          # sortbuf size for LOAD
# used in load script only
TPCD_LOAD_PARALLELISM=0    # degree of parallelism to use
on load
# 0 = use the "intelligent default" that
# the load will chose at run time
# used in load script only
TPCD_COPY_DIR=NULL         # directory where copy image is
created
# on load command CURRENTLY UNUSED
# used in load script only
TPCD_FASTPARSE=yes         # use fastparse on load
# used in load script only

# Backup and logfile specific parameters follow:
TPCD_BACKUP_DIR=z:\db2_tpch\backupdir # directory where backup
files are placed
TPCD_LOGPRIMARY=NULL       # NULL/value = how many
primary log files
# to configure. If NULL is specified then
# the default is not changed.
TPCD_LOGFILSIZ=1000        # NULL/value = how 4KB pages to
use for
# logfilsiz db cfg parameter. If NULL is
# specified then the default is not changed
TPCD_LOGSECOND=NULL        # NULL/value = how many
secondary log files
# to configure. If NULL is specified then
# the default is not changed.
TPCD_LOG_DIR=NULL          # directory where log files stored..
# NULL leaves them in the dbpath
TPCD_LOG_QUAL_DIR=NULL     # directory where qual log files
stored
# NULL leaves them in the dbpath
TPCD_LOG=no                # yes/no - whether to turn LOG_RETAIN
on
# i.e. are backups needed to be taken

# CONFIG specific parameters
TPCD_DB2SET_SCRIPT=cpq_db2set_run.bat # Script that contains the
db2set commands
# for the benchmark run. Use NULL if not
# specified
TPCD_CONFIGFILE=cpq.dbcfg.run # name of configuration file in
ddl path
# that will be used for the benchmark run
TPCD_DBM_CONFIG=cpq.dbmcfg.run # name of config file for
database manager
# cfg parms
TPCD_QUALCONFIGFILE=NULL    # name of database cfg file in
ddl path
# for qualification database
TPCD_DBM_QUALCONFIG=NULL    # name of config file for
database
# manager cfg parms

TPCD_MACHINE=small         # set to NULL if using load config
file
# big/medium/small size of machine used to

```

```

# determine buffpage, sortheap,sheapthres
# and ioservers parms for load, create
# index and runstats
# NOTE that this parameter is ignored if
# a TPCD_LOAD_CONFIGFILE

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

TPCD_ACTIVATE=yes          # activate the database upon build
# completion

# run specific parameters
TPCD_AUDIT=yes             # no/yes
# no - don't set up qualification db stuff
# yes - set up qualification db queries
# - build the update function tables
# and data before we get into the
# timing of the creation of the
# tables and the load.
TPCD_TMP_DIR=C:\temp       # place to put temp working files

TPCD_SHARED_TEMP_FULL_PATHNAME=NULL # just added
TPCD_QUERY_TEMPLATE_DIR=standard.V2 # subdirectory in
AUDIT_DIR/queries
# to use as the source of the query
# templates. Currently there are
# v2 ones and pe ones. You can make
# your own directory following the same
# form as in the v2 directory using
# any variant you wish
TPCD_QUAL_DBNAME=TPCD      # name of qualification
database
TPCD_NUMSTREAM=6           # number of streams for the
throughput test
TPCD_FLATFILES=c:\uf_flatfiles # where to generate flat files
# TPCD_FLATFILES=t:\uf_flatfiles.2 # where to generate flat files
# for update functions
TPCD_STAGING_TABLE_DDL=createUfTbls # script that contains the
ddl for creating
# the staging tables if they are used for
# the update functions
TPCD_PRELOAD_STAGING_TABLE_SCRIPT=loadSampleUFData # file
that contains the sql for preloading
# and gathering stats on sample UF data
# Note that the data used is sample data
# and is not data from any of the applied
# update pairs
TPCD_DELETE_STAGING_TABLE_SQL=delSampleUFData.sql # file
that contains the sql for deleting
# the preloaded data from the staging
# tables
TPCD_UPDATE_IMPORT=false   # true = use import for the
staging tables
# for UNI/SMP mode only (code change in
# tpcdbatch) (if not uni mode then must
# change load_update)
# false = use load for staging tables
# The default is false if not set.
# NOTE that this parm is only for UNI/SMP
# it is not for multi node invocation

TPCD_SPLIT_UPDATES=256     # number of chunks to split the
update
# function into.

```

```

TPCD_CONCURRENT_INSERTS=32      # number of insert chunks
that are run
                                # concurrently. TPCD_SPLIT_UPDATES
                                # should be evenly divisible by this number
TPCD_CONCURRENT_INSERTS_LOAD=32  # number of insert
chunks that are loaded
                                # concurrently. TPCD_SPLIT_UPDATES should
                                # be evenly divisible by this number.
                                # this controls the load portion of the
                                # insert routine for partitioned databases
TPCD_SPLIT_DELETES=256          # number of portions to split the
delete
                                # function into.
                                # this variable is only valid in UNI/SMP
                                # mode.
TPCD_CONCURRENT_DELETES=32      # number of delete chunks
that are run
                                # concurrently. This number should be
                                # evenly divisible by TPCD_SPLIT_DELETES
TPCD_GEN_UPDATEPAIRS=30         # 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=3          # timing interval for
vmstats/iostats
TPCD_STATS_THRU_INT=60         # timing interval for
vmstats/iostats for
                                # throughput run
TPCD_GATHER_STATS=off          # on/off - only implement for
AIX yet
                                # on = gather statistics around power
                                # test run (vmstat,iostat,netstat)
                                # off = no stats gathered during power run

TPCD_UFTEMP=UFTEMP             # base name of tablespace(s) where
the
                                # staging tables for the update functions
                                # are created
                                # this name will be used as the
                                # basename for the tablespaces...eg
                                # UFTEMP1 UFTEMP2 ....
TPCD_HAVECOMPILER=yes          # rebuild tpcdbatch executable
                                # yes/no
TPCD_SLEEP=5                   # ?
TPCD_INLISTMAX=default          # max num of keys to delete at a
time
                                # for UF2, use "default" for default.
TPCD_LOAD_SCRIPT=LOAD.bat       # script to run for loading tables
                                # in TPCD_DDLPATH directory under mln/mpp
                                # leave as NULL if using default genloaduni
TPCD_LOAD_SCRIPT_QUAL=NULL     # script to run for loading
tables in
                                # TPCD_DDLPATH directory under mln/mpp
                                # for QUAL db
TPCD_ROOTPRIV=no               # do you have root privileges to be
able

```

```

                                # get values of things like schedtune
                                # and vmtune (currently on AIX only)
                                # acid test specific information
TPCD_DB2LOG=z:\BIGBLUE\BLUE    # directory where the
db2diag.log can
                                # be found for the durability tests
TPCD_APPEND_ON=no              # set to no if the cluster indexes are
used

```

B.3 cpq.dbmcfg.load

```

update database manager configuration using
svcename DB2cBLUE
cpuspeed 6.297923e-007
comm_bandwidth 10
sheapthres 320000
agent_stack_sz 16
aslheapsz 15
rqrioblk 32767
intra_parallel yes
max_querydegree -1
maxagents 1500
fcm_num_rqb 7700
num_poolagents 64
num_initagents 4
diaglevel 3;

```

```
get database manager configuration;
```

```
--connect reset;
```

B.4 cpq.dbcfg.load

```

update database configuration for tpcd using
NUM_FREQVALUES 0
--NUM_QUANTILES 300
NUM_QUANTILES 600
buffpage 20000
catalogcache_sz 386
chnpggs_thresh 40
dbheap 6654
locklist 16384
logbufsz 512
logfilsiz 4094
logprimary 100
logsecond 25
newlogpath \\PhysicalDrive20
maxappls 40
maxlocks 75
mincommit 1
num_iocleaners 16
num_ioservers 16
pckcachesz 320
softmax 100
sortheap 32000
stat_heap_sz 4384
stmthep 4096
util_heap_sz 80000
applheapsz 768
app_ctl_heap_sz 1024
dft_degree 10;

```

```
get database configuration for tpcd;
```

```
--connect reset;
```

B.5 cpq_db2set_build.bat

```
db2set DB2_STRIPED_CONTAINERS=ON
db2set DB2_PARALLEL_IO=*
db2set DB2BPVARS=z:\tpch\cpq_dll\cpq_scattered-read
db2set DB2_LIKE_VARCHAR=y,y
db2set DB2_HASH_JOIN=Y
db2set DB2_RR_TO_RS=ON
db2set DB2_ANTIJOIN=Y
db2set DB2OPTIONS="-t -v +c"
db2set DB2NTNOCACHE=ON
db2set DB2MEMMAXFREE=137438953472,20
```

B.6 cpq_create_tablespace

```
create regular tablespace LINEITEM_TABLE
pagesize 32k
managed by database
using (device '\\.\PhysicalDrive0' 147350,
      device '\\.\PhysicalDrive5' 147350,
      device '\\.\PhysicalDrive10' 147350,
      device '\\.\PhysicalDrive15' 147350)
bufferpool BP16K
extentsize 16
prefetchsize 768
overhead 6
transferrate 0.32;
```

```
create regular tablespace LINEITEM_INDEXES
pagesize 32k
managed by database
using (device '\\.\PhysicalDrive1' 16250,
      device '\\.\PhysicalDrive6' 16250,
      device '\\.\PhysicalDrive11' 16250,
      device '\\.\PhysicalDrive16' 16250)
bufferpool BP16K
extentsize 16
prefetchsize 768
overhead 6
transferrate 0.32;
```

```
create temporary tablespace TEMP_TABLE1
pagesize 32k
managed by database
using (device '\\.\PhysicalDrive2' 262000,
      device '\\.\PhysicalDrive7' 262000,
      device '\\.\PhysicalDrive12' 262000,
      device '\\.\PhysicalDrive17' 262000)
bufferpool BP16K
extentsize 16
prefetchsize 768
overhead 6
transferrate 0.32;
```

```
create regular tablespace OTHER_STUFF
pagesize 32k
managed by database
using (device '\\.\PhysicalDrive3' 65250,
      device '\\.\PhysicalDrive8' 65250,
      device '\\.\PhysicalDrive13' 65250,
      device '\\.\PhysicalDrive18' 65250)
bufferpool BP16K
extentsize 16
prefetchsize 768
overhead 6
transferrate 0.32;
```

```
create regular tablespace OTHER_INDEXES
```

```
pagesize 32k
managed by database
using (device '\\.\PhysicalDrive4' 12150,
      device '\\.\PhysicalDrive9' 12150,
      device '\\.\PhysicalDrive14' 12150,
      device '\\.\PhysicalDrive19' 12150)
bufferpool BP16K
extentsize 16
prefetchsize 768
overhead 6
transferrate 0.32;
```

```
create nodegroup SMALLTABLE on node(0);
create regular tablespace SMALLTABLE in nodegroup SMALLTABLE
managed by system
USING ( 'z:\small_tables') on node (0);
```

```
--drop tablespace USERSPACE1;
```

```
--drop tablespace TEMPSPACE1;
```

```
commit work;
```

B.7 cpq_create_tables

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

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

```
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_STUFF
INDEX IN OTHER_INDEXES;
```

```
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_STUFF
INDEX IN OTHER_INDEXES;
```

```
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_STUFF
```

```

INDEX IN OTHER_INDEXES;

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_STUFF
INDEX IN OTHER_INDEXES;

CREATE TABLE TPCD.ORDERS ( 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)
IN OTHER_STUFF
INDEX IN OTHER_INDEXES;

CREATE TABLE TPCD.LINEITEM ( 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)
IN LINEITEM_TABLE
INDEX IN LINEITEM_INDEXES;

COMMIT WORK;

```

B.8 Load

```

CONNECT TO TPCD;

LOAD FROM t:\flatfiles\supplier.tbl.1.pmap,
t:\flatfiles\supplier.tbl.2.pmap,
t:\flatfiles\supplier.tbl.3.pmap,
t:\flatfiles\supplier.tbl.4.pmap,
t:\flatfiles\supplier.tbl.5.pmap,
t:\flatfiles\supplier.tbl.6.pmap,
t:\flatfiles\supplier.tbl.7.pmap,
t:\flatfiles\supplier.tbl.8.pmap,
t:\flatfiles\supplier.tbl.9.pmap,
t:\flatfiles\supplier.tbl.10.pmap OF DEL
MODIFIED BY COLDEL| FASTPARSE
SAVECOUNT 100000
MESSAGES t:\temp\msgs\supplier.msg
REPLACE INTO TPCD.SUPPLIER
STATISTICS NO

```

```

NONRECOVERABLE
CPU_PARALLELISM 16;

LOAD FROM t:\flatfiles\part.tbl.1.pmap,
t:\flatfiles\part.tbl.2.pmap,
t:\flatfiles\part.tbl.3.pmap,
t:\flatfiles\part.tbl.4.pmap,
t:\flatfiles\part.tbl.5.pmap,
t:\flatfiles\part.tbl.6.pmap,
t:\flatfiles\part.tbl.7.pmap,
t:\flatfiles\part.tbl.8.pmap,
t:\flatfiles\part.tbl.9.pmap,
t:\flatfiles\part.tbl.10.pmap OF DEL
MODIFIED BY COLDEL| FASTPARSE
SAVECOUNT 100000
MESSAGES t:\temp\msgs\part.msg
REPLACE INTO TPCD.PART
STATISTICS NO
NONRECOVERABLE
CPU_PARALLELISM 16;

LOAD FROM t:\flatfiles\customer.tbl.1.pmap,
t:\flatfiles\customer.tbl.2.pmap,
t:\flatfiles\customer.tbl.3.pmap,
t:\flatfiles\customer.tbl.4.pmap,
t:\flatfiles\customer.tbl.5.pmap,
t:\flatfiles\customer.tbl.6.pmap,
t:\flatfiles\customer.tbl.7.pmap,
t:\flatfiles\customer.tbl.8.pmap,
t:\flatfiles\customer.tbl.9.pmap,
t:\flatfiles\customer.tbl.10.pmap OF DEL
MODIFIED BY COLDEL| FASTPARSE
SAVECOUNT 100000
MESSAGES t:\temp\msgs\customer.msg
REPLACE INTO TPCD.CUSTOMER
STATISTICS NO
NONRECOVERABLE
CPU_PARALLELISM 16;

LOAD FROM t:\flatfiles\partsupp.tbl.1.pmap,
t:\flatfiles\partsupp.tbl.2.pmap,
t:\flatfiles\partsupp.tbl.3.pmap,
t:\flatfiles\partsupp.tbl.4.pmap,
t:\flatfiles\partsupp.tbl.5.pmap,
t:\flatfiles\partsupp.tbl.6.pmap,
t:\flatfiles\partsupp.tbl.7.pmap,
t:\flatfiles\partsupp.tbl.8.pmap,
t:\flatfiles\partsupp.tbl.9.pmap,
t:\flatfiles\partsupp.tbl.10.pmap,
t:\flatfiles\partsupp.tbl.11.pmap,
t:\flatfiles\partsupp.tbl.12.pmap,
t:\flatfiles\partsupp.tbl.13.pmap,
t:\flatfiles\partsupp.tbl.14.pmap,
t:\flatfiles\partsupp.tbl.15.pmap,
t:\flatfiles\partsupp.tbl.16.pmap,
t:\flatfiles\partsupp.tbl.17.pmap,
t:\flatfiles\partsupp.tbl.18.pmap,
t:\flatfiles\partsupp.tbl.19.pmap,
t:\flatfiles\partsupp.tbl.20.pmap OF DEL
MODIFIED BY COLDEL| FASTPARSE
SAVECOUNT 100000
MESSAGES t:\temp\msgs\partsupp.msg
REPLACE INTO TPCD.PARTSUPP
STATISTICS NO
NONRECOVERABLE
CPU_PARALLELISM 16;

LOAD FROM t:\flatfiles\orders.tbl.1.pmap,
t:\flatfiles\orders.tbl.2.pmap,

```

```
t:\flatfiles\orders.tbl.3.pmap,
t:\flatfiles\orders.tbl.4.pmap,
t:\flatfiles\orders.tbl.5.pmap,
t:\flatfiles\orders.tbl.6.pmap,
t:\flatfiles\orders.tbl.7.pmap,
t:\flatfiles\orders.tbl.8.pmap,
t:\flatfiles\orders.tbl.9.pmap,
t:\flatfiles\orders.tbl.10.pmap,
t:\flatfiles\orders.tbl.11.pmap,
t:\flatfiles\orders.tbl.12.pmap,
t:\flatfiles\orders.tbl.13.pmap,
t:\flatfiles\orders.tbl.14.pmap,
t:\flatfiles\orders.tbl.15.pmap,
t:\flatfiles\orders.tbl.16.pmap,
t:\flatfiles\orders.tbl.17.pmap,
t:\flatfiles\orders.tbl.18.pmap,
t:\flatfiles\orders.tbl.19.pmap,
t:\flatfiles\orders.tbl.20.pmap OF DEL
MODIFIED BY COLDEL| FASTPARSE
SAVECOUNT 100000
MESSAGES t:\temp\msgs\orders.msg
REPLACE INTO TPCD.ORDERS
STATISTICS NO
NONRECOVERABLE
CPU_PARALLELISM 16;
```

```
LOAD FROM t:\flatfiles\lineitem.tbl.1.pmap,
t:\flatfiles\lineitem.tbl.2.pmap,
t:\flatfiles\lineitem.tbl.3.pmap,
t:\flatfiles\lineitem.tbl.4.pmap,
t:\flatfiles\lineitem.tbl.5.pmap,
t:\flatfiles\lineitem.tbl.6.pmap,
t:\flatfiles\lineitem.tbl.7.pmap,
t:\flatfiles\lineitem.tbl.8.pmap,
t:\flatfiles\lineitem.tbl.9.pmap,
t:\flatfiles\lineitem.tbl.10.pmap,
t:\flatfiles\lineitem.tbl.11.pmap,
t:\flatfiles\lineitem.tbl.12.pmap,
t:\flatfiles\lineitem.tbl.13.pmap,
t:\flatfiles\lineitem.tbl.14.pmap,
t:\flatfiles\lineitem.tbl.15.pmap,
t:\flatfiles\lineitem.tbl.16.pmap,
t:\flatfiles\lineitem.tbl.17.pmap,
t:\flatfiles\lineitem.tbl.18.pmap,
t:\flatfiles\lineitem.tbl.19.pmap,
t:\flatfiles\lineitem.tbl.20.pmap,
t:\flatfiles\lineitem.tbl.21.pmap,
t:\flatfiles\lineitem.tbl.22.pmap,
t:\flatfiles\lineitem.tbl.23.pmap,
t:\flatfiles\lineitem.tbl.24.pmap,
t:\flatfiles\lineitem.tbl.25.pmap,
t:\flatfiles\lineitem.tbl.26.pmap,
t:\flatfiles\lineitem.tbl.27.pmap,
t:\flatfiles\lineitem.tbl.28.pmap,
t:\flatfiles\lineitem.tbl.29.pmap,
t:\flatfiles\lineitem.tbl.30.pmap,
t:\flatfiles\lineitem.tbl.31.pmap,
t:\flatfiles\lineitem.tbl.32.pmap,
t:\flatfiles\lineitem.tbl.33.pmap,
t:\flatfiles\lineitem.tbl.34.pmap,
t:\flatfiles\lineitem.tbl.35.pmap,
t:\flatfiles\lineitem.tbl.36.pmap,
t:\flatfiles\lineitem.tbl.37.pmap,
t:\flatfiles\lineitem.tbl.38.pmap,
t:\flatfiles\lineitem.tbl.39.pmap,
t:\flatfiles\lineitem.tbl.40.pmap,
t:\flatfiles\lineitem.tbl.41.pmap,
t:\flatfiles\lineitem.tbl.42.pmap,
t:\flatfiles\lineitem.tbl.43.pmap,
t:\flatfiles\lineitem.tbl.44.pmap,
```

```
t:\flatfiles\lineitem.tbl.45.pmap,
t:\flatfiles\lineitem.tbl.46.pmap,
t:\flatfiles\lineitem.tbl.47.pmap,
t:\flatfiles\lineitem.tbl.48.pmap,
t:\flatfiles\lineitem.tbl.49.pmap,
t:\flatfiles\lineitem.tbl.50.pmap,
t:\flatfiles\lineitem.tbl.51.pmap,
t:\flatfiles\lineitem.tbl.52.pmap,
t:\flatfiles\lineitem.tbl.53.pmap,
t:\flatfiles\lineitem.tbl.54.pmap,
t:\flatfiles\lineitem.tbl.55.pmap,
t:\flatfiles\lineitem.tbl.56.pmap,
t:\flatfiles\lineitem.tbl.57.pmap,
t:\flatfiles\lineitem.tbl.58.pmap,
t:\flatfiles\lineitem.tbl.59.pmap,
t:\flatfiles\lineitem.tbl.60.pmap,
t:\flatfiles\lineitem.tbl.61.pmap,
t:\flatfiles\lineitem.tbl.62.pmap,
t:\flatfiles\lineitem.tbl.63.pmap,
t:\flatfiles\lineitem.tbl.64.pmap,
t:\flatfiles\lineitem.tbl.65.pmap,
t:\flatfiles\lineitem.tbl.66.pmap,
t:\flatfiles\lineitem.tbl.67.pmap,
t:\flatfiles\lineitem.tbl.68.pmap,
t:\flatfiles\lineitem.tbl.69.pmap,
t:\flatfiles\lineitem.tbl.70.pmap,
t:\flatfiles\lineitem.tbl.71.pmap,
t:\flatfiles\lineitem.tbl.72.pmap,
t:\flatfiles\lineitem.tbl.73.pmap,
t:\flatfiles\lineitem.tbl.74.pmap,
t:\flatfiles\lineitem.tbl.75.pmap,
t:\flatfiles\lineitem.tbl.76.pmap,
t:\flatfiles\lineitem.tbl.77.pmap,
t:\flatfiles\lineitem.tbl.78.pmap,
t:\flatfiles\lineitem.tbl.79.pmap,
t:\flatfiles\lineitem.tbl.80.pmap OF DEL
MODIFIED BY COLDEL| FASTPARSE
SAVECOUNT 100000
MESSAGES t:\temp\msgs\lineitem.msg
REPLACE INTO TPCD.LINEITEM
STATISTICS NO
NONRECOVERABLE
CPU_PARALLELISM 16;
```

```
COMMIT WORK;
CONNECT RESET;
TERMINATE;
```

B.9 cpq_tpch_indexes

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

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

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

values(current timestamp);
CREATE UNIQUE INDEX "TPCD"."S_SK" ON "TPCD"."SUPPLIER"
```

```

        ("S_SUPPKEY" ASC)
        PCTFREE 0 ;
commit work;

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

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

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

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

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

values(current timestamp);
CREATE UNIQUE INDEX "TPCD"."O_OK" ON "TPCD"."ORDERS"
        ("O_ORDERKEY" ASC)
        PCTFREE 5 ;
commit work;

values(current timestamp);
CREATE INDEX "TPCD"."O_CK" ON "TPCD"."ORDERS"
        ("O_CUSTKEY" ASC)
        PCTFREE 5 ;
commit work;

values(current timestamp);
CREATE INDEX "TPCD"."O_OD" ON "TPCD"."ORDERS"
        ("O_ORDERDATE" ASC)
        CLUSTER
        PCTFREE 5 ;
commit work;

values(current timestamp);
CREATE INDEX "TPCD"."L_OK" ON "TPCD"."LINEITEM"
        ("L_ORDERKEY" ASC)
        PCTFREE 5 ;
commit work;

values(current timestamp);
CREATE INDEX "TPCD"."L_SD" ON "TPCD"."LINEITEM"
        ("L_SHIPDATE" ASC)
        CLUSTER
        PCTFREE 5 ;
commit work;
values(current timestamp);

```

B.10 cpq_runstats

```

values(current timestamp);
RUNSTATS ON TABLE TPCD.NATION WITH DISTRIBUTION AND
DETAILED INDEXES ALL;
commit;
values(current timestamp);
RUNSTATS ON TABLE TPCD.REGION WITH DISTRIBUTION AND
DETAILED INDEXES ALL;
commit;
values(current timestamp);
RUNSTATS ON TABLE TPCD.SUPPLIER WITH DISTRIBUTION AND
DETAILED INDEXES ALL;
commit;
values(current timestamp);
RUNSTATS ON TABLE TPCD.PART WITH DISTRIBUTION AND
DETAILED INDEXES ALL;
commit;
values(current timestamp);
RUNSTATS ON TABLE TPCD.PARTSUPP WITH DISTRIBUTION
AND DETAILED INDEXES ALL;
commit;
values(current timestamp);
RUNSTATS ON TABLE TPCD.CUSTOMER WITH DISTRIBUTION
AND DETAILED INDEXES ALL;
commit;
values(current timestamp);
RUNSTATS ON TABLE TPCD.ORDERS WITH DISTRIBUTION AND
DETAILED INDEXES ALL;
commit;
values(current timestamp);
RUNSTATS ON TABLE TPCD.LINEITEM WITH DISTRIBUTION AND
DETAILED INDEXES ALL;
COMMIT WORK;
values(current timestamp);

```

B.11 cpq.dbmcfg.run

```

update database manager configuration using
agentpri 2
numdb 1
svcname DB2cBLUE
cpuspeed 6.297923e-007
comm_bandwidth 10
sheapthres 430000
agent_stack_sz 16
aslheapsz 15
rqrioblk 32767
intra_parallel yes
max_querydegree -1
maxagents 160
num_poolagents 64
num_initagents 4
fcm_num_rqb 1536
fcm_num_connect 1280
fcm_num_anchors 128
fcm_num_buffers 768
diagpath z:\bigblue\blue
diaglevel 3
notifylevel 2;

```

get database manager configuration;

B.12 cpq.dbcfg.run

```

update database configuration for tpcd using
NUM_QUANTILES 600
buffpage 55000
catalogcache_sz 386

```

```
dbheap 20000
locklist 384
logbufsz 2048
logfilsiz 8192
logprimary 100
logsecond 25
newlogpath \\.\PhysicalDrive20
maxappls 40
avg_appls 1
maxfilop 1024
maxlocks 15
mincommit 1
num_iocleaners 16
num_ioservers 16
DLCHKTIME 5000
pckcachesz 640
softmax 800
sortheap 22500
stat_heap_sz 2048
stmtheap 2048
util_heap_sz 20000
applheapsz 768
app_ctl_heap_sz 1024
dft_degree 4
dft_queryopt 7
chngpgs_thresh 80;
```

get database configuration for tpcd;

--connect reset;

B.13 cpq_db2set_run.bat

```
db2set DB2_STRIPED_CONTAINERS=ON
db2set DB2_PARALLEL_IO=*
db2set DB2BPVARS=z:\tpch\cpq_dll\cpq_scattered-read
db2set DB2_LIKE_VARCHAR=y,y
db2set DB2_HASH_JOIN=Y
db2set DB2_RR_TO_RS=ON
db2set DB2_ANTIJOIN=Y
db2set DB2OPTIONS="-t -v +c"
db2set DB2NTNOCACHE=ON
db2set DB2_BINSORT=N
db2set DB2MEMMAXFREE=137438953472,20
```

B.14 cpq_buffer_pools

```
--connect to tpcd;
alter bufferpool ibmdefaultbp size 1000;
create bufferpool BP16K size -1 pagesize 32k;
-- create bufferpool BP32K size 40000 pagesize 32k;
commit;
--connect reset;
```

B.15 createuftbls

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.16 Prepdrvvr.bat

```
db2 connect to TPCD;
db2 prep driver.sqc BINDFILE OPTLEVEL 1 ISOLATION RS
MESSAGES prep.msg NOLINEMACRO;
db2 bind driver.bnd GRANT PUBLIC MESSAGES bnd.msg;
db2 connect reset;
db2 terminate;
```

B.17 build.c

```

/* @(#)build.c 2.1.8.1 */
/* Sccsid:  @(#)build.c 9.1.1.17 11/15/95 12:52:28 */
/* stuff related to the customer table */
#include <stdio.h>
#include <string.h>
#ifdef VMS
#include <sys/types.h>
#endif
#if defined(SUN)
#include <unistd.h>
#endif
#include <math.h>

#include "dss.h"
#include "dsstypes.h"
#include "bcd2.h"
#ifdef ADHOC
#include "adhoc.h"

extern adhoc_t adhocs[];
// extern int Numnodes;
#endif /* ADHOC */

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

#define MAX_KEY_LEN 32          // guojian@0725
// #define NumNodes 04          // guojian@0726
#define LARGE_TABLE 1          // DJD
#define SMALL_TABLE 0          // DJD
/*****
*****/

#define LEAP_ADJ(yr, mnth) \
((LEAP(yr) && (mnth) >= 2) ? 1 : 0)
#define JDAY_BASE 8035 /* start from 1/1/70 a la unix */
#define JMNTH_BASE (-70 * 12) /* start from 1/1/70 a la unix */
#define JDAY(date) ((date) - STARTDATE + JDAY_BASE + 1)
#define PART_SUPP_BRIDGE(tgt, p, s) \
{ \
    long tot_scnt = tdefs[SUPP].base * scale; \
    tgt = (p + s * (tot_scnt / SUPP_PER_PART + \
        (long) ((p - 1) / tot_scnt))) % tot_scnt + 1; \
}
#define RPRICE_BRIDGE(tgt, p) tgt = rpb_routine(p)
#define V_STR(avg, sd, tgt) a_rnd((int)(avg * V_STR_LOW), \
(int)(avg * V_STR_HGH), sd, tgt)
#define TEXT(avg, sd, tgt) \
dbg_text(tgt, (int)(avg * V_STR_LOW), (int)(avg * V_STR_HGH), sd)
static void gen_phone_PROTO((long ind, char *target, long seed));

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

int isltme (unsigned _int64, struct sqlupi*, int);          // guojian@0723

/*guojian@0725 stuffs for calling sqlgrpn */
unsigned short  countrycode = 1;  // country code
unsigned short  codepage = 1252; // code page
unsigned short  chklvl = 3;      // check level
short          sDataFormat = 0x0; // data format
/* guojian@0725 end */

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

```

```

long
rpb_routine(long p)
{
    long price;

    price = 90000;
    price += (p/10) % 20001;    /* limit contribution to $200 */
    price += (p % 1000) * 100;

    return(price);
}

static void
gen_phone(long ind, char *target, long seed)
{
    long    acode,
            exchg,
            number;

    RANDOM(acode, 100, 999, seed);
    RANDOM(exchg, 100, 999, seed);
    RANDOM(number, 1000, 9999, seed);

    sprintf(target, "%02d", 10 + (ind % NATIONS_MAX));
    sprintf(target + 3, "%03d", acode);
    sprintf(target + 7, "%03d", exchg);
    sprintf(target + 11, "%04d", number);
    target[2] = target[6] = target[10] = '-';

    return;
}

long
mk_cust(_int64 n_cust, customer_t *c)
{
    {
        long rc = 0;
        long i;

        c->custkey = n_cust;
        sprintf(c->name, C_NAME_FMT, C_NAME_TAG, n_cust);
        c->alen = V_STR(C_ADDR_LEN, C_ADDR_SD, c->address);
        RANDOM(i, 0, (nations.count - 1), C_NTRG_SD);
        c->nation_code = i;
        gen_phone(i, c->phone, (long)C_PHNE_SD);
        RANDOM(c->acctbal, C_ABAL_MIN, C_ABAL_MAX,
C_ABAL_SD);
        pick_str(&c_mseg_set, C_MSEG_SD, c->mktsegment);
        c->cflen = TEXT(C_CMNT_LEN, C_CMNT_SD, c->comment);

        /* guojian@010908 if the record belongs to this node, return 0,
        otherwise, return 1 */
        if (!isltme ((unsigned _int64)c->custkey, part_info_for_customer,
SMALL_TABLE)) {
            rc = 1;
        }
        /* guojian@010908 */

        return(rc);    //Guojian.
    }

    /*
    * generate the numbered order and its associated lineitems
    */
    //mk_sparse (long i, DSS_HUGE *ok, long seq)
    void
    mk_sparse (DSS_HUGE i, DSS_HUGE *ok, long seq)
    {

```

```

#ifndef SUPPORT_64BITS
    if (scale < MAX_32B_SCALE)
#endif
    ez_sparse(i, ok, seq);
#ifndef SUPPORT_64BITS
    else
        hd_sparse(i, ok, seq);
#endif
    return;
}

/*
 * the "simple" version of mk_sparse, used on systems with 64b support
 * and on all systems at SF <= 300G where 32b support is sufficient
 */
//ez_sparse(long i, DSS_HUGE *ok, long seq)
void
ez_sparse(DSS_HUGE i, DSS_HUGE *ok, long seq)
{
    long low_bits;

    LONG2HUGE(i, ok);
    //low_bits = (long)i & ((1 << SPARSE_KEEP) - 1);
    low_bits = (DSS_HUGE)i & ((1 << SPARSE_KEEP) - 1);
    *ok = *ok >> SPARSE_KEEP;
    *ok = *ok << SPARSE_BITS;
    *ok += seq;
    *ok = *ok << SPARSE_KEEP;
    *ok += low_bits;

    return;
}

#ifndef SUPPORT_64BITS
void
hd_sparse(DSS_HUGE i, DSS_HUGE *ok, long seq)
{
    long low_mask, seq_mask;
    static int init = 0;
    static DSS_HUGE *base, *res;

    if (init == 0)
    {
        INIT_HUGE(base);
        INIT_HUGE(res);
        init = 1;
    }

    low_mask = (1 << SPARSE_KEEP) - 1;
    seq_mask = (1 << SPARSE_BITS) - 1;
    bin_bcd2(i, base, base + 1);
    HUGE_SET (base, res);
    HUGE_DIV (res, 1 << SPARSE_KEEP);
    HUGE_MUL (res, 1 << SPARSE_BITS);
    HUGE_ADD (res, seq, res);
    HUGE_MUL (res, 1 << SPARSE_KEEP);
    HUGE_ADD (res, *base & low_mask, res);
    bcd2_bin (&low_mask, *res);
    bcd2_bin (&seq_mask, *(res + 1));
    *ok = low_mask;
    *(ok + 1) = seq_mask;
    return;
}
#endif

long
mk_order(DSS_HUGE index, order_t *o, long upd_num)
{
    long rc = 0; // guojian
    long lcnt;

```

```

    long rprice;
    long ocnt;
    long tmp_date;
    long s_date;
    long r_date;
    long c_date;
    long clk_num;
    long supp_num;
    static char **asc_date = NULL;
    char tmp_str[2];
    char **mk_ascdate PROTO((void));
    int delta = 1;

    if (asc_date == NULL)
        asc_date = mk_ascdate();
    mk_sparse (index, o->okey,
        (upd_num == 0) ? 0 : 1 + upd_num / (10000 / refresh));

    RANDOM(o->custkey, O_CKEY_MIN, O_CKEY_MAX,
O_CKEY_SD);
    while (o->custkey % CUST_MORTALITY == 0)
    {
        o->custkey += delta;
        o->custkey = MIN(o->custkey, O_CKEY_MAX);
        delta *= -1;
    }

    RANDOM(tmp_date, O_ODATE_MIN, O_ODATE_MAX,
O_ODATE_SD);
    strcpy(o->odate, asc_date[tmp_date - STARTDATE]);

    pick_str(&o->priority_set, O_PRIO_SD, o->opriority);
    RANDOM(clk_num, 1, MAX((scale * O_CLRK_SCL),
O_CLRK_SCL), O_CLRK_SD);
    sprintf(o->clerk, O_CLRK_FMT,
        O_CLRK_TAG,
        clk_num);
    o->clen = TEXT(O_CMNT_LEN, O_CMNT_SD, o->comment);
#ifndef DEBUG
    if (o->clen > O_CMNT_MAX) fprintf(stderr, "comment error:
O%d\n", partsupp.tblindex);
#endif /* DEBUG */
    o->spriority = 0;

    o->totalprice = 0;
    o->orderstatus = 'O';
    ocnt = 0;

    RANDOM(o->lines, O_LCNT_MIN, O_LCNT_MAX,
O_LCNT_SD);
    for (lcnt = 0; lcnt < o->lines; lcnt++)
    {
        HUGE_SET(o->okey, o->l[lcnt].okey);
        o->l[lcnt].lcnt = lcnt + 1;
        RANDOM(o->l[lcnt].quantity, L_QTY_MIN, L_QTY_MAX,
L_QTY_SD);
        RANDOM(o->l[lcnt].discount, L_DCNT_MIN, L_DCNT_MAX,
L_DCNT_SD);
        RANDOM(o->l[lcnt].tax, L_TAX_MIN, L_TAX_MAX,
L_TAX_SD);
        pick_str(&l_instruct_set, L_SHIP_SD, o->l[lcnt].shipinstruct);
        pick_str(&l_smode_set, L_SMODE_SD, o->l[lcnt].shipmode);
        o->l[lcnt].clen = TEXT(L_CMNT_LEN, L_CMNT_SD, o-
>l[lcnt].comment);
        RANDOM(o->l[lcnt].partkey, L_PKEY_MIN, L_PKEY_MAX,
L_PKEY_SD);
        RPRICE_BRIDGE( rprice, o->l[lcnt].partkey);
        RANDOM(supp_num, 0, 3, L_SKEY_SD);

```

```

PART_SUPP_BRIDGE( o->l[lcnt].supkey, o->l[lcnt].partkey,
supp_num);
o->l[lcnt].eprice = rprice * o->l[lcnt].quantity;

o->totalprice +=
((o->l[lcnt].eprice *
((long)100 - o->l[lcnt].discount)) / (long)PENNIES ) *
((long)100 + o->l[lcnt].tax)
/ (long)PENNIES;

RANDOM(s_date, L_SDTE_MIN, L_SDTE_MAX,
L_SDTE_SD);
s_date += tmp_date;
RANDOM(c_date, L_CDTE_MIN, L_CDTE_MAX,
L_CDTE_SD);
c_date += tmp_date;
RANDOM(r_date, L_RDTE_MIN, L_RDTE_MAX,
L_RDTE_SD);
r_date += s_date;

strcpy(o->l[lcnt].sdate, asc_date[s_date - STARTDATE]);
strcpy(o->l[lcnt].cdate, asc_date[c_date - STARTDATE]);
strcpy(o->l[lcnt].rdate, asc_date[r_date - STARTDATE]);

if (julian(r_date) <= CURRENTDATE)
{
pick_str(&l_rflag_set, L_RFLG_SD, tmp_str);
o->l[lcnt].rflag[0] = *tmp_str;
}
else
o->l[lcnt].rflag[0] = 'N';

if (julian(s_date) <= CURRENTDATE)
{
ocnt++;
o->l[lcnt].lstatus[0] = 'F';
}
else
o->l[lcnt].lstatus[0] = 'O';
}

if (ocnt > 0)
o->orderstatus = 'P';
if (ocnt == o->lines)
o->orderstatus = 'F';

/* guojian@010908 */
if (!isItMe ((unsigned_int64)o->okey, part_info_for_orders,
LARGE_TABLE))
rc = 1;

return (rc); //Guojian
}

long
mk_part(_int64 index, part_t *p)
{
long rc = 0;
long temp;
long snum;
long brnd;

p->partkey = index;
agg_str(&colors, (long)P_NAME_SCL, (long)P_NAME_SD, p-
>name);
RANDOM(temp, P_MFG_MIN, P_MFG_MAX, P_MFG_SD);
sprintf(p->mfg, P_MFG_FMT, P_MFG_TAG, temp);
RANDOM(brnd, P_BRND_MIN, P_BRND_MAX, P_BRND_SD);

```

```

sprintf(p->brand, P_BRND_FMT,
P_BRND_TAG,
(temp * 10 + brnd));
p->tlen = pick_str(&p_types_set, P_TYPE_SD, p->type);
p->tlen = strlen(p_types_set.list[p->tlen].text);
RANDOM(p->size, P_SIZE_MIN, P_SIZE_MAX, P_SIZE_SD);
pick_str(&p_cntr_set, P_CNTR_SD, p->container);
RPRICE_BRIDGE( p->retailprice, index);
p->cflen = TEXT(P_CMNT_LEN, P_CMNT_SD, p->comment);

for (snum = 0; snum < SUPP_PER_PART; snum++)
{
p->s[snum].partkey = p->partkey;
PART_SUPP_BRIDGE( p->s[snum].supkey, index, snum);
RANDOM(p->s[snum].qty, PS_QTY_MIN, PS_QTY_MAX,
PS_QTY_SD);
RANDOM(p->s[snum].scost, PS_SCST_MIN, PS_SCST_MAX,
PS_SCST_SD);
p->s[snum].cflen = TEXT(PS_CMNT_LEN, PS_CMNT_SD, p-
>s[snum].comment);
}

/* guojian@010908 */
if (!isItMe ((unsigned_int64)p->partkey, part_info_for_part,
SMALL_TABLE))
rc = 1;

return (rc);
}

long
mk_supp(_int64 index, supplier_t *s)
{
long rc = 0;
long i,
bad_press,
noise,
offset,
type;

s->supkey = index;

sprintf(s->name, S_NAME_FMT, S_NAME_TAG, index);
s->alen = V_STR(S_ADDR_LEN, S_ADDR_SD, s->address);
RANDOM(i, 0, nations.count - 1, S_NTRG_SD);
s->nation_code= i;
gen_phone(i, s->phone, S_PHNE_SD);
RANDOM(s->acctbal, S_ABAL_MIN, S_ABAL_MAX,
S_ABAL_SD);

s->cflen = TEXT(S_CMNT_LEN, S_CMNT_SD, s->comment);
/* these calls should really move inside the if stmt below,
* but this will simplify seedless parallel load
*/
RANDOM(bad_press, 1, 10000, BBB_CMNT_SD);
RANDOM(type, 0, 100, BBB_TYPE_SD);
RANDOM(noise, 0, (s->cflen - BBB_CMNT_LEN), BBB_JNK_SD);
RANDOM(offset, 0, (s->cflen - (BBB_CMNT_LEN + noise)),
BBB_OFFSET_SD);
if (bad_press <= S_CMNT_BBB)
{
type = (type < BBB_DEADBEATS) ?0:1;
memcpy(s->comment + offset, BBB_BASE, BBB_BASE_LEN);
if (type == 0)
memcpy(s->comment + BBB_BASE_LEN + offset + noise,
BBB_COMPLAIN, BBB_TYPE_LEN);
else
memcpy(s->comment + BBB_BASE_LEN + offset + noise,
BBB_COMMEND, BBB_TYPE_LEN);
}
}

```

```

/* guojian@010908 */
if(!isItMe((unsigned _int64)s->supkey, part_info_for_supplier,
SMALL_TABLE))
    rc = 1;

return (rc); // guojian
}

struct
{
    char    *mdes;
    long    days;
    long    dcnt;
}    months[] =

{
    {NULL, 0, 0},
    {"JAN", 31, 31},
    {"FEB", 28, 59},
    {"MAR", 31, 90},
    {"APR", 30, 120},
    {"MAY", 31, 151},
    {"JUN", 30, 181},
    {"JUL", 31, 212},
    {"AUG", 31, 243},
    {"SEP", 30, 273},
    {"OCT", 31, 304},
    {"NOV", 30, 334},
    {"DEC", 31, 365}
};

long
mk_time(long index, dss_time_t *t)
{
    long    m = 0;
    long    y;
    long    d;

    t->timekey = index + JDAY_BASE;
    y = julian(index + STARTDATE - 1) / 1000;
    d = julian(index + STARTDATE - 1) % 1000;
    while (d > months[m].dcnt + LEAP_ADJ(y, m))
        m++;
    PR_DATE(t->alpha, y, m,
        d - months[m - 1].dcnt - ((LEAP(y) && m > 2) ? 1 : 0));
    t->year = 1900 + y;
    t->month = m + 12 * y + JMNTH_BASE;
    t->week = (d + T_START_DAY - 1) / 7 + 1;
    t->day = d - months[m - 1].dcnt - LEAP_ADJ(y, m-1);

    return (0);
}

int
mk_nation(_int64 index, code_t *c)
{
    c->code = (long) index - 1;
    c->text = nations.list[index - 1].text;
    c->join = nations.list[index - 1].weight;
    c->clen = TEXT(N_CMNT_LEN, N_CMNT_SD, c->comment);
    return(0);
}

int
mk_region(_int64 index, code_t *c)
{
    c->code = (long) index - 1;
    c->text = regions.list[index - 1].text;

```

```

c->join = 0; /* for completeness */
c->clen = TEXT(R_CMNT_LEN, R_CMNT_SD, c->comment);
return(0);
}

/*****
*****/
/* This function helps judging if the record should be populated to the
current node */
/*****
*****/
int isItMe(unsigned _int64 key, struct sqlupi *part_info, int tablesize)
// guojian@723
{
    int i;
    int rc;
    unsigned short    num_ptrs    = 1; // number of pointers
    unsigned char    *ptr_arr[1]; // an array of pointers to char string
    unsigned short    ptr_len[1]; // an array of character string lengths
    short    part_num; // partition number
    SQL_PDB_NODE_TYPE    node_number; // node number
    struct sqlca    sqlca_tpch; // SQLCA

    /* allocate memory
    * 19 bytes is the larges
    */
    ptr_arr[0] = (unsigned char*)malloc(MAX_KEY_LEN);
    memset(ptr_arr[0], 0, sizeof(unsigned char) * MAX_KEY_LEN);

    // get the value for ptr_arr[0]
    if(tablesize == LARGE_TABLE)
        sprintf(ptr_arr[0], HUGE_FORMAT, (DSS_HUGE *)key);
    else
        sprintf(ptr_arr[0], "%I64u\0", key);

    /* get the length of ptr_arr[0] */
    ptr_len[0] = strlen(ptr_arr[0]);
    part_num=0;
    node_number=0;

    // printf("Key = %s, Key length = %d\n", ptr_arr[0], ptr_len[0]);

    sqlugrpn (num_ptrs,
        ptr_arr,
        ptr_len,
        countrycode,
        codepage,
        part_info,
        &part_num,
        &node_number,
        chklvl,
        &sqlca_tpch,
        sDataFormat,
        (void*)0,
        (void*)0);

    // free up memory des@0725 (memfree) guojian@0725 (free)
    free (ptr_arr[0]);
    // printf("(%ld - ", key);
    // printf("%d)\n", part_num);
    if(sqlca_tpch.sqlcode != 0) {
        printf("SQLUGRPN Failed - sqlcode = %d\n", sqlca_tpch.sqlcode);
        exit(1);
    }
    //printf("Node number = %d, me = (%d / %d) = %d\n", node_number,
    part_num, Numnodes, (part_num / Numnodes));
    // see if it is me
    if ((part_num % Numnodes) != me)

```

```

    rc = 0;
else
    rc = 1;

return rc;
}

/* @(#)driver.c 2.1.8.4 */
/* main driver for dss banchmark */

#define DECLARER                /* EXTERN references get
defined here */
#define NO_FUNC (int (*) ()) NULL /* to clean up tdefs */
#define NO_LFUNC (long (*) ()) NULL /* to clean up tdefs */

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

#define NUM_TABLES 6            // guojian@0725

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

#include "config.h"
#include <stdlib.h>
#if (defined(_POSIX_))||!defined(WIN32)    /* Change for Windows
NT */
#include <unistd.h>
#include <sys/wait.h>
#endif /* WIN32 */
#include <stdio.h>                /* */
#include <limits.h>
#include <math.h>
#include <ctype.h>
#include <signal.h>
#include <string.h>
#include <errno.h>
#ifdef HP
#include <strings.h>
#endif
#if (defined(WIN32)&&!defined(_POSIX_))
#include <process.h>
#pragma warning(disable:4201)
#pragma warning(disable:4214)
#pragma warning(disable:4514)
#define WIN32_LEAN_AND_MEAN
#define NOATOM
#define NOGDICAPMASKS
#define NOMETAFILE
#define NOMINMAX
#define NOMSG
#define NOOPENFILE
#define NORASTEROPS
#define NOSCROLL
#define NOSOUND
#define NOSYSMETRICS
#define NOTEXTMETRIC
#define NOWH
#define NOCOMM
#define NOKANJI
#define NOMCX
#include <windows.h>
#pragma warning(default:4201)
#pragma warning(default:4214)
#endif

#include "dss.h"
#include "dsstypes.h"
#include "bcd2.h"

```

B.18 driver.sqc

```

EXEC SQL BEGIN DECLARE SECTION;          /* Declare Host
Variables */
char dbname[8];
EXEC SQL END DECLARE SECTION;

/*
 * Function prototypes
 */
void usage (void);
int prep_direct (char *);
int close_direct (void);
void kill_load (void);
int pload (int tbl);
//void gen_tbl (int tnum, long start, long count, long upd_num);
void gen_tbl (int tnum, DSS_HUGE start, DSS_HUGE count, long
upd_num);
int pr_drange (int tbl, long min, long cnt, long num);
int set_files (int t, int pload);
int partial (int, int);

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

void getPart_info (void);                // guojian@0725

extern char *tablename[] = {"tpcd.supplier",
                           "tpcd.customer",
                           "tpcd.part",
                           "tpcd.partsupp",
                           "tpcd.orders",
                           "tpcd.lineitem"};    // guojian@0725

struct sqlupi part_info[NUM_TABLES];      // guojian@0725
struct sqlca sqlca_tpch[NUM_TABLES];     // guojian@0725

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

extern int optind, opterr;
extern char *optarg;
//long rowcnt = 0, minrow = 0, upd_num = 0;
DSS_HUGE rowcnt = 0, minrow = 0;
long upd_num = 0;
double flt_scale;
#if (defined(WIN32)&&!defined(_POSIX_))
char *spawn_args[25];
#endif

/*
 * general table descriptions. See dss.h for details on structure
 * NOTE: tables with no scaling info are scaled according to
 * another table
 *
 * the following is based on the tdef structure defined in dss.h as:
 * typedef struct
 * {
 * char *name;          -- name of the table;
 *                          flat file output in <name>.tbl

```

```

* long   base;           -- base scale rowcount of table;
*                               0 if derived
* int     (*header) ();   -- function to prep output
* int     (*loader[2]) (); -- functions to present output
* long    (*gen_seed) ();  -- functions to seed the RNG
* int     (*verify) ();   -- function to verify the data set without building it
* int     child;          -- non-zero if there is an associated detail table
* unsigned long vttotal;   -- "checksum" total
* }         tdef;
*
*/

/*
* flat file print functions; used with -F(lat) option
*/
int pr_cust (customer_t * c, int mode);
int pr_line (order_t * o, int mode);
int pr_order (order_t * o, int mode);
int pr_part (part_t * p, int mode);
int pr_psupp (part_t * p, int mode);
int pr_supp (supplier_t * s, int mode);
int pr_order_line (order_t * o, int mode);
int pr_part_psupp (part_t * p, int mode);
int pr_nation (code_t * c, int mode);
int pr_region (code_t * c, int mode);

/*
* inline load functions; used with -D(irect) option
*/
int ld_cust (customer_t * c, int mode);
int ld_line (order_t * o, int mode);
int ld_order (order_t * o, int mode);
int ld_part (part_t * p, int mode);
int ld_psupp (part_t * p, int mode);
int ld_supp (supplier_t * s, int mode);
int ld_order_line (order_t * o, int mode);
int ld_part_psupp (part_t * p, int mode);
int ld_nation (code_t * c, int mode);
int ld_region (code_t * c, int mode);

/*
* seed generation functions; used with '-O s' option
*/
long sd_cust (int child, long skip_count);
long sd_line (int child, long skip_count);
long sd_order (int child, long skip_count);
long sd_part (int child, long skip_count);
long sd_psupp (int child, long skip_count);
long sd_supp (int child, long skip_count);
long sd_order_line (int child, long skip_count);
long sd_part_psupp (int child, long skip_count);

/*
* header output functions; used with -h(eader) option
*/
int hd_cust (FILE * f);
int hd_line (FILE * f);
int hd_order (FILE * f);
int hd_part (FILE * f);
int hd_psupp (FILE * f);
int hd_supp (FILE * f);
int hd_order_line (FILE * f);
int hd_part_psupp (FILE * f);
int hd_nation (FILE * f);
int hd_region (FILE * f);

/*
* data verification functions; used with -O v option
*/
int vrf_cust (customer_t * c, int mode);

```

```

int vrf_line (order_t * o, int mode);
int vrf_order (order_t * o, int mode);
int vrf_part (part_t * p, int mode);
int vrf_psupp (part_t * p, int mode);
int vrf_supp (supplier_t * s, int mode);
int vrf_order_line (order_t * o, int mode);
int vrf_part_psupp (part_t * p, int mode);
int vrf_nation (code_t * c, int mode);
int vrf_region (code_t * c, int mode);

tdef tdefs[] =
{
    {"part.tbl", "part table", 200000, hd_part,
     {pr_part, ld_part}, sd_part, vrf_part, PSUPP, 0},
    {"partsupp.tbl", "partsupplier table", 200000, hd_psupp,
     {pr_psupp, ld_psupp}, sd_psupp, vrf_psupp, NONE, 0},
    {"supplier.tbl", "suppliers table", 10000, hd_supp,
     {pr_supp, ld_supp}, sd_supp, vrf_supp, NONE, 0},
    {"customer.tbl", "customers table", 150000, hd_cust,
     {pr_cust, ld_cust}, sd_cust, vrf_cust, NONE, 0},
    {"orders.tbl", "order table", 150000, hd_order,
     {pr_order, ld_order}, sd_order, vrf_order, LINE, 0},
    {"lineitem.tbl", "lineitem table", 150000, hd_line,
     {pr_line, ld_line}, sd_line, vrf_line, NONE, 0},
    {"orders.tbl", "orders/lineitem tables", 150000, hd_order_line,
     {pr_order_line, ld_order_line}, sd_order, vrf_order_line, LINE,
0},
    {"part.tbl", "part/partsupplier tables", 200000, hd_part_psupp,
     {pr_part_psupp, ld_part_psupp}, sd_part, vrf_part_psupp,
PSUPP, 0},
    {"nation.tbl", "nation table", NATIONS_MAX, hd_nation,
     {pr_nation, ld_nation}, NO_LFUNC, vrf_nation, NONE, 0},
    {"region.tbl", "region table", NATIONS_MAX, hd_region,
     {pr_region, ld_region}, NO_LFUNC, vrf_region, NONE, 0},
};

int *pids;

/*
* routines to handle the graceful cleanup of multi-process loads
*/

void
stop_proc (int signum)
{
    exit (0);
}

void
kill_load (void)
{
    int i;

    #if !defined(U2200) && !defined(DOS)
        for (i = 0; i < children; i++)
            if (pids[i])
                KILL (pids[i]);
    #endif /* !U2200 && !DOS */
    return;
}

/*
* re-set default output file names
*/
int
set_files (int i, int pload)
{
    char line[80], *new_name;

```

```

    if (table & (1 << i))
child_table:
    {
        if (pload != -1)
            sprintf (line, "%s.%d", tdefs[i].name, pload);
        else
        {
            printf ("Enter new destination for %s data: ",
                    tdefs[i].name);
            if (fgets (line, sizeof (line), stdin) == NULL)
                return (-1);
            if ((new_name = strchr (line, '\n')) != NULL)
                *new_name = '\0';
            if (strlen (line) == 0)
                return (0);
        }
        new_name = (char *) malloc (strlen (line) + 1);
        MALLOC_CHECK (new_name);
        strcpy (new_name, line);
        tdefs[i].name = new_name;
        if (tdefs[i].child != NONE)
        {
            i = tdefs[i].child;
            tdefs[i].child = NONE;
            goto child_table;
        }
    }

    return (0);
}

/*
 * read the distributions needed in the benchamrk
 */
void
load_dists (void)
{
    read_dist (env_config (DIST_TAG, DIST_DFLT), "p_cntr",
&p_cntr_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "colors", &colors);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "p_types",
&p_types_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "nations",
&nations);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "regions",
&regions);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "o_oprio",
&o_priority_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "instruct",
&l_instruct_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "smode",
&l_smode_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "category",
&l_category_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "rflag",
&l_rflag_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "msegmnt",
&c_mseg_set);

    /* load the distributions that contain text generation */
    read_dist (env_config (DIST_TAG, DIST_DFLT), "nouns", &nouns);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "verbs", &verbs);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "adjectives",
&adjectives);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "adverbs",
&adverbs);

```

```

    read_dist (env_config (DIST_TAG, DIST_DFLT), "auxillaries",
&auxillaries);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "terminators",
&terminators);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "articles",
&articles);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "prepositions",
&prepositions);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "grammar",
&grammar);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "np", &np);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "vp", &vp);
}

/*
 * generate a particular table
 */
//gen_tbl (int tnum, long start, long count, long upd_num)
void
gen_tbl (int tnum, DSS_HUGE start, DSS_HUGE count, long upd_num)
{
    static order_t o;
    supplier_t supp;
    customer_t cust;
    part_t part;
    code_t code;
    static int completed = 0;
    static int init = 0;
    DSS_HUGE i;

    int rows_per_segment=0;
    int rows_this_segment=-1;
    int residual_rows=0;

    if (insert_segments)
    {
        rows_per_segment = count / insert_segments;
        residual_rows = count - (rows_per_segment * insert_segments);
    }

    if (init == 0)
    {
        INIT_HUGE(o.okey);
        for (i=0; i < O_LCNT_MAX; i++)
            INIT_HUGE(o.l[i].okey);
        init = 1;
    }

    for (i = start; count; count--, i++)
    {
        LIFENOISE (1000, i);
        row_start(tnum);
        switch (tnum)
        {
            case LINE:
            case ORDER:
            case ORDER_LINE:
                if (!mk_order (i, &o, upd_num % 10000)) {
                    if (insert_segments && (upd_num > 0))
                        if((upd_num / 10000) < residual_rows)
                        {
                            if((++rows_this_segment) > rows_per_segment)
                            {
                                rows_this_segment=0;
                                upd_num += 10000;
                            }
                        }
                    else
                    {

```

```

        if(++rows_this_segment) >= rows_per_segment)
        {
            rows_this_segment=0;
            upd_num += 10000;
        }
    }

    if (set_seeds == 0)
    if (validate)
        tdefs[tnum].verify(&o, 0);
    else
        tdefs[tnum].loader[direct] (&o, upd_num);
}
break;
case SUPP:
    if (!mk_supp (i, &supp))
    {
        if (set_seeds == 0)
        if (validate)
            tdefs[tnum].verify(&supp, 0);
        else
            tdefs[tnum].loader[direct] (&supp, upd_num);
    }
    break;
case CUST:
    if (!mk_cust (i, &cust) )
    {
        if (set_seeds == 0)
        if (validate)
            tdefs[tnum].verify(&cust, 0);
        else
            tdefs[tnum].loader[direct] (&cust, upd_num);
    }
    break;
case PSUPP:
case PART:
case PART_PSUPP:
    if (!mk_part (i, &part))
    {
        if (set_seeds == 0)
        if (validate)
            tdefs[tnum].verify(&part, 0);
        else
            tdefs[tnum].loader[direct] (&part, upd_num);
    }
    break;
case NATION:
    mk_nation (i, &code);
    if (set_seeds == 0)
    if (validate)
        tdefs[tnum].verify(&code, 0);
    else
        tdefs[tnum].loader[direct] (&code, 0);
    break;
case REGION:
    mk_region (i, &code);
    if (set_seeds == 0)
    if (validate)
        tdefs[tnum].verify(&code, 0);
    else
        tdefs[tnum].loader[direct] (&code, 0);
    break;
}
row_stop(tnum);
if (set_seeds && (i % tdefs[tnum].base) < 2)
{
    printf("\nSeeds for %s at rowcount %ld\n",
tdefs[tnum].comment, i);
    dump_seeds(tnum);
}

```

```

    }
    completed |= 1 << tnum;
}

void
usage (void)
{
    fprintf (stderr, "%s\n%s\n\t%s\n%s %s\n\n",
        "USAGE:",
        "dbgen [-{vffD}] [-O {fhmsv}][-T {pcsoPSOL}]",
        "[-s <scale>][-C <procs>][-S <step>]",
        "dbgen [-v] [-O {dfhmr}] [-s <scale>]",
        "[-U <updates>] [-r <percent>]");
    fprintf (stderr, "-b <s> -- load distributions for <s>\n");
    fprintf (stderr, "-C <n> -- use <n> processes to generate data\n");
    fprintf (stderr, "    [Under DOS, must be used with -S]\n");
    fprintf (stderr, "-D -- do database load in line\n");
    fprintf (stderr, "-d <n> -- split deletes between <n> files\n");
    fprintf (stderr, "-f -- force. Overwrite existing files\n");
    fprintf (stderr, "-F -- generate flat files output\n");
    fprintf (stderr, "-h -- display this message\n");
    fprintf (stderr, "-i <n> -- split inserts between <n> files\n");
    fprintf (stderr, "-n <s> -- inline load into database <s>\n");
    fprintf (stderr, "-O d -- generate SQL syntax for deletes\n");
    fprintf (stderr, "-O f -- over-ride default output file names\n");
    fprintf (stderr, "-O h -- output files with headers\n");
    fprintf (stderr, "-O m -- produce columnar output\n");
    fprintf (stderr, "-O r -- generate key ranges for deletes\n");
    fprintf (stderr, "-O v -- Verify data set without generating it.\n");
    fprintf (stderr, "-q -- enable QUIET mode\n");
    fprintf (stderr, "-r <n> -- updates refresh (n/100)% of the\n");
    fprintf (stderr, "    data set\n");
    fprintf (stderr, "-s <n> -- set Scale Factor (SF) to <n> \n");
    fprintf (stderr, "-S <n> -- build the <n>th step of the data/update
set\n");
    fprintf (stderr, "-T c -- generate cutomers ONLY\n");
    fprintf (stderr, "-T l -- generate nation/region ONLY\n");
    fprintf (stderr, "-T L -- generate lineitem ONLY\n");
    fprintf (stderr, "-T n -- generate nation ONLY\n");
    fprintf (stderr, "-T o -- generate orders/lineitem ONLY\n");
    fprintf (stderr, "-T O -- generate orders ONLY\n");
    fprintf (stderr, "-T p -- generate parts/partsupp ONLY\n");
    fprintf (stderr, "-T P -- generate parts ONLY\n");
    fprintf (stderr, "-T r -- generate region ONLY\n");
    fprintf (stderr, "-T s -- generate suppliers ONLY\n");
    fprintf (stderr, "-T S -- generate partsupp ONLY\n");
    fprintf (stderr, "-U <s> -- generate <s> update sets\n");
    fprintf (stderr, "-v -- enable VERBOSE mode\n");
    fprintf (stderr,
        "\nTo generate the SF=1 (1GB), validation database population,
use:\n");
    fprintf (stderr, "\tdbgen -vff -s 1\n");
    fprintf (stderr, "\nTo generate updates for a SF=1 (1GB), use:\n");
    fprintf (stderr, "\tdbgen -v -U 1 -s 1\n");
}

/*
 * pload() -- handle the parallel loading of tables
 */
#ifdef DOS
/*
 * int partial(int tbl, int s) -- generate the s-th part of the named tables data
 */
int
partial (int tbl, int s)
{
    DSS_HUGE rowcnt;
    long extra;

```

```

    if (verbose > 0)
    {
        fprintf(stderr, "\tStarting to load stage %d of %d for %s...",
            s, children, tdefs[tbl].comment);
    }

    if (direct == 0)
        set_files (tbl, s);

    rowcnt = set_state(tbl, scale, children, s, &extra);

    if (s == children)
        gen_tbl (tbl, rowcnt * (s - 1) + 1, rowcnt + extra, upd_num);
    else
        gen_tbl (tbl, rowcnt * (s - 1) + 1, rowcnt, upd_num);

    if (verbose > 0)
        fprintf(stderr, "done.\n");

    return (0);
}

int
pload (int tbl)
{
    int c = 0, i, status;

    if (verbose > 0)
    {
        fprintf(stderr, "Starting %d children to load %s",
            children, tdefs[tbl].comment);
    }
    for (c = 0; c < children; c++)
    {
        pids[c] = SPAWN ();
        if (pids[c] == -1)
        {
            perror ("Child loader not created");
            kill_load ();
            exit (-1);
        }
        else if (pids[c] == 0) /* CHILD */
        {
            SET_HANDLER (stop_proc);
            verbose = 0;
            partial (tbl, c+1);
            exit (0);
        }
        else if (verbose > 0) /* PARENT */
            fprintf (stderr, ".");
    }

    if (verbose > 0)
        fprintf (stderr, "waiting...");

    c = children;
    while (c)
    {
        i = WAIT (&status, pids[c - 1]);
        if (i == -1 && children)
        {
            if (errno == ECHILD)
                fprintf (stderr, "\nCould not wait on pid %d\n", pids[c -
1]);
            else if (errno == EINTR)
                fprintf (stderr, "\nProcess %d stopped abnormally\n",
pids[c - 1]);
            else if (errno == EINVAL)

```

```

        fprintf (stderr, "\nProgram bug\n");
    }
    if (! WIFEXITED(status)) {
        (void) fprintf(stderr, "\nProcess %d: ", i);
        if (WIFSIGNALED(status)) {
            (void) fprintf(stderr, "rcvd signal %d\n",
                WTERMSIG(status));
        } else if (WIFSTOPPED(status)) {
            (void) fprintf(stderr, "stopped, signal %d\n",
                WSTOPSIG(status));
        }
    }
    c--;
}

if (verbose > 0)
    fprintf (stderr, "done\n");
return (0);
}
#endif /* !DOS */

void connect_to_TM() {
    SQL_STRUCTURE sqlca sqlca;
    char *dbname_ptr;
    void *ctx;

    if ((dbname_ptr=getenv("TPCD_DBNAME")) != NULL)
        strcpy(dbname, dbname_ptr);
    else {
        printf(" Database name not defined.\n");
        exit(1);
    }

    EXEC SQL CONNECT TO :dbname IN SHARE MODE;

    if (SQLCODE != 0) {
        fprintf(stderr, "%s: CONNECT TO %s: SQLCODE = %d\n",
            __FILE__, dbname, SQLCODE);
        exit(-1);
    }
} /* connect_to_TM */

void disconnect_from_TM() {
    SQL_STRUCTURE sqlca sqlca;
    void *ctx;

    EXEC SQL CONNECT RESET;

    if (SQLCODE != 0)
        fprintf(stderr, "%s: CONNECT RESET: SQLCODE = %d\n",
            __FILE__, SQLCODE);
} /* disconnect_from_TM */

void
process_options (int count, char **vector)
{
    int option;

    while ((option = getopt (count, vector,
        "b:C:Dd:Ffi:hn:O:P:qr:s:S:T:U:v")) != -1)
        switch (option)
        {
            case 'b': /* load distributions from named file
*/
                d_path = (char *)malloc(strlen(optarg) + 1);
                MALLOC_CHECK(d_path);

```

```

        strcpy(d_path, optarg);
        break;
case 'q':                /* all prompts disabled */
    verbose = -1;
    break;
case 'i':
    insert_segments = atoi (optarg);
    break;
case 'd':
    delete_segments = atoi (optarg);
    break;
case 'S':                /* generate a particular STEP */
    step = atoi (optarg);
    break;
case 'v':                /* life noises enabled */
    verbose = 1;
    break;
case 'f':                /* blind overwrites; Force */
    force = 1;
    break;
case 'T':                /* generate a specifc table */
    switch (*optarg)
    {
        case 'c':        /* generate customer ONLY */
            table = 1 << CUST;
            break;
        case 'L':        /* generate lineitems ONLY */
            table = 1 << LINE;
            break;
        case 'l':        /* generate code table ONLY */
            table = 1 << NATION;
            table |= 1 << REGION;
            break;
        case 'n':        /* generate nation table ONLY */
            table = 1 << NATION;
            break;
        case 'O':        /* generate orders ONLY */
            table = 1 << ORDER;
            break;
        case 'o':        /* generate orders/lineitems ONLY */
            table = 1 << ORDER_LINE;
            break;
        case 'P':        /* generate part ONLY */
            table = 1 << PART;
            break;
        case 'p':        /* generate part/partsupp ONLY */
            table = 1 << PART_PSUPP;
            break;
        case 'r':        /* generate region table ONLY */
            table = 1 << REGION;
            break;
        case 'S':        /* generate partsupp ONLY */
            table = 1 << PSUPP;
            break;
        case 's':        /* generate suppliers ONLY */
            table = 1 << SUPP;
            break;
        default:
            fprintf (stderr, "Unknown table name %s\n",
                    optarg);
            usage ();
            exit (1);
    }
    break;
case 's':                /* scale by Percentage of base
rowcount */
    case 'P':            /* for backward compatibility */
        flt_scale = atof (optarg);
        if (flt_scale < MIN_SCALE)
        {

```

```

            int i;

            scale = 1;
            for (i = PART; i < REGION; i++)
            {
                tdefs[i].base *= flt_scale;
                if (tdefs[i].base < 1)
                    tdefs[i].base = 1;
            }
        }
        else
            scale = (long) flt_scale;
        if (scale > MAX_SCALE)
        {
            fprintf (stderr, "%s %5.0f %s\n\t%s\n\n",
                    "NOTE: Data generation for scale factors >",
                    MAX_SCALE,
                    "GB is still in development,",
                    "and is not yet supported.\n");
            fprintf (stderr,
                    "Your resulting data set MAY NOT BE
COMPLIANT!\n");
        }
        break;
case 'O':                /* optional actions */
    switch (tolower (*optarg))
    {
        case 'd':        /* generate SQL for deletes */
            gen_sql = 1;
            break;
        case 'f':        /* over-ride default file names */
            fnames = 1;
            break;
        case 'h':        /* generate headers */
            header = 1;
            break;
        case 'm':        /* generate columnar output */
            columnar = 1;
            break;
        case 'r':        /* generate key ranges for delete */
            gen_rng = 1;
            break;
        case 's':        /* calibrate the RNG usage */
            set_seeds = 1;
            break;
        case 'v':        /* validate the data set */
            validate = 1;
            break;
        default:
            fprintf (stderr, "Unknown option name %s\n",
                    optarg);
            usage ();
            exit (1);
    }
    break;
case 'D':                /* direct load of generated data
*/
        direct = 1;
        break;
case 'F':                /* generate flat files for later
loading */
        direct = 0;
        break;
case 'U':                /* generate flat files for update
stream */
        updates = atoi (optarg);
        break;
case 'r':                /* set the refresh (update)
percentage */
        refresh = atoi (optarg);

```

```

        break;
#endif DOS
        case 'C':
            children = atoi (optarg);
            break;
#endif /* !DOS */
        case 'n':
            /* set name of database for direct
load */
            db_name = (char *) malloc (strlen (optarg) + 1);
            MALLOC_CHECK (db_name);
            strcpy (db_name, optarg);
            break;
        default:
            printf ("ERROR: option '%c' unknown.\n",
                *(vector[optind] + 1));
        case 'h':
            /* something unexpected */
            fprintf (stderr,
                "%s Population Generator (Version
%d.%d.%d%s)\n",
                NAME, VERSION, RELEASE,
                MODIFICATION, PATCH);
            fprintf (stderr, "Copyright %s %s\n", TPC,
C_DATES);
            usage ();
            exit (1);
    }

#ifdef DOS
    if (children != 1 && step == -1)
    {
        pids = malloc(children * sizeof(pid_t));
        MALLOC_CHECK(pids)
    }
#else
    if (children != 1 && step < 0)
    {
        fprintf(stderr, "ERROR: -C must be accompanied by -S on this
platform\n");
        exit(1);
    }
#endif /* DOS */

    return;
}

/*
 * MAIN
 */
/* assumes the existence of getopt() to clean up the command
 * line handling
 */
int
main (int ac, char **av)
{
    int i;
    char *numnodesptr;

    /****** Get partition info for each table
    *****/
    char *p;
    if((p=getenv("DSS_NODE")) == NULL) {
        printf("Node Number not defined.\n");
        exit(1);
    }
    else {
        me = atoi(p);
        printf("Generating Data for Node %d.\n", me);
    }
    getPart_info();
    // guojian@0725

```

```

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

    table = (1 << CUST) |
        (1 << SUPP) |
        (1 << NATION) |
        (1 << REGION) |
        (1 << PART_PSUPP) |
        (1 << ORDER_LINE);

    force = 0;
    insert_segments=0;
    delete_segments=0;
    insert_orders_segment=0;
    insert_lineitem_segment=0;
    delete_segment=0;
    verbose = 0;
    columnar = 0;
    set_seeds = 0;
    header = 0;
    direct = 0;
    scale = 1;
    flt_scale = 1.0;
    updates = 0;
    refresh = UPD_PCT;
    step = -1;
    tdefs[ORDER].base *=
        ORDERS_PER_CUST;
        /* have to do this after init
    */
    tdefs[LIN].base *=
        ORDERS_PER_CUST;
        /* have to do this after init
    */
    tdefs[ORDER_LINE].base *=
        ORDERS_PER_CUST;
        /* have to do this after init
    */

    fnames = 0;
    db_name = NULL;
    gen_sql = 0;
    gen_mrg = 0;
    children = 1;
    d_path = NULL;

#ifdef NO_SUPPORT
    signal (SIGINT, exit);
#endif /* NO_SUPPORT */
    process_options (ac, av);
    if (defined(WIN32)&&!defined(_POSIX_))
        for (i = 0; i < ac; i++)
        {
            spawn_args[i] = malloc ((strlen (av[i]) + 1) * sizeof (char));
            MALLOC_CHECK (spawn_args[i]);
            strcpy (spawn_args[i], av[i]);
        }
    spawn_args[ac] = NULL;
#endif

    if (verbose >= 0)
    {
        fprintf (stderr,
            "%s Population Generator (Version %d.%d.%d%s)\n",
            NAME, VERSION, RELEASE, MODIFICATION,
PATCH);
        fprintf (stderr, "Copyright %s %s\n", TPC, C_DATES);
    }

    //DJD
    if ((numnodesptr=getenv("DSS_NUMNODES")) != NULL) {
        Numnodes = atoi(numnodesptr);
        printf( "Using %d Nodes\n", Numnodes);
    }
    else {

```

```

    printf(" Database name not defined.\n");
    exit(1);
}

load_dists ();
/* have to do this after init */
tdefs[NATION].base = nations.count;
tdefs[REGION].base = regions.count;

/*
 * updates are never parallelized
 */
if (updates)
{
    /*
     * Set RNG to start generating rows beyond SF=scale
     */
    double fix1;
    set_state (ORDER, scale, 1, 2, (long *)&i);
    fix1 = (double)tdefs[ORDER_LINE].base / (double)10000;
    rowcnt = (int)(fix1 * scale * refresh);
    if (step > 0)
    {
        /*
         * adjust RNG for any prior update generation
         */
        sd_order(0, rowcnt * (step - 1));
        sd_line(0, rowcnt * (step - 1));
        upd_num = step - 1;
    }
    else
        upd_num = 0;

    while (upd_num < updates)
    {
        if (verbose > 0)
            fprintf (stderr,
                    "Generating update pair #%d for %s [pid: %d]",
                    upd_num + 1, tdefs[ORDER_LINE].comment,
                    DSS_PROC);
        insert_orders_segment=0;
        insert_lineitem_segment=0;
        delete_segment=0;
        minrow = upd_num * rowcnt + 1;

        gen_tbl (ORDER_LINE, minrow, rowcnt, upd_num + 1);
        if (verbose > 0)
            fprintf (stderr, "done.\n");
        pr_drange (ORDER_LINE, minrow, rowcnt, upd_num + 1);
        upd_num++;
    }

    exit (0);
}

/**
 ** actual data generation section starts here
 **/
/*
 * open database connection or set all the file names, as appropriate
 */
if (direct)
    prep_direct ((db_name) ? db_name : DBNAME);
else if (fnames)
    for (i = PART; i <= REGION; i++)
    {
        if (table & (1 << i))
            if (set_files (i, -1))
            {
                fprintf (stderr, "Load aborted!\n");

```

```

                exit (1);
            }
        }

    /*
     * traverse the tables, invoking the appropriate data generation routine for
     any to be built
     */
    for (i = PART; i <= REGION; i++)
        if (table & (1 << i))
        {
            if (children > 1 && i < NATION)
                if (step >= 0)
                {
                    if (validate)
                    {
                        INTERNAL_ERROR("Cannot validate
parallel data generation");
                    }
                    else
                        partial (i, step);
                }
            #ifdef DOS
            else
            {
                fprintf (stderr,
                        "Parallel load is not supported on your
platform.\n");
                exit (1);
            }
            #else
            else
            {
                if (validate)
                {
                    INTERNAL_ERROR("Cannot validate
parallel data generation");
                }
                else
                    pload (i);
            }
            #endif /* DOS */
            else
            {
                minrow = 1;
                if (i < NATION)
                    rowcnt = tdefs[i].base * scale;
                else
                    rowcnt = tdefs[i].base;
                if (verbose > 0)
                    fprintf (stderr, "%s data for %s [pid: %ld]",
                            (validate)?"Validating":"Generating",
                            tdefs[i].comment, DSS_PROC);
                gen_tbl (i, minrow, rowcnt, upd_num); /*
start@guojian/00708 */
                if (verbose > 0)
                    fprintf (stderr, "done.\n");
            }
            if (validate)
                printf("Validation checksum for %s at %d GB:
%0x\n",
                        tdefs[i].name, scale, tdefs[i].vtotal);
        }

        if (direct)
            close_direct ();

        return (0);
    }
}

```

```

/*****
*****/
/* get partition info for each table guojian@0725 */
/*****
*****/
void getPart_info (void)
{
    int i;
    struct sqlupi *dummyptr;

    connect_to_TM();

    for (i = 0; i < NUM_TABLES; i++) {
        sqlugtpi( tablename[i], &part_info[i], &sqlca_tpch[i]);
    } /* endfor */

    part_info_for_supplier = &part_info[0];
    dummyptr = part_info_for_supplier;
    part_info_for_customer = &part_info[1];
    part_info_for_part = &part_info[2];
    part_info_for_partsupp = &part_info[3];
    part_info_for_orders = &part_info[4];
    part_info_for_lineitem = &part_info[5];

    disconnect_from_TM();

    return;
}

```

B.19 dss.h

```

/*
 * Scsid:  @(#)dss.h      2.1.8.5
 *
 * general definitions and control information for the DSS code
 * generator; if it controls the data set, it's here
 */
#ifndef DSS_H
#define DSS_H
#ifdef TPC
#define NAME          "TPC-H"
#define VERSION      1
#define RELEASE      3
#define MODIFICATION  0
#define PATCH        ""
#endif
#ifdef TPCR
#define NAME          "TPC-R"
#define VERSION      1
#define RELEASE      3
#define MODIFICATION  0
#define PATCH        ""
#endif
#ifndef NAME
#error Benchmark version must be defined in config.h
#endif
#define TPC          "Transaction Processing Performance Council"
#define C_DATES      "1994 - 2000"

#include "config.h"
#include "shared.h"

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

```

```

#include <sqludf.h>           // guojian@0725
#include <sqlutil.h>         // guojian@0725

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

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

#define NONE      -1
#define PART      0
#define PSUPP     1
#define SUPP      2
#define CUST      3
#define ORDER     4
#define LINE      5
#define ORDER_LINE 6
#define PART_PSUPP 7
#define NATION    8
#define REGION    9
#define UPDATE    10
#define MAX_TABLE 11
#define ONE_STREAM 1
#define ADD_AT_END 2

#ifdef MAX
#undef MAX
#endif
#ifdef MIN
#undef MIN
#endif
#define MAX(a,b) ((a > b)?a:b)
#define MIN(A,B) ((A) < (B) ? (A) : (B))

#define INTERNAL_ERROR(p) {fprintf(stderr,"%s", p);abort();}
#define LN_CNT 4
static char lnoise[4] = {'l', 'r', 't', 'n'};
#define LIFENOISE(n, var) \
    if (verbose > 0) fprintf(stderr, "%c\b", lnoise[(var%LN_CNT)])

#define MALLOC_CHECK(var) \
    if ((var) == NULL) \
    { \
        fprintf(stderr, "Malloc failed at %s:%d\n", \
            __FILE__, __LINE__); \
        exit(1);\
    }
#define OPEN_CHECK(var, path) \
    if ((var) == NULL) \
    { \
        fprintf(stderr, "Open failed for %s at %s:%d\n", \
            path, __FILE__, __LINE__); \
        exit(1);\
    }
#ifndef MAX_CHILDREN
#define MAX_CHILDREN 1000
#endif

/*
 * macros that control sparse keys
 *
 * refer to Porting.Notes for a complete explanation
 */
#ifndef BITS_PER_LONG
#define BITS_PER_LONG 32
#define MAX_LONG      0x7FFFFFFF
#endif /* BITS_PER_LONG */
#define SPARSE_BITS    2
#define SPARSE_KEEP    3
#define MK_SPARSE(key, seq) \

```

```

((((key>>3)<<2))(seq & 0x0003))<<3))(key & 0x0007))

#define RANDOM(tgt, lower, upper, stream)    dss_random(&tgt, lower,
upper, stream)

typedef struct
{
    long    weight;
    char    *text;
}    set_member;

typedef struct
{
    int    count;
    int    max;
    set_member *list;
    long *permute;
}    distribution;
/*
 * some handy access functions
 */
#define DIST_SIZE(d)        d->count
#define DIST_MEMBER(d, i)    ((set_member *)((d)->list + i))->text

typedef struct
{
    char    *name;
    char    *comment;
    long    base;
    int    (*header) ();
    int    (*loader[2]) ();
    long    (*gen_seed)();
    int    (*verify) ();
    int    child;
    unsigned long vttotal;
}    tdef;

typedef struct SEED_T {
    long table;
    long value;
    long usage;
    long boundary;
} seed_t;

#if defined(__STDC__)
#define PROTO(s) s
#else
#define PROTO(s) ()
#endif

/* bm_utils.c */
char    *env_config PROTO((char *var, char *dflt));
long    yes_no PROTO((char *prompt));
int    a_rnd PROTO((int min, int max, int column, char *dest));
int    tx_rnd PROTO((long min, long max, long column, char *tgt));
long    julian PROTO((long date));
long    unjulian PROTO((long date));
FILE    *tbl_open PROTO((int tbl, char *mode));
long    dssncasecmp PROTO((char *s1, char *s2, int n));
long    dsscascmp PROTO((char *s1, char *s2));
int    pick_str PROTO((distribution * s, int c, char *target));
void    agg_str PROTO((distribution *set, long count, long col, char
*dest));
void    read_dist PROTO((char *path, char *name, distribution * target));
void    embed_str PROTO((distribution *d, int min, int max, int stream,
char *dest));
#endif
STDLIB_HAS_GETOPT

```

```

int    getopt PROTO((int arg_cnt, char **arg_vect, char *oprions));
#endif /* STDLIB_HAS_GETOPT */
long    set_state PROTO((int t, long scale, long procs, long step, long *e));

/* md.c */
long    NextRand PROTO((long nSeed));
long    UnifInt PROTO((long nLow, long nHigh, long nStream));
double    UnifReal PROTO((double dLow, double dHigh, long nStream));
double    Exponential PROTO((double dMean, long nStream));
void    dss_random(long *tgt, long min, long max, long seed);
void    row_start(int t);
void    row_stop(int t);
void    dump_seeds(int t);

/* text.c */
#define MAX_GRAMMAR_LEN 12    /* max length of grammar
component */
#define MAX_SENT_LEN    256 /* max length of populated sentence */
#define RNG_PER_SENT    27    /* max number of RNG calls per
sentence */

int    dbg_text PROTO((char * t, int min, int max, int s));

#ifndef DECLARER
#define EXTERN
#else
#define EXTERN extern
#endif    /* DECLARER */

EXTERN distribution nations;
EXTERN distribution nations2;
EXTERN distribution regions;
EXTERN distribution o_priority_set;
EXTERN distribution l_instruct_set;
EXTERN distribution l_smode_set;
EXTERN distribution l_category_set;
EXTERN distribution l_rflag_set;
EXTERN distribution c_mseg_set;
EXTERN distribution colors;
EXTERN distribution p_types_set;
EXTERN distribution p_cntr_set;

/* distributions that control text generation */
EXTERN distribution articles;
EXTERN distribution nouns;
EXTERN distribution adjectives;
EXTERN distribution adverbs;
EXTERN distribution prepositions;
EXTERN distribution verbs;
EXTERN distribution terminators;
EXTERN distribution auxillaries;
EXTERN distribution np;
EXTERN distribution vp;
EXTERN distribution grammar;

EXTERN long scale;
EXTERN int refresh;
EXTERN int resume;
EXTERN long verbose;
EXTERN long force;
EXTERN long header;
EXTERN long columnar;
EXTERN long direct;
EXTERN long updates;
EXTERN long table;
EXTERN long children;
EXTERN long fnames;
EXTERN int gen_sql;

```

```

EXTERN int gen_rng;
EXTERN char *db_name;
EXTERN int step;
EXTERN int set_seeds;
EXTERN int validate;
EXTERN char *d_path;

/* added for segmented updates */
EXTERN int insert_segments;
EXTERN int delete_segments;
EXTERN int insert_orders_segment;
EXTERN int insert_lineitem_segment;
EXTERN int delete_segment;

/*****
*****/
/* guojian@0725 part_info for each table */

EXTERN struct sqlupi *part_info_for_part;
EXTERN struct sqlupi *part_info_for_supplier;
EXTERN struct sqlupi *part_info_for_partsupp;
EXTERN struct sqlupi *part_info_for_customer;
EXTERN struct sqlupi *part_info_for_orders;
EXTERN struct sqlupi *part_info_for_lineitem;
EXTERN int me;
EXTERN int Numnodes;
/*****
*****/

#ifndef DECLARER
extern tdef tdefs[];

#endif /* DECLARER */

/*****
*****/
** table level defines use the following naming convention: t_ccc_xxx
** with: t, a table identifier
** ccc, a column identifier
** xxx, a limit type

*****/
*/

/*
* defines which control the parts table
*/
#define P_SIZE 126
#define P_NAME_SCL 5
#define P_MFG_TAG "Manufacturer#"
#define P_MFG_FMT "%s%01d"
#define P_MFG_MIN 1
#define P_MFG_MAX 5
#define P_BRND_TAG "Brand#"
#define P_BRND_FMT "%s%02d"
#define P_BRND_MIN 1
#define P_BRND_MAX 5
#define P_SIZE_MIN 1
#define P_SIZE_MAX 50
#define P_MCST_MIN 100
#define P_MCST_MAX 99900
#define P_MCST_SCL 100.0
#define P_RCST_MIN 90000
#define P_RCST_MAX 200000
#define P_RCST_SCL 100.0
/*
* defines which control the suppliers table
*/

```

```

#define S_SIZE 145
#define S_NAME_TAG "Supplier#"
#define S_NAME_FMT "%s%09ld"
#define S_ABAL_MIN -99999
#define S_ABAL_MAX 999999
#define S_CMNT_MAX 101
#define S_CMNT_BBB 10 /* number of BBB comments/SF */
#define BBB_DEADBEATS 50 /* % that are complaints */
#define BBB_BASE "Customer "
#define BBB_COMPLAIN "Complaints"
#define BBB_COMMEND "Recommends"
#define BBB_CMNT_LEN 19
#define BBB_BASE_LEN 9
#define BBB_TYPE_LEN 10

/*
* defines which control the partsupp table
*/
#define PS_SIZE 145
#define PS_SKEY_MIN 0
#define PS_SKEY_MAX ((tdefs[SUPP].base - 1) * scale)
#define PS_SCST_MIN 100
#define PS_SCST_MAX 100000
#define PS_QTY_MIN 1
#define PS_QTY_MAX 9999
/*
* defines which control the customers table
*/
#define C_SIZE 165
#define C_NAME_TAG "Customer#"
#define C_NAME_FMT "%s%09d"
#define C_MSEG_MAX 5
#define C_ABAL_MIN -99999
#define C_ABAL_MAX 999999
/*
* defines which control the order table
*/
#define O_SIZE 109
#define O_CKEY_MIN 1
#define O_CKEY_MAX (long)(tdefs[CUST].base * scale)
#define O_ODATE_MIN STARTDATE
#define O_ODATE_MAX (STARTDATE + TOTDATE - \
(L_SDTE_MAX + L_RDTE_MAX) - 1)
#define O_CLRK_TAG "Clerk#"
#define O_CLRK_FMT "%s%09d"
#define O_CLRK_SCL 1000
#define O_LCNT_MIN 1
#define O_LCNT_MAX 7

/*
* defines which control the lineitem table
*/
#define L_SIZE 144L
#define L_QTY_MIN 1
#define L_QTY_MAX 50
#define L_TAX_MIN 0
#define L_TAX_MAX 8
#define L_DCNT_MIN 0
#define L_DCNT_MAX 10
#define L_PKEY_MIN 1
#define L_PKEY_MAX (tdefs[PART].base * scale)
#define L_SDTE_MIN 1
#define L_SDTE_MAX 121
#define L_CDTE_MIN 30
#define L_CDTE_MAX 90
#define L_RDTE_MIN 1
#define L_RDTE_MAX 30
/*
* defines which control the time table
*/

```

```

#define T_SIZE    30
#define T_START_DAY 3 /* wednesday ? */
#define LEAP(y) ((!(y % 4) && (y % 100)) ? 1 : 0)

/*****
*****

*****
*****
*** general or inter table defines
***

*****
*****

*****/
#define SUPP_PER_PART 4
#define ORDERS_PER_CUST 10 /* sync this with CUST_MORTALITY
*/
#define CUST_MORTALITY 3 /* portion with have no orders */
#define NATIONS_MAX 90 /* limited by country codes in phone
numbers */
#define PHONE_FMT "%02d-%03d-%03d-%04d"
#define STARTDATE 92001
#define CURRENTDATE 95168
#define ENDDATE 98365
#define TOTDATE 2557
#define UPD_PCT 10
#define MAX_STREAM 47
#define V_STR_LOW 0.4
#define PENNIES 100 /* for scaled int money arithmetic */
#define Q11_FRACTION (double)0.0001
/*
* max and min SF in GB; Larger SF will require changes to the build
routines
*/
#define MIN_SCALE 1.0
#define MAX_SCALE 1000.0
/*
* beyond this point we need to allow for BCD calculations
*/
#define MAX_32B_SCALE 1000.0
#define INIT_HUGE(v) { \
    v = (DSS_HUGE *)malloc(sizeof(DSS_HUGE) *
HUGE_COUNT); \
    MALLOC_CHECK(v); \
}
#define FREE_HUGE(v) free(v)
#ifdef SUPPORT_64BITS
#define LONG2HUGE(src, dst) *dst = (DSS_HUGE)src
#define HUGE2LONG(src, dst) *dst = (long)src
#define HUGE_SET(src, dst) *dst = *src
#define HUGE_MUL(op1, op2) *op1 *= op2
#define HUGE_DIV(op1, op2) *op1 /= op2
#define HUGE_ADD(op1, op2, dst) *dst = *op1 + op2
#define HUGE_SUB(op1, op2, dst) *dst = *op1 - op2
#define HUGE_MOD(op1, op2) *op1 % op2
#define HUGE_CMP(op1, op2) (*op1 == *op2) ? 0 : (*op1 < *op2) -
1:1
#else
#define LONG2HUGE(src, dst) { *dst = src; *(dst + 1) = 0; }
#define HUGE2LONG(src, dst) { dst = 0; \
    bcd2_bin(dst, (src + 1)); \
    bcd2_bin(dst, src); }
#define HUGE_SET(src, dst) { *dst = *src; *(dst + 1) = *(src + 1);
}
#define HUGE_MUL(op1, op2) bcd2_mul(op1, (op1 + 1), op2)
#define HUGE_DIV(op1, op2) bcd2_div(op1, (op1 + 1), op2)

```

```

#define HUGE_ADD(op1, op2, d) { \
    HUGE_SET(op1, d); \
    bcd2_add(d, (d + 1), op2); \
}
#define HUGE_SUB(op1, op2, d) { \
    HUGE_SET(op1, d); \
    bcd2_sub(d, (d + 1), op2); \
}
#define HUGE_MOD(op1, op2) bcd2_mod(op1, (op1 + 1), op2)
#define HUGE_CMP(op1, op2) (bcd2_cmp(op1, (op1 + 1), op2)
== 0) ? 0 : \
    ((bcd2_cmp(op1, (op1 + 1), op2) < 0) ? -1 : 1)
#endif /* SUPPORT_64BITS */

/***** environmental variables and defaults *****/
#define DIST_TAG "DSS_DIST" /* environment var to override ...
*/
#define DIST_DFLT "dists.dss" /* default file to hold distributions */
#define PATH_TAG "DSS_PATH" /* environment var to override
... */
#define PATH_DFLT "." /* default directory to hold tables
*/
#define CONFIG_TAG "DSS_CONFIG" /* environment var to
override ... */
#define CONFIG_DFLT "." /* default directory to config
files */
#define ADHOC_TAG "DSS_ADHOC" /* environment var to
override ... */
#define ADHOC_DFLT "adhoc.dss" /* default file name for adhoc
vars */

/***** output macros *****/
#ifndef SEPARATOR
#define SEPARATOR '|' /* field separator for generated flat files */
#endif
/* Data type flags for a single print routine */
#define DT_STR 0
#ifdef MVS
#define DT_VSTR DT_STR
#else
#define DT_VSTR 1
#endif /* MVS */
#define DT_INT 2
#define DT_HUGE 3
#define DT_KEY 4
#define DT_MONEY 5
#define DT_CHR 6

int dbg_print(int dt, FILE *tgt, void *data, int len, int eol);
#define PR_STR(f, str, len) dbg_print(DT_STR, f, (void *)str, len,
1)
#define PR_VSTR(f, str, len) dbg_print(DT_VSTR, f, (void *)str, len, 1)
#define PR_VSTR_LAST(f, str, len) dbg_print(DT_VSTR, f, (void
*)str, len, 0)
#define PR_INT(f, str) dbg_print(DT_INT, f, (void *)str, 0, 1)
#define PR_HUGE(f, str) dbg_print(DT_HUGE, f, (void *)str, 0,
1)
#define PR_KEY(f, str) dbg_print(DT_KEY, f, (void *)str, 0, -1)
#define PR_MONEY(f, str) dbg_print(DT_MONEY, f, (void
*)str, 0, 1)
#define PR_CHR(f, str) dbg_print(DT_CHR, f, (void *)str, 0, 1)
#define PR_STRT(fp) /* any line prep for a record goes here */
#define PR_END(fp) fprintf(fp, "\n") /* finish the record here */
#ifdef MDY_DATE
#define PR_DATE(tgt, yr, mn, dy) \
    sprintf(tgt, "%02d-%02d-19%02d", mn, dy, yr)
#else
#define PR_DATE(tgt, yr, mn, dy) \
    sprintf(tgt, "19%02d-%02d-%02d", yr, mn, dy)
#endif /* DATE_FORMAT */

```

```

/*
 * verification macros
 */
#define VRF_STR(t, d) {char *xx = d; while (*xx) tdefs[t].vtotal +=
*xx++;}
#define VRF_INT(t,d) tdefs[t].vtotal += d
#define SUPPORT_64BITS
#define VRF_HUGE(t,d) tdefs[t].vtotal = *((long *)&d) + *((long *)&d
+ 1))
#define VRF_HUGE(t,d) tdefs[t].vtotal += d[0] + d[1]
#define SUPPORT_64BITS */
/* assume float is a 64 bit quantity */
#define VRF_MONEY(t,d) tdefs[t].vtotal = *((long *)&d) + *((long *)&d
+ 1))
#define VRF_CHR(t,d) tdefs[t].vtotal += d
#define VRF_STRT(t)
#define VRF_END(t)

/***** distribuitons currently defined *****/
#define UNIFORM 0

/*
 * seed indexes; used to separate the generation of individual columns
 */
#define P_MFG_SD 0
#define P_BRND_SD 1
#define P_TYPE_SD 2
#define P_SIZE_SD 3
#define P_CNTR_SD 4
#define P_RCST_SD 5
#define PS_QTY_SD 7
#define PS_SCST_SD 8
#define O_SUPP_SD 10
#define O_CLRK_SD 11
#define O_ODATE_SD 13
#define L_QTY_SD 14
#define L_DCNT_SD 15
#define L_TAX_SD 16
#define L_SHIP_SD 17
#define L_SMODE_SD 18
#define L_PKEY_SD 19
#define L_SKEY_SD 20
#define L_SDTE_SD 21
#define L_CDTE_SD 22
#define L_RDTE_SD 23
#define L_RFLG_SD 24
#define C_NTRG_SD 27
#define C_PHNE_SD 28
#define C_ABAL_SD 29
#define C_MSEG_SD 30
#define S_NTRG_SD 33
#define S_PHNE_SD 34
#define S_ABAL_SD 35
#define P_NAME_SD 37
#define O_PRIO_SD 38
#define HVAR_SD 39
#define O_CKEY_SD 40
#define N_CMNT_SD 41
#define R_CMNT_SD 42
#define O_LCNT_SD 43
#define BBB_JNK_SD 44
#define BBB_TYPE_SD 45
#define BBB_CMNT_SD 46
#define BBB_OFFSET_SD 47
#define DSS_H */

```

B.20 dsstypes.h

```

/*
 * Sccsid:  @(#)dsstypes.h 2.1.8.1
 *
 * general definitions and control information for the DSS data types
 * and function prototypes
 */

/* guojian@0725 structure for wrapping those parameters passed into
sqlugrpn */

```

en excuses use0 q Á	°	ÔĎé		NO	RŠ	ĴŠ	bŠ
------------------------	---	-----	--	----	----	----	----

ô NO RŠ ”Š [Š | NONE TRUCK | K ironic

dŠ COLLECT COD REG AIR | ” L ruthless foxes accord0 q em G

p , NO RŠ HŠ | YŠ COLLECT COD RAIL

ole. regular

| th0 q ‡q -) ¢ à“ ? ô

RAIL ‘ N furiously express 0 q Á, ... | 2 P4 ,ÿÃ ,

, even | excuses0 q ,Ä ð f _ X~0 d

Æ S blithely
even0 q " ½ 4 @œ °°

TRUCK Š T furiously | u0 q &C 7 Å

NO RŠ bŠ UŠ DELIVER IN | PERSON REG AIR “ U bold requests

s detect care0 q ã´ `.	,	– NO	İÄ´ RŠ	DŠ	VŠ	NONE	REG AIR
------------------------	---	------	-----------	----	----	------	---------

)	-	Ø)	S X	@ ,	NO	RŠ		MŠ	iŠ	NONE	FOB
---	---	----	--------	--------	----	----	--	----	----	------	-----

Š X slyly | brave0 q ¥Ó ¶B p @~ 0 Z d

TRUCK

| Y

furiously special accounts

| cajole

à 0eŠ! È NO RŠ | GŠ kŠ COLLECT COD SHIP

~ Z requests wake quickly. | re

|

» ^ ‹ a Î D , (• ì W Æ \$ ê Q ¹) † ù V ¶

F

1
2

ê
N

| © k Õ 1 " _ Ò ; i o í ' Š ì `

FŠ 3) H | i Ð%p+ , d NO RŠ

FOB	CE \	quickly		ironi0 q cT`	@œ	%kô	ô
-----	------	---------	--	--------------	----	-----	---

ross the slyly | exp0 q □ J è @~ @êa(d

dŠ	COLLECT COD	FOB		s0	q	▣	”	Ÿ	–	carefully even accounts cajole
----	-------------	-----	--	----	---	---	---	---	---	--------------------------------

6		NO	RŠ	”Š	\Š	NONE
---	--	----	----	----	----	------

RŠ NŠ jŠ DELIVER IN PERSON | FOB • a slyly express

b doggedly | silen0 q f® D' A @~ Đ²▯

à4		¼	NO	RŠ	”Š	mŠ
----	--	---	----	----	----	----

• e final requests wake; | s0 q Āí w= à Đl ” ^ „

TRUCK %o f furiously 0 q b ©- | n @~ à½

@Š

fŠ

DELIVER IN

AIR										q	Cn	e%	>	0u	Š	¬ ^{ai} _i	ZŠ	COLLECT	È
æ g blithely special										COD		NO	RŠ		Š	s	blithely special packages		
y special dependencies0 q † uE G Đû										print0	q	è	'7	ã	@				
0 q † uE G Đû																			
SHIP IP inst0 q #[μ F																			
[μ F																			
F																			
*i , È NO RŠ rŠ																			
sleep0 q á ç/ (Pç p																			
fur0 q a 5I																			
regular pac0 q ‡Û Z4 š P¥ P¥ l ¼ ¼ d																			
P¥ l ¼ d NO																			
l ¼ d NO RŠ																			
d NO RŠ Š IŠ COL																			
O RŠ Š IŠ COLLECT COD REG																			
AIR de0 q G# E o RŠ ZŠ iŠ "COLLECT																			
COD AIR ċ m slyly regular depths wake:																			
fluffily0 q ä 2' + NO RŠ ^Š 0¶A bŠ X NON																			
E FOB í ± °- idly even																			
plate0 q &™ í ± °- àë-																			
el0 q &™ í ± °- àë-																			
X NO RŠ																			
Š ZŠ TAKE BACK RETURN RAIL																			
o quickly even d0 q â— Ô -																			
P4 Å È d NO																			
O RŠ nŠ bŠ DELIVER IN PERSON																			
MAIL dolph0 q â H= Ũ q <K TŠ TAK																			
E BACK RETURN FOB £ q unusual, ironic																			
0 q â × ì à°—																			
eŠ NONE RAIL r bold																			
asymptotes u0 q Cn e% > 0u ¬ ^{ai} _i																			

t	slyly final		instruction0	q	fô	5	c	0æ
---	-------------	--	--------------	---	----	---	---	----

NO RŠ DŠ eŠ TAKE | BACK RETURN AIR Ÿ u pending dugouts

| " w slyly ironic

2

	yŠ		aŠ	TAKE BACK RETURN	AIR
	•	z	quickly blithe		
package0	q	...¹	Æ@	s	pó
	X		NO	RŠ	øđÖ"

Š	lŠ	COLLECT COD	MAIL	maint0	q	ç¼	r	NO	RŠ	ˆ	”	È
	• {	regular deposits		RAIL	,		–		carefully regular attai0	q	ÄË	Š
				w		ô	@œ	0-	NO	RŠ		HŠ
				IŠ		NONE		AIR		™	}	blithely

express

multiplie0 q	ÄU	□/	È		đ°	Öüy	È
¼		NO		RŠ	Š	cŠ	CO
LLECT COD		RAIL		™	~	furiously	unusual
tithes a0 q	aN	D	Ä				
0	6I	È				NO	RŠ

+Š	TŠ	COLLECT COD	REG AIR
	NO	RŠ	UŠ gŠ NONE
	-	idly pending pinto bean0 q āb	¥:
	â	L ³	NO

RŠ	5Š	YŠ	TAKE BACK RETURN	AIR	¥	accounts haggle fluffily according to
				0 q `ç	½- f	İ
					NO	İL È
				RETURN	MAIL	RŠ :Š VŠ TAKE BACK
				e0 q j	½	, regular foxes
						Èb½&

X	NO	RŠ	QŠ	_Š	COLLE	CT COD fu0 q Āê	FOB) . NO RŠ cŠ DELIVER IN PERSON • „ furiously special	Œ f â	frets wake ØM	d
---	----	----	----	----	-------	--------------------	--	----------	------------------	---

pint0 q âm - Ç @~ `İ°" è		NO	RŠ	8Š	lŠ	NONE
	TRUCK		...	silent asymptotes are carefully.		
	0 q âm ^		à	H~\$#	d	¼
		NO	RŠ	/Š	bŠ	COLLECT COD
	TRUCK		†	ideas above the		

s0 q	x	‘	PŸ	lâ (	d	d COD	NO SHIP	RŠ	†Š ‰ ‡	bŠ even, regu0	COLLECT q @đ
------	---	---	----	------	---	----------	------------	----	-----------	-------------------	-----------------

M ÀÔ @gÃ RŠ Š final,

_Š COLLECT COD RAIL a ^

express pinto beans hinder

furiously0 q "¢ í\$ S I & hŠ D

È NO RŠ #Š £ %o finally

ELIVER IN PERSON SHIP express pinto beans cajole f0 q Ç

K	R	8	ß	È	È	RETURN	MAIL	' Š	slyly careful	
NO	RŠ	"Š	fŠ	TAKE BACK	gifts0 q	‡6 æ	à"	NO	RŠ	PŠ
					i	UŠ	DELIVER IN PERSON	FOB		
						' <	ironic, final			
					asym0 q	5 Ç				
					i	Ńc0	,		NO	RŠ
					OŠ	oŠ	DELIVER IN PERSON	FOB		

Œ	special, even	r0	q	%+	NO	RŠ	đI	eŠ	Óx	UŠ	è	NONE
		d				—						
		FOB						blithely bold accounts				
		h										

z ì [É & | ó c Ã . ¢ o Û N µ † ñ \ Ð E
± u ß

A

| ±

...

< A d Ð+ ¼

NO	RŠ	0Š	SŠ	DELIVER IN PERSON
RAIL		Ÿ	unusual packages are blithely	
ir0 q d5 ‡% c		P4	ðµû ¼	
X	NO	RŠ	0Š	jŠ COLLECT
COD	REG AIR		— ‘	furiously unusual
reques0 q cs H		đI	üÿB	

NO RŠ =Š eŠ TAKE BACK RETURN
SHIP © ' carefully regular deposits cajole
blithely0 q Ao ^" e ÄÖ Ä8,

NO RŠ :Š bŠ N ONE SHIP Ć “ regular
warho0 q D; K4 ” ÷ pú&
¼ NO RŠ Š eŠ TAKE
BACK RETURN SHIP ” quickly thin
acco0 q ÆM 2? Ç \ X

Y | X , NO RŠ |Š

cŠ TAKE BACK RETURN FOB

• final

packages cajole. unusual
ac0 q ĸT û \ PÃ øµA è
NO RŠ „Š pŠ NONE
FOB £ – ideas sublate slyly. even, bold
depo0 q Å ›3 °_ <š

NO	RŠ	\$Š	nŠ	TAKE BACK RETURN	TRUCK	Œ —	even
				platelet0 q Å V			
				‡	İ	h·E	ô
	NO	RŠ		– ~	„Š carefully bold	SŠ	X NONE
MAIL				– ~			
packages0 q G				‘2 Đ		†	Đý
¼							

NO	RŠ	7Š	iŠ	DELIVER IN	PERSON thinl0 q f•	SHIP)2 t	‘ TM	requests nag
					REG AIR	NO	Đl Š	àÓP- ô
					Ī	†	œ š à	fŠ NONE
							instructions 0 q iŮ	¼

NO RŠ

Š ZŠ TAKE BACK RETURN AIR

bl0 q Ÿ ō/ carefully ironic instructions sleep.

NO RŠ – 8Š ìXZ dŠ ” DELIVER

IN PERSON SHIP %o æ furiously

0 q „p | O ÂÔ P+¶ , | X NO RŠ DŠ oŠ NONE

X NO RŠ ^Š pŠ | NONE RAIL bold, regular

deposits across | the0 q ãñ PL X `ê Ø“4 È

wake slyly

| i0 q F • · PĀ pŠ° „

y regular thead0	q	D ⁻	ü	u	p,	õ,	X				NO	RŠ	wŠ	iŠ	DELI
---------------------	---	----------------	---	---	----	----	---	--	--	--	----	----	----	----	------

| ing, unusual accounts haggle fur0 q ã

, [(– | ‘ NO RŠ [Š

YŠ COLLECT COD REG AIR | • £ regular foxes kindle

€ □ slyly regular dependencies cajole | b0 q Å 74 p, ps ,

X NO RŠ +Š _Š DELIVER | IN PERSON REG AIR • ¥ quickly ironic

platele0 q B | ½C a đ° ;@ È

MAIL ” | pending accounts | acro0 q ` èH ¥ p ~vš ô

È NO RŠ aŠ TŠ COLLE | CT COD AIR ' § furiously final

dep0 q □

| D ¼ °- X~Æ è ô

RAIL “ ” final theodolites | ha0 q Æ ¾= ' 0u p-

accounts sleep.

| blithely0 q & { : Ì z t 7 d

| the0 q , - ¾ q jã

%o « silent | req0 q f ñ3 1 z ´ Y#

ar, even hockey players cajole | sly0 q nF C à“ ¼ó

SŠ TAKE BACK RETURN FOB | - final, pending ideas nag

@ | pÅ d NO RŠ

```
aref
tpcd.customer,
tpcd.orders,
tpcd.lineitem,
tpcd.supplier,
tpcd.nation,
tpcd.region
where
c_custkey = o_custkey
and o_orderkey = l_orderkey
and l_suppkey = s_suppkey
and c_nationkey = s_nationkey
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = 'ASIA'
and o_orderdate >= date ('1994-01-01')
and o_orderdate < date ('1994-01-01') + 1 year
group by
n_name
order by
revenue desc
```

N_NAME	REVENUE
INDONESIA	55502041.170
VIETNAM	55295086.997
CHINA	53724494.257
INDIA	52035512.000
JAPAN	45410175.695

Number of rows retrieved is: 5

Qualification Query 6

-- Query 06 - Var_0 Rev_01 - Forecasting Revenue Change Query

Tag: Q6 Stream: -1 Sequence number: 5

```
select
sum(l_extendedprice * l_discount) as revenue
from
tpcd.lineitem
where
l_shipdate >= date ('1994-01-01')
and l_shipdate < date ('1994-01-01') + 1 year
and l_discount between .06 - 0.01 and .06 + 0.01
and l_quantity < 24
```

REVENUE
123141078.228

Number of rows retrieved is: 1

Qualification Query 7

-- Query 07 - Var_0 Rev_01 - Volume Shipping Query

Tag: Q7 Stream: -1 Sequence number: 21

```
select
supp_nation,
```

```
cust_nation,
l_year,
sum(volume) as revenue
from
(
select
n1.n_name as supp_nation,
n2.n_name as cust_nation,
year (l_shipdate) as l_year,
l_extendedprice * (1 - l_discount) as volume
from
tpcd.supplier,
tpcd.lineitem,
tpcd.orders,
tpcd.customer,
tpcd.nation n1,
tpcd.nation n2
where
s_suppkey = l_suppkey
and o_orderkey = l_orderkey
and c_custkey = o_custkey
and s_nationkey = n1.n_nationkey
and c_nationkey = n2.n_nationkey
and (
(n1.n_name = 'FRANCE' and n2.n_name = 'GERMANY')
or (n1.n_name = 'GERMANY' and n2.n_name = 'FRANCE')
)
and l_shipdate between date('1995-01-01') and date('1996-12-31')
) as shipping
group by
supp_nation,
cust_nation,
l_year
order by
supp_nation,
cust_nation,
l_year
```

SUPP_NATION REVENUE	CUST_NATION	L_YEAR
FRANCE 54639732.734	GERMANY	1995
FRANCE 54633083.308	GERMANY	1996
GERMANY 52531746.670	FRANCE	1995
GERMANY 52520549.022	FRANCE	1996

Number of rows retrieved is: 4

Qualification Query 8

-- Query 08 - Var_0 Rev_01 - National Market Share Query

Tag: Q8 Stream: -1 Sequence number: 8

```
select
o_year,
sum(case
when nation = 'BRAZIL' then volume
else 0
end) / sum(volume) as mkt_share
from
(
```

```
select
year(o_orderdate) as o_year,
l_extendedprice * (1 - l_discount) as volume,
n2.n_name as nation
from
tpcd.part,
tpcd.supplier,
tpcd.lineitem,
tpcd.orders,
tpcd.customer,
tpcd.nation n1,
tpcd.nation n2,
tpcd.region
where
p_partkey = l_partkey
and s_suppkey = l_suppkey
and l_orderkey = o_orderkey
and o_custkey = c_custkey
and c_nationkey = n1.n_nationkey
and n1.n_regionkey = r_regionkey
and r_name = 'AMERICA'
and s_nationkey = n2.n_nationkey
and o_orderdate between date('1995-01-01') and date('1996-12-31')
and p_type = 'ECONOMY ANODIZED STEEL'
) as all_nations
group by
o_year
order by
o_year
```

O_YEAR	MKT_SHARE
1995	0.034
1996	0.041

Number of rows retrieved is: 2

Qualification Query 9

-- Query 09 - Var_0 Rev_01 - Product Type Profit Measure Query

Tag: Q9 Stream: -1 Sequence number: 3

```
select
nation,
o_year,
sum(amount) as sum_profit
from
(
select
n_name as nation,
year(o_orderdate) as o_year,
l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity as amount
from
tpcd.part,
tpcd.supplier,
tpcd.lineitem,
tpcd.partsupp,
tpcd.orders,
tpcd.nation
where
s_suppkey = l_suppkey
and ps_suppkey = l_suppkey
and ps_partkey = l_partkey
and p_partkey = l_partkey
and o_orderkey = l_orderkey
```

```

and s_nationkey = n_nationkey
and p_name like '%green%'
) as profit
group by
nation,
o_year
order by
nation,
o_year desc

```

NATION	O_YEAR	SUM_PROFIT
ALGERIA	1998	31342867.235
ALGERIA	1997	57138193.023
ALGERIA	1996	56140140.133
ALGERIA	1995	53051469.653
ALGERIA	1994	53867582.129
ALGERIA	1993	54942718.132
ALGERIA	1992	54628034.713
ARGENTINA	1998	30211185.708
ARGENTINA	1997	50805741.752
ARGENTINA	1996	51923746.575
ARGENTINA	1995	49298625.767
ARGENTINA	1994	50835610.109
ARGENTINA	1993	51646079.178
ARGENTINA	1992	50410314.995
.... Additional rows deleted		
UNITED STATES	1993	48029946.801
UNITED STATES	1992	48671944.498
VIETNAM	1998	30442736.059
VIETNAM	1997	50309179.794
VIETNAM	1996	50488161.410
VIETNAM	1995	49658284.612
VIETNAM	1994	50596057.261
VIETNAM	1993	50953919.152
VIETNAM	1992	49613838.315

Number of rows retrieved is: 175

Qualification Query 10

-- Query 10 - Var_0 Rev_01 - Returned Item Reporting Query

Tag: Q10 Stream: -1 Sequence number: 18

```

select
c_custkey,
c_name,
sum(l_extendedprice * (1 - l_discount)) as revenue,
c_acctbal,
n_name,
c_address,
c_phone,
c_comment
from
tpcd.customer,
tpcd.orders,
tpcd.lineitem,
tpcd.nation
where
c_custkey = o_custkey
and l_orderkey = o_orderkey
and o_orderdate >= date ('1993-10-01')

```

```

and o_orderdate < date ('1993-10-01') + 3 month
and l_returnflag = 'R'
and c_nationkey = n_nationkey
group by
c_custkey,
c_name,
c_acctbal,
c_phone,
n_name,
c_address,
c_comment
order by
revenue desc
fetch first 20 rows only

```

C_CUSTKEY	C_NAME	REVENUE	C_ACCTBAL
N_NAME	C_ADDRESS		C_PHONE
C_COMMENT			

57040	Customer#000057040	734235.245	632.870
JAPAN	Eioyzjf4pp	22-895-641-3466	
requests sleep blithely about the furiously i			
143347	Customer#000143347	721002.695	2557.470
EGYPT	1aReFYv,Kw4	14-742-935-3718	
fluffily bold excuses haggle finally after the u			
60838	Customer#000060838	679127.308	2454.770
BRAZIL	64EaJ5vMAHWJIBOXklpNc2RjiWE	12-913-	
494-9813 furiously even pinto beans integrate under the ruthless foxes;			
ironic, even dolphins across the slyl			
101998	Customer#000101998	637029.567	3790.890
UNITED KINGDOM	01c9CILnNtfOQYmZj	33-593-	
865-6378 accounts doze blithely! enticing, final deposits sleep blithely			
special accounts. slyly express accounts pla			
125341	Customer#000125341	633508.086	4983.510
GERMANY	S29ODD6bceU8QSuuEJznkNaK	17-	
582-695-5962 quickly express requests wake quickly blithely			
25501	Customer#000025501	620269.785	7725.040
ETHIOPIA	W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ		
15-874-808-6793 quickly special requests sleep evenly among the special			
deposits. special deposi			
115831	Customer#000115831	596423.867	5098.100
FRANCE	rFeBbEEyk dl ne7zV5fDrmiq1oK09wV7pxqCgIc		
16-715-386-3788 carefully bold excuses sleep alongside of the thinly idle			
84223	Customer#000084223	594998.024	528.650
UNITED KINGDOM	nAVZCs6BaWap rrM27N 2qBnzc5WBauxbA		
33-442-824-8191 pending, final ideas haggle final requests. unusual, regular			
asymptotes affix according to the even foxes.			
54289	Customer#000054289	585603.392	5583.020
IRAN	vXCxoCsU0Bad5JQI ,oobkZ	20-834-292-	
4707 express requests sublate blithely regular requests. regular, even ideas			
solve.			
39922	Customer#000039922	584878.113	7321.110
GERMANY	Zgy4s50l2GKN4pLDPBU8m342gIw6R	17-	
147-757-8036 even pinto beans haggle. slyly bold accounts inte			
6226	Customer#000006226	576783.761	2230.090
UNITED KINGDOM	8gPu8,NPGkfyQQ0hcIYUGPIBWc,ybP5g,		
33-657-701-3391 quickly final requests against the regular instructions wake			
blithely final instructions. pa			
922	Customer#000000922	576767.533	3869.250
GERMANY	Az9RFaut7NkPnc5zSD2PwHgVvr4jRzq	17-	
945-916-9648 boldly final requests cajole blith			
147946	Customer#000147946	576455.132	2030.130
ALGERIA	iANyZHjqhy7Ajah0pTrYyhJ	10-886-	
956-3143 furiously even accounts are blithely above the furiouslyl			
115640	Customer#000115640	569341.193	6436.100
ARGENTINA	Vtgfia9qI 7EpHgecU1X	11-411-543-	
4901 final instructions are slyly according to the			

73606 Customer#000073606 568656.858 1785.670
JAPAN xuR0Tro5yChDfOCrjkd2ol 22-437-653-
6966 furiously bold orbits about the furiously busy requests wake across the
furiously quiet theodolites. d
110246 Customer#000110246 566842.981 7763.350
VIETNAM 7KzflgX MDOq7sOkI 31-943-426-
9837 dolphins sleep blithely among the slyly final
142549 Customer#000142549 563537.237 5085.990
INDONESIA ChqEoK43OysjdHbtKCp6dKqjNyvvi9 19-
955-562-2398 regular, unusual dependencies boost slyly; ironic attainments
nag fluffily into the unusual packages?
146149 Customer#000146149 557254.987 1791.550
ROMANIA s87fvzFQpU 29-744-164-6487
silent, unusual requests detect quickly slyly regul
52528 Customer#000052528 556397.351 551.790
ARGENTINA NFztyTOR10UOJ 11-208-192-
3205 unusual requests detect. slyly dogged theodolites use slyly. deposit
23431 Customer#000023431 554269.536 3381.860
ROMANIA HgiV0phqhaIa9aydNoIlb 29-915-458-
2654 instructions nag quickly. furiously bold accounts cajol

Number of rows retrieved is: 20

Qualification Query 11

-- Query 11 - Var_0 Rev_01 - Important Stock Identification Query

Tag: Q11 Stream: -1 Sequence number: 15

```
select
ps_partkey,
sum(ps_supplycost * ps_availqty) as value
from
tpcd.partsupp,
tpcd.supplier,
tpcd.nation
where
ps_suppkey = s_suppkey
and s_nationkey = n_nationkey
and n_name = 'GERMANY'
group by
ps_partkey having
sum(ps_supplycost * ps_availqty) > (
select
sum(ps_supplycost * ps_availqty) * 0.0001000000
from
tpcd.partsupp,
tpcd.supplier,
tpcd.nation
where
ps_suppkey = s_suppkey
and s_nationkey = n_nationkey
and n_name = 'GERMANY'
)
order by
value desc
```

PS_PARTKEY	VALUE
129760	17538456.860
166726	16503353.920
191287	16474801.970
161758	16101755.540
34452	15983844.720
139035	15907078.340

.... Additional rows removed

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

Number of rows retrieved is: 1048

Qualification Query 12

-- Query 12 - Var_0 Rev_02 - Shipping Modes and Order Priority Query

Tag: Q12 Stream: -1 Sequence number: 22

```
select
l_shipmode,
sum(case
when o_orderpriority = '1-URGENT'
or o_orderpriority = '2-HIGH'
then 1
else 0
end) as high_line_count,
sum(case
when o_orderpriority <> '1-URGENT'
and o_orderpriority <> '2-HIGH'
then 1
else 0
end) as low_line_count
from
tpcd.orders,
tpcd.lineitem
where
o_orderkey = l_orderkey
and l_shipmode in ('MAIL', 'SHIP')
and l_commitdate < l_receiptdate
and l_shipdate < l_commitdate
and l_receiptdate >= date ('1994-01-01')
and l_receiptdate < date ('1994-01-01') + 1 year
group by
l_shipmode
order by
l_shipmode
```

L_SHIPMODE	HIGH_LINE_COUNT	LOW_LINE_COUNT
MAIL	6202	9324
SHIP	6200	9262

Number of rows retrieved is: 2

Qualification Query 13

-- Query 13 - Var_0 Rev_01 - Customer Distribution Query

Tag: Q13 Stream: -1 Sequence number: 10

```

select
c_count,
count(*) as custdist
from
(
select
c_custkey,
count(o_orderkey)
from
tpcd.customer left outer join tpcd.orders on
c_custkey = o_custkey
and o_comment not like '%special%requests%'
group by
c_custkey
) as c_orders (c_custkey, c_count)
group by
c_count
order by
custdist desc,
c_count desc

```

C_COUNT	CUSTDIST
---------	----------

0	50004
9	6641
10	6566
11	6058
8	5949
12	5553
13	4989
19	4748
7	4707
18	4625
15	4552
17	4530
14	4484
20	4461
16	4323
21	4217
22	3730
6	3334
23	3129
24	2622
25	2079
5	1972
26	1593
27	1185
4	1033
28	869
29	559
3	398
30	373
31	235
2	144
32	128
33	71
34	48
35	33
1	23
36	17
37	7
40	4
38	4
39	2
41	1

Number of rows retrieved is: 42

Qualification Query 14

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

```
-- Query 14 - Var_0 Rev_01 - Promotion Effect Query
```

Tag: Q14 Stream: -1 Sequence number: 1

```

select
100.00 * sum(case
when p_type like 'PROMO%'
then l_extendedprice * (1 - l_discount)
else 0
end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
tpcd.lineitem,
tpcd.part
where
l_partkey = p_partkey
and l_shipdate >= date ('1995-09-01')
and l_shipdate < date ('1995-09-01') + 1 month

```

PROMO_REVENUE

16.381

Number of rows retrieved is: 1

Qualification Query 15

```
-- Query 15 - Var_a Rev_01 - Top Supplier Query
```

Tag: Q15a Stream: -1 Sequence number: 16

```

with revenue (supplier_no, total_revenue) as (
select
l_suppkey,
sum(l_extendedprice * (1-l_discount))
from
tpcd.lineitem
where
l_shipdate >= date ('1996-01-01')
and l_shipdate < date ('1996-01-01') + 3 month
group by
l_suppkey
)
select
s_suppkey,
s_name,
s_address,
s_phone,
total_revenue
from
tpcd.supplier,
revenue
where
s_suppkey = supplier_no
and total_revenue = (
select
max(total_revenue)
from
revenue
)
order by
s_suppkey

```

S_SUPPKEY	S_NAME	S_ADDRESS
S_PHONE	TOTAL_REVENUE	
8449	Supplier#000008449	Wp3zim9qYFbVctdW
20-469-856-8873	1772627.209	

Number of rows retrieved is: 1

Qualification Query 16

-- Query 16 - Var_0 Rev_01 - Parts/Supplier Relationship Query

Tag: Q16 Stream: -1 Sequence number: 13

```
select
p_brand,
p_type,
p_size,
count(distinct ps_suppkey) as supplier_cnt
from
tpcd.partsupp,
tpcd.part
where
p_partkey = ps_partkey
and p_brand <> 'Brand#45'
and p_type not like 'MEDIUM POLISHED%'
and p_size in (49, 14, 23, 45, 19, 3, 36, 9)
and ps_suppkey not in (
select
s_suppkey
from
tpcd.supplier
where
s_comment like '%Customer%Complaints%'
)
group by
p_brand,
p_type,
p_size
order by
supplier_cnt desc,
p_brand,
p_type,
p_size
```

P_BRAND	P_TYPE	P_SIZE	SUPPLIER_CNT
Brand#41	MEDIUM BRUSHED TIN	3	28
Brand#54	STANDARD BRUSHED COPPER	14	27
Brand#11	STANDARD BRUSHED TIN	23	24
Brand#11	STANDARD BURNISHED BRASS	36	24
Brand#15	MEDIUM ANODIZED NICKEL	3	24
Brand#15	SMALL ANODIZED BRASS	45	24
Brand#15	SMALL BURNISHED NICKEL	19	24
Brand#21	MEDIUM ANODIZED COPPER	3	24
Brand#22	SMALL BRUSHED NICKEL	3	24
Brand#22	SMALL BURNISHED BRASS	19	24

....Additional rows removed

Brand#25	MEDIUM BURNISHED COPPER	36	24
Brand#31	PROMO POLISHED COPPER	36	24
Brand#33	LARGE POLISHED TIN	23	24
Brand#33	PROMO POLISHED STEEL	14	24

Brand#35	PROMO BRUSHED NICKEL	14	24
Brand#41	ECONOMY BRUSHED STEEL	9	24
Brand#41	ECONOMY POLISHED TIN	19	24
Brand#41	LARGE PLATED COPPER	36	24
Brand#42	ECONOMY PLATED BRASS	3	24
Brand#33	SMALL ANODIZED BRASS	9	3
Brand#35	MEDIUM ANODIZED TIN	19	3
Brand#51	SMALL PLATED BRASS	23	3
Brand#52	MEDIUM BRUSHED BRASS	45	3
Brand#53	MEDIUM BRUSHED TIN	45	3
Brand#54	ECONOMY POLISHED BRASS	9	3
Brand#55	PROMO PLATED BRASS	19	3
Brand#55	STANDARD PLATED TIN	49	3

Number of rows retrieved is: 18314

Qualification Query 17

-- Query 17 - Var_0 Rev_01 - Small-Quantity-Order Revenue Query

Tag: Q17 Stream: -1 Sequence number: 6

```
select
sum(l_extendedprice) / 7.0 as avg_yearly
from
tpcd.lineitem,
tpcd.part
where
p_partkey = l_partkey
and p_brand = 'Brand#23'
and p_container = 'MED BOX'
and l_quantity < (
select
0.2 * avg(l_quantity)
from
tpcd.lineitem
where
l_partkey = p_partkey
)
```

AVG_YEARLY
348406.054

Number of rows retrieved is: 1

Qualification Query 18

-- Query 18 - Var_0 Rev_01 - Large Volume Customer Query

Tag: Q18 Stream: -1 Sequence number: 7

```
select
c_name,
c_custkey,
o_orderkey,
o_orderdate,
o_totalprice,
sum(l_quantity)
from
tpcd.customer,
tpcd.orders,
```

```

tpcd.lineitem
where
o_orderkey in (
select
l_orderkey
from
tpcd.lineitem
group by
l_orderkey having
sum(l_quantity) > 300
)
and c_custkey = o_custkey
and o_orderkey = l_orderkey
group by
c_name,
c_custkey,
o_orderkey,
o_orderdate,
o_totalprice
order by
o_totalprice desc,
o_orderdate
fetch first 100 rows only

```

C_NAME	C_CUSTKEY	O_ORDERKEY
O_ORDERDATE	O_TOTALPRICE	6

Customer#000128120	128120	4722021 1994-04-07
544089.090	323.000	
Customer#000144617	144617	3043270 1997-02-12
530604.440	317.000	
Customer#000013940	13940	2232932 1997-04-13
522720.610	304.000	
Customer#000066790	66790	2199712 1996-09-30
515531.820	327.000	
Customer#000046435	46435	4745607 1997-07-03
508047.990	309.000	
Customer#000015272	15272	3883783 1993-07-28
500241.330	302.000	

.... Additional rows deleted

Customer#000012599	12599	4259524 1998-02-12
415200.610	304.000	
Customer#000105410	105410	4478371 1996-03-05
412754.510	302.000	
Customer#000149842	149842	5156581 1994-05-30
411329.350	302.000	
Customer#000010129	10129	5849444 1994-03-21
409129.850	309.000	
Customer#000069904	69904	1742403 1996-10-19
408513.000	305.000	
Customer#000017746	17746	6882 1997-04-09
408446.930	303.000	
Customer#000013072	13072	1481925 1998-03-15
399195.470	301.000	
Customer#000082441	82441	857959 1994-02-07
382579.740	305.000	
Customer#000088703	88703	2995076 1994-01-30
363812.120	302.000	

Number of rows retrieved is: 57

Qualification Query 19

-- Query 19 - Var_0 Rev_01 - Discounted Revenue Query

Tag: Q19 Stream: -1 Sequence number: 19

```

select
sum(l_extendedprice* (1 - l_discount)) as revenue
from
tpcd.lineitem,
tpcd.part
where
(
p_partkey = l_partkey
and p_brand = 'Brand#12'
and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
and l_quantity >= 1 and l_quantity <= 1 + 10
and p_size between 1 and 5
and l_shipmode in ('AIR', 'AIR REG')
and l_shipinstruct = 'DELIVER IN PERSON'
)
or
(
p_partkey = l_partkey
and p_brand = 'Brand#23'
and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
and l_quantity >= 10 and l_quantity <= 10 + 10
and p_size between 1 and 10
and l_shipmode in ('AIR', 'AIR REG')
and l_shipinstruct = 'DELIVER IN PERSON'
)
)

```

REVENUE

3083843.058

Number of rows retrieved is: 1

Qualification Query 20

-- Query 20 - Var_0 Rev_01 - Potential Part Promotion Query

Tag: Q20 Stream: -1 Sequence number: 4

```

select
s_name,
s_address
from
tpcd.supplier,
tpcd.nation
where
s_suppkey in (
select
ps_suppkey
from
tpcd.partsupp

```

```

where
ps_partkey in (
select
p_partkey
from
tpcd.part
where
p_name like 'forest%'
)
and ps_availqty > (
select
0.5 * sum(l_quantity)
from
tpcd.lineitem
where
l_partkey = ps_partkey
and l_suppkey = ps_suppkey
and l_shipdate >= date ('1994-01-01')
and l_shipdate < date ('1994-01-01') + 1 year
)
)
and s_nationkey = n_nationkey
and n_name = 'CANADA'
order by
s_name

```

S_NAME	S_ADDRESS
Supplier#000000020	iybAE,RmTymrZVYaFZva2SH,j
Supplier#000000091	YV45D7TkfdQanOOZ7q9QxkyGUapU1oOWU6q3
Supplier#000000197	YC2Acon6kjY3zj3Fbxs2k4Vdf7X0cd2F
Supplier#000000226	83qOdU2EYRdPQAQhEtn GRZEd
Supplier#000000285	Br7e1nnt1yxrw6lmgpJ7YdhFDjuBf
Supplier#000000378	FfbhyCxWvcPrO8ltp9
Supplier#000000402	i9Sw4DoyMhzhKXCH9By,AYSgmd
Supplier#000000530	0qwCMwobKY OcmLyfRXlagA8ukENJv,
Supplier#000000688	D fw5ocppmZpYBBIP1718hCihLDZ5KhKX
Supplier#000000710	f19YPvOyb QoYwjKC,oPycpGfieBAcwKJo
Supplier#000000736	l6i2nMwVuovfKnuVgaSGK2rDy65DlAFLegiL7
Supplier#000000761	zLSLeQUj2XrvTTFnv7WAcYZGvvMTx882d4
Supplier#000000884	bmhEShejaS

.... Additional rows deleted

Supplier#000000887	urEaTejH5POADP2ARf
Supplier#000000935	ij98czM 2KzWe7dTOxB8sq0UfCdvX
Supplier#000000975	,AC e,tBpNwKb5xMUzeohxlRn,
hdZJo73gFQF8y	
Supplier#000001263	rQWrfn8ZhB2TAiIDlvo5Io
Supplier#000001399	LmrocnIMSyYOWuANx7
Supplier#000001446	lch9HMNU1R7a0LIybsUodVknk6
Supplier#000001454	TOpimgu2TVXljiL93h,
Supplier#000009862	rJzweWeN58
Supplier#000009868	ROjGgx5gvtkmnUUoeyy7v
Supplier#000009869	
ucLqxzrpBTRMewGSM29t0rNTM30g1Tu3Xgg3mKag	
Supplier#000009899	7XdpAHzr1t,UQFZE
Supplier#000009974	7wJ,J5DKcxSU4Kp1cQLpbcAvB5AsvKT

Number of rows retrieved is: 204

Qualification Query 21

-- Query 21 - Var_0 Rev_01 - Suppliers Who Kept Orders Waiting Query

Tag: Q21 Stream: -1 Sequence number: 9

```

select
s_name,
count(*) as numwait
from
tpcd.supplier,
tpcd.lineitem l1,
tpcd.orders,
tpcd.nation
where
s_suppkey = l1.l_suppkey
and o_orderkey = l1.l_orderkey
and o_orderstatus = 'F'
and l1.l_receiptdate > l1.l_commitdate
and exists (
select
*
from
tpcd.lineitem l2
where
l2.l_orderkey = l1.l_orderkey
and l2.l_suppkey <> l1.l_suppkey
)
and not exists (
select
*
from
tpcd.lineitem l3
where
l3.l_orderkey = l1.l_orderkey
and l3.l_suppkey <> l1.l_suppkey
and l3.l_receiptdate > l3.l_commitdate
)
and s_nationkey = n_nationkey
and n_name = 'SAUDI ARABIA'
group by
s_name
order by
numwait desc,
s_name
fetch first 100 rows only

```

S_NAME	NUMWAIT
Supplier#000002829	20
Supplier#000005808	18
Supplier#000000262	17
Supplier#000000496	17
Supplier#000002160	17
Supplier#000002301	17
Supplier#000002540	17
Supplier#000003063	17
Supplier#0000005178	17
Supplier#000008331	17
Supplier#000002005	16
Supplier#000002095	16

.... Additional rows deleted

Supplier#000009435	13
Supplier#000009436	13
Supplier#000009564	13
Supplier#000009896	13
Supplier#000000379	12
Supplier#000000673	12
Supplier#000000762	12
Supplier#000000811	12
Supplier#000000821	12

Supplier#000001337 12
Supplier#000001916 12
Supplier#000001925 12
Supplier#000002039 12
Supplier#000002357 12
Supplier#000002483 12

Number of rows retrieved is: 100

Qualification Query 22

-- Query 22 - Var_0 Rev_01 - Global Sales Opportunity Query

Tag: Q22 Stream: -1 Sequence number: 12

```
select
cntrycode,
count(*) as numcust,
sum(c_acctbal) as totacctbal
from
(
select
substr(c_phone, 1, 2) as cntrycode,
c_acctbal
from
tpcd.customer
where
substr(c_phone, 1, 2) in
('13', '31', '23', '29', '30', '18', '17')
and c_acctbal > (
select
avg(c_acctbal)
from
tpcd.customer
where
c_acctbal > 0.00
and substr(c_phone, 1, 2) in
('13', '31', '23', '29', '30', '18', '17')
)
and not exists (
select
*
from
tpcd.orders
where
o_custkey = c_custkey
)
) as custsale
group by
cntrycode
order by
cntrycode
```

CNTRYCODE	NUMCUST	TOTACCTBAL
13	888	6737713.990
17	861	6460573.720
18	964	7236687.400
23	892	6701457.950
29	948	7158866.630
30	909	6808436.130
31	922	6806670.180

Number of rows retrieved is: 7

C.2 First 10 Rows of Test Database Tables

SELECT * FROM TPCD.REGION FETCH FIRST 10 ROWS ONLY

R_REGIONKEY	R_NAME	R_COMMENT
0	AFRICA	special Tiresias about the furiously even dolphins are furi
1	AMERICA	even, ironic theodolites according to the bold platelets wa
2	ASIA	silent, bold requests sleep slyly across the quickly sly dependencies. furiously silent instructions alongside
3	EUROPE	special, bold deposits haggle foxes. platelet
4	MIDDLE EAST	furiously unusual packages use carefully above the unusual, exp
5 record(s) selected.		

SELECT * FROM TPCD.NATION FETCH FIRST 10 ROWS ONLY

N_NATIONKEY	N_NAME	N_REGIONKEY	N_COMMENT
0	ALGERIA	0	final accounts wake quickly. special reques
1	ARGENTINA	1	idly final instructions cajole stealthily. regular instructions wake carefully blithely express accounts. fluffi
2	BRAZIL	1	always pending pinto beans sleep sil
3	CANADA	1	foxes among the bold requests
4	EGYPT	4	pending accounts haggle furiously. furiously bold accounts detect. platelets at the packages haggle caref
5	ETHIOPIA	0	fluffily ruthless requests integrate fluffily. pending ideas wake blithely acco
6	FRANCE	3	even requests detect near the pendin
7	GERMANY	3	blithely ironic foxes grow. quickly pending accounts are b

8 INDIA 2 ironic packages should have to are slyly
around the special, ironic accounts. iron

9 INDONESIA 2 unusual excuses are quickly
requests. slyly ironic accounts haggle carefully above the pendin

10 record(s) selected.

SELECT * FROM TPCD.PART FETCH FIRST 10 ROWS ONLY

P_PARTKEY	P_NAME	P_SIZE	P_MFGR
P_BRAND	P_TYPE	P_CONTAINER	
P_RETAILPRICE	P_COMMENT		

5179 yellow peach orchid turquoise dodger	Manufacturer#2
Brand#21 STANDARD POLISHED COPPER	8 WRAP JAR
+1.084170000000000E+003 quickly ex	

5249 indian navajo chiffon black khaki	Manufacturer#5
Brand#53 LARGE ANODIZED STEEL	1 WRAP PKG
+1.154240000000000E+003 regular deposits	

5250 olive grey green bisque puff	Manufacturer#1
Brand#14 SMALL BURNISHED NICKEL	31 JUMBO CASE
+1.155250000000000E+003 quick	

5253 slate navajo bisque honeydew olive	Manufacturer#5
Brand#51 PROMO PLATED STEEL	35 WRAP PKG
+1.158250000000000E+003 carefully	

5289 olive wheat seashell purple medium	Manufacturer#3
Brand#31 SMALL BURNISHED STEEL	31 JUMBO BAG
+1.194280000000000E+003 ironic packages	

5307 tomato aquamarine ghost drab navy	Manufacturer#1
Brand#12 PROMO PLATED TIN	43 WRAP CAN
+1.212300000000000E+003 deposits int	

5317 smoke goldenrod aquamarine beige light	Manufacturer#2
Brand#21 MEDIUM ANODIZED TIN	27 MED CAN
+1.222310000000000E+003 ironic, pen	

5338 papaya metallic ivory azure wheat	Manufacturer#4
Brand#43 STANDARD BURNISHED STEEL	34 SM BOX
+1.243330000000000E+003 deposits thro	

5342 drab smoke chiffon chocolate seashell	Manufacturer#3
Brand#32 ECONOMY BRUSHED STEEL	50 MED PACK
+1.247340000000000E+003 Tiresias	

5352 lemon coral pink sky honeydew	Manufacturer#5
Brand#51 PROMO BRUSHED TIN	9 SM PACK
+1.257350000000000E+003 even deposits	

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

4818 Supplier#000004818	pq
8W7GgiPQEHo4qMvREULChRuKqxSP9GQm2ox	4 14-584-467-9693
+9.793280000000000E+003 pending ideas grow blithely among the regular requests. platelets us	

4852 Supplier#000004852	Uwdz2nnPMHBMxJeM9zCsiqo0mxjq
23 33-384-566-3301	+8.701720000000000E+003 final requests across the blithely silent package

4899 Supplier#000004899	D4b3yDzDgK2WyaUdXWo0r
16 26-897-272-3576	+1.819500000000000E+003 slyly final deposits use blith

4912 Supplier#000004912	2NUOQlm1bhr1bf
11 21-127-612-6452	+4.876370000000000E+003 fluffily final pinto beans haggle carefully according to the blithely unusual accounts. furiously en

4956 Supplier#000004956	ibDuLQ0D8W1hK3m
7sMKFt5fw8XGzp1aU	24 34-954-986-5576
+9.131890000000000E+003 even excuses cajole furiously. carefully bold packages up the express, unusual packag	

4967 Supplier#000004967	lIvbn4iSOB4F VwG0xLQU3QHn4
6 16-766-968-2266	+8.990070000000000E+003 unusual, regular packages wake final platelets.

5003 Supplier#000005003	dv34hDJQK,S9C6HIxdkX AbHvFe
14 24-644-394-1098	+9.244340000000000E+003 express courts breach slyly

5015 Supplier#000005015	HOXohc8D8KJ Yx2WIOl
5EvdnTqwXSo	3 13-358-401-5005
+2.096160000000000E+003 express pinto beans wake above the quickly regular deposits. blithely final theod	

5049 Supplier#000005049	22J7yDqk9pnGoguOvx0x
11 21-500-137-4477	+2.063910000000000E+003 blithely unusual requests run furiously. fluffily final instruct

5068 Supplier#000005068	lywBhkmjFn885,AxSnroD
6 16-491-789-4986	+9.481980000000000E+003 asymptotes affix fluffily about the pending foxes. ideas maintain blithely. carefully ironi

10 record(s) selected.

SELECT * FROM TPCD.PARTSUPP FETCH FIRST 10 ROWS ONLY

PS_PARTKEY PS_SUPPKEY PS_AVAILQTY PS_SUPPLYCOST
PS_COMMENT

464 2250465 9366 +1.918000000000000E+001 theodolites
serve. blithely pending accounts above the furiously unusual requests are
according to the frays. thin sentiments cajole blithely unusual deposits:
fluffily regular packages after the

474 475 3200 +7.281500000000000E+002 furiously regular
theodolites cajole slyly. slyly ironic requests cajole always furiously even
warthogs? final epitaphs despite the regular, u

474 750475 6439 +4.209800000000000E+002 furiously ironic
asymptotes cajole regularly regular attainments. carefully pending accounts
across t

474 1500475 6252 +6.596000000000000E+002 requests ought
to doze: furiously regular dinos do boost furiously. fluffily final deposits
about the regular braids haggle af

474 2250475 6728 +6.776000000000000E+001 slyly regular
accounts should have to breach blithely pending packages. express deposits
nag always across the blithely express mult

478 479 4424 +5.065500000000000E+002 slyly regular
instructions are unusual requests. slyly final instructions against the slyly
even pinto beans cajole blithely along the furiously unusual accounts.
furiously regular platelets cajole

478 750479 2049 +2.025000000000000E+002 carefully unusual
pearls sleep braids: regular, special accounts across the ironic instructions
wake furiously among the

478 1500479 9781 +1.075900000000000E+002 final excuses use
carefully. theodolites sleep slyl

478 2250479 4653 +3.834100000000000E+002 fluffily express
requests along the thin requests use quickly special asymptotes. slyly even
requests boost blithely ironic notornis. asymptotes x-ray over the furiously
regul

538 539 4074 +2.302700000000000E+002 carefully bold
packages haggle furiously. blithely special asymptotes need

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

43 Customer#000000043
ouSbjHk8lh5fKX3zGso3ZSlj9Aa3PoaFd 19 29-316-665-2897

+9.904280000000000E+003 MACHINERY idly regular sentiments affix.
slyly pending foxes around the deposits haggle according to th

47 Customer#000000047 b0UgocSqEW5 gdVbhNT
2 12-427-271-9466 +2.745800000000000E+002 BUILDING blithely final
deposits about the furiously special packages play even idea

89 Customer#000000089 dtR, y9JQWUO6FoJEryp8whOU
14 24-394-451-5404 +1.530760000000000E+003 FURNITURE bold
accounts cajole furiously after the permanently even excuses. accounts
cajole. fluffily even requ

98 Customer#000000098 7yiheXNSpuEAwbswDW
12 22-885-845-6889 -5.513700000000000E+002 BUILDING blithely
final foxes along the unusual realms detect accounts. idle theodolites use
pinto beans. ev

119 Customer#000000119 M1ETOIecuvH8DtM0Y0nryXfW
7 17-697-919-8406 +3.930350000000000E+003 FURNITURE regular,
ironic theodolites alongside of the instructions haggle carefully even
requests. packages

122 Customer#000000122
yp5slqoNd26IAENZW3a67wSfXA6hTF 3 13-702-694-4520
+7.865460000000000E+003 HOUSEHOLD furiously regular instructions
cajole furiously ideas. quickly bold foxes wake careful

125 Customer#000000125 ,wSZXdVR
xxIIIfm9s8ITyLl3kgjT6UC07GY0Y 19 29-261-996-3120 -
2.341200000000000E+002 FURNITURE special instructions use above the
carefully quick packages. closely fi

126 Customer#000000126 ha4EHmbx3kg
DYCsP6DFeUOmavtQlHhcfagr 22 32-755-914-7592
+1.001390000000000E+003 HOUSEHOLD carefully special requests after
the slyly regular packages sleep furiously slyly final epitaphs.

144 Customer#000000144 VxYZ3ebhgbltnetaGjNC8qCccjYU05
fePLOno8y 1 11-717-379-4478 +6.417310000000000E+003
MACHINERY fluffily regular requests at the blithely even theodolites
haggle around the requests. special frets x-

169 Customer#000000169 NjhmHa7xrcjE
18 28-362-499-3728 +4.483830000000000E+003 FURNITURE furiously
pending dugouts use furiously unusual, regular deposits. carefully regul

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

18692741 18337831 F +1.065448000000000E+005
01/01/1992 2-HIGH Clerk#000096561 0 furiously special
tithes wake against the theodolites. carefull

18729029 32989486 F +1.671037300000000E+005
01/01/1992 5-LOW Clerk#000023492 0 ironic requests sleep
slyly at

18729697 25047500 F +1.298983900000000E+005
01/01/1992 1-URGENT Clerk#000065079 0 quickly regular
asymptotes are

18751555 38974309 F +1.463093300000000E+005
01/01/1992 2-HIGH Clerk#000157727 0 furiously pending
deposits are furiously unusual i

19025506 3953378 F +8.501953000000000E+004
01/01/1992 2-HIGH Clerk#000159871 0 pinto beans was.
slyly pending deposits haggle, unusual instruct

19056358 16826557 F +1.068840100000000E+005
01/01/1992 1-URGENT Clerk#000260621 0 furiously express
gifts detect about the accounts. bold pac

19302627 35805052 F +1.167429700000000E+005
01/01/1992 2-HIGH Clerk#000164311 0 regular pinto beans
haggle furiously. regular, bold pinto beans integrate bli

19521698 14548637 F +5.763300000000000E+004
01/01/1992 2-HIGH Clerk#000108921 0 quickly even
theodolites against the closely regular courts will affix afte

19547653 35437115 F +2.355920000000000E+005
01/01/1992 4-NOT SPECIFIED Clerk#000216089 0 ironic requests
caj

19554786 39859541 F +2.008538900000000E+005
01/01/1992 5-LOW Clerk#000166573 0 furiously ironic frets
about the blithely regu

10 record(s) selected.

SELECT * FROM TPCD.LINEITEM FETCH FIRST 10 ROWS ONLY

L_ORDERKEY L_PARTKEY L_SUPPKEY L_LINENUMBER
L_QUANTITY L_EXTENDEDPRICE L_DISCOUNT
L_TAX L_RETURNFLAG L_LINESTATUS L_SHIPDATE
L_COMMITDATE L_RECEIPTDATE L_SHIPINSTRUCT
L_SHIPMODE L_COMMENT

1054181 4865078 1865079 1
+4.500000000000000E+001 +4.692735000000000E+004
+3.000000000000000E-002 +8.000000000000000E-002 R F
01/02/1992 02/05/1992 01/15/1992 NONE MAIL even
instructions kindle furio

5018977 24611600 611601 1
+2.000000000000000E+001 +3.020740000000000E+004
+0.000000000000000E+000 +0.000000000000000E+000 A F
01/02/1992 03/19/1992 01/15/1992 NONE SHIP quickly

ironic excu

19547653 37197566 1947579 1
+4.800000000000000E+001 +7.976208000000000E+004
+0.000000000000000E+000 +7.000000000000000E-002 R F
01/02/1992 02/08/1992 01/26/1992 NONE MAIL
regular accounts haggle. h

19918976 6334057 2584064 6
+3.100000000000000E+001 +3.381294000000000E+004
+5.000000000000000E-002 +6.000000000000000E-002 R F
01/02/1992 02/23/1992 01/03/1992 DELIVER IN PERSON REG
AIR carefully fluffy requests

27011394 1996025 2746026 4
+4.200000000000000E+001 +4.707906000000000E+004
+4.000000000000000E-002 +5.000000000000000E-002 A F
01/02/1992 03/21/1992 01/20/1992 NONE REG AIR final,
ironic warhorses

35459264 33148881 2398915 2
+1.300000000000000E+001 +2.506699000000000E+004
+1.000000000000000E-001 +1.000000000000000E-002 A F
01/02/1992 02/02/1992 01/07/1992 COLLECT COD AIR
even packages abo

35584422 24142067 2392092 2
+2.000000000000000E+000 +2.215720000000000E+003
+0.000000000000000E+000 +1.000000000000000E-002 R F
01/02/1992 03/03/1992 01/28/1992 NONE SHIP slyly
regular theodolit

37766533 7675859 1675860 5
+1.600000000000000E+001 +2.935152000000000E+004
+4.000000000000000E-002 +2.000000000000000E-002 A F
01/02/1992 02/23/1992 01/07/1992 COLLECT COD FOB
courts are blithely across the iron

45828802 3935061 935062 7
+4.500000000000000E+001 +4.931415000000000E+004
+3.000000000000000E-002 +4.000000000000000E-002 A F
01/02/1992 03/29/1992 01/14/1992 TAKE BACK RETURN FOB
carefully special fo

47726080 17850144 2850145 1
+2.400000000000000E+001 +2.623800000000000E+004
+2.000000000000000E-002 +2.000000000000000E-002 A F
01/02/1992 02/23/1992 01/16/1992 DELIVER IN PERSON RAIL
ironic excuses ab

10 record(s) selected.

C.3 Query Substitution Parameters

"Power stream Seed = 403202005"
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 403202005 as a seed to the RNG

Q1 DELTA 115
Q2 SIZE 45
TYPE BRASS
REGION AMERICA
Q3 SEGMENT HOUSEHOLD
DATE 1995-03-11
Q4 DATE 1994-08-01
Q5 REGION MIDDLE EAST
DATE 1995-01-01
Q6 DATE 1995-01-01
DISCOUNT 0.09

QUANTITY 25
 Q7 NATION1 UNITED STATES
 NATION2 SAUDI ARABIA
 Q8 NATION SAUDI ARABIA
 REGION MIDDLE EAST
 TYPE ECONOMY BURNISHED TIN
 Q9 COLOR pink
 Q10 DATE 1994-03-01
 Q11 NATION MOZAMBIQUE
 FRACTION 0.0000003333
 Q12 SHIPMODE1 MAIL
 SHIPMODE2 AIR
 DATE 1995-01-01
 Q13 WORD1 pending
 WORD2 requests
 Q14 DATE 1995-04-01
 Q15 DATE 1993-02-01
 Q16 BRAND Brand#21
 TYPE STANDARD POLISHED
 SIZE1 8
 SIZE2 49
 SIZE3 37
 SIZE4 11
 SIZE5 17
 SIZE6 41
 SIZE7 2
 SIZE8 25
 Q17 BRAND Brand#11
 CONTAINER SM DRUM
 Q18 QUANTITY 312
 Q19 BRAND1 Brand#15
 BRAND2 Brand#42
 BRAND3 Brand#15
 QUANTITY1 7
 QUANTITY2 13
 QUANTITY3 27
 Q20 COLOUR violet
 DATE 1996-01-01
 NATION GERMANY
 Q21 NATION MOROCCO
 Q22 I1 10
 I2 11
 I3 23
 I4 26
 I5 18
 I6 27
 I7 20

"Throughput Stream = 1 Seed = 403202006"
 -- TPC TPC-H Parameter Substitution (Version 1.3.0)
 -- using 403202006 as a seed to the RNG

Q1 DELTA 62
 Q2 SIZE 33
 TYPE NICKEL
 REGION MIDDLE EAST
 Q3 SEGMENT BUILDING
 DATE 1995-03-27
 Q4 DATE 1997-03-01
 Q5 REGION AFRICA
 DATE 1996-01-01
 Q6 DATE 1996-01-01
 DISCOUNT 0.07
 QUANTITY 25
 Q7 NATION1 MOZAMBIQUE
 NATION2 JAPAN
 Q8 NATION JAPAN
 REGION ASIA
 TYPE LARGE BRUSHED TIN
 Q9 COLOR orange
 Q10 DATE 1994-12-01

Q11 NATION EGYPT
 FRACTION 0.0000003333
 Q12 SHIPMODE1 TRUCK
 SHIPMODE2 RAIL
 DATE 1995-01-01
 Q13 WORD1 pending
 WORD2 requests
 Q14 DATE 1995-07-01
 Q15 DATE 1995-08-01
 Q16 BRAND Brand#51
 TYPE LARGE ANODIZED
 SIZE1 13
 SIZE2 32
 SIZE3 15
 SIZE4 45
 SIZE5 7
 SIZE6 34
 SIZE7 8
 SIZE8 3
 Q17 BRAND Brand#13
 CONTAINER LG BAG
 Q18 QUANTITY 314
 Q19 BRAND1 Brand#22
 BRAND2 Brand#25
 BRAND3 Brand#14
 QUANTITY1 2
 QUANTITY2 14
 QUANTITY3 23
 Q20 COLOUR grey
 DATE 1994-01-01
 NATION RUSSIA
 Q21 NATION GERMANY
 Q22 I1 32
 I2 13
 I3 26
 I4 21
 I5 16
 I6 28
 I7 30

"Throughput Stream = 2 Seed = 403202007"
 -- TPC TPC-H Parameter Substitution (Version 1.3.0)
 -- using 403202007 as a seed to the RNG

Q1 DELTA 70
 Q2 SIZE 21
 TYPE TIN
 REGION AMERICA
 Q3 SEGMENT HOUSEHOLD
 DATE 1995-03-13
 Q4 DATE 1994-12-01
 Q5 REGION AMERICA
 DATE 1996-01-01
 Q6 DATE 1996-01-01
 DISCOUNT 0.04
 QUANTITY 24
 Q7 NATION1 INDIA
 NATION2 EGYPT
 Q8 NATION EGYPT
 REGION MIDDLE EAST
 TYPE LARGE PLATED TIN
 Q9 COLOR mint
 Q10 DATE 1993-09-01
 Q11 NATION PERU
 FRACTION 0.0000003333
 Q12 SHIPMODE1 RAIL
 SHIPMODE2 TRUCK
 DATE 1993-01-01
 Q13 WORD1 pending
 WORD2 accounts
 Q14 DATE 1995-10-01

Q15 DATE 1993-05-01
 Q16 BRAND Brand#41
 TYPE PROMO BURNISHED
 SIZE1 9
 SIZE2 28
 SIZE3 19
 SIZE4 5
 SIZE5 24
 SIZE6 26
 SIZE7 34
 SIZE8 27
 Q17 BRAND Brand#15
 CONTAINER LG PACK
 Q18 QUANTITY 315
 Q19 BRAND1 Brand#24
 BRAND2 Brand#53
 BRAND3 Brand#13
 QUANTITY1 7
 QUANTITY2 15
 QUANTITY3 20
 Q20 COLOUR saddle
 DATE 1997-01-01
 NATION IRAQ
 Q21 NATION UNITED STATES
 Q22 I1 30
 I2 25
 I3 22
 I4 26
 I5 12
 I6 21
 I7 27

"Throughput Stream = 3 Seed = 403202008"
 -- TPC TPC-H Parameter Substitution (Version 1.3.0)
 -- using 403202008 as a seed to the RNG
 Q1 DELTA 78
 Q2 SIZE 9
 TYPE COPPER
 REGION MIDDLE EAST
 Q3 SEGMENT AUTOMOBILE
 DATE 1995-03-29
 Q4 DATE 1997-07-01
 Q5 REGION ASIA
 DATE 1996-01-01
 Q6 DATE 1996-01-01
 DISCOUNT 0.02
 QUANTITY 25
 Q7 NATION1 ALGERIA
 NATION2 VIETNAM
 Q8 NATION VIETNAM
 REGION ASIA
 TYPE LARGE ANODIZED NICKEL
 Q9 COLOR linen
 Q10 DATE 1994-07-01
 Q11 NATION ETHIOPIA
 FRACTION 0.0000003333
 Q12 SHIPMODE1 REG AIR
 SHIPMODE2 RAIL
 DATE 1996-01-01
 Q13 WORD1 pending
 WORD2 accounts
 Q14 DATE 1996-02-01
 Q15 DATE 1995-12-01
 Q16 BRAND Brand#21
 TYPE SMALL POLISHED
 SIZE1 7
 SIZE2 24
 SIZE3 14
 SIZE4 36
 SIZE5 13

SIZE6 28
 SIZE7 10
 SIZE8 31
 Q17 BRAND Brand#12
 CONTAINER LG DRUM
 Q18 QUANTITY 313
 Q19 BRAND1 Brand#22
 BRAND2 Brand#41
 BRAND3 Brand#52
 QUANTITY1 2
 QUANTITY2 16
 QUANTITY3 27
 Q20 COLOUR cream
 DATE 1996-01-01
 NATION ARGENTINA
 Q21 NATION MOZAMBIQUE
 Q22 I1 16
 I2 23
 I3 19
 I4 14
 I5 25
 I6 21
 I7 11

"Throughput Stream = 4 Seed = 403202009"
 -- TPC TPC-H Parameter Substitution (Version 1.3.0)
 -- using 403202009 as a seed to the RNG
 Q1 DELTA 86
 Q2 SIZE 47
 TYPE BRASS
 REGION ASIA
 Q3 SEGMENT HOUSEHOLD
 DATE 1995-03-15
 Q4 DATE 1995-04-01
 Q5 REGION MIDDLE EAST
 DATE 1996-01-01
 Q6 DATE 1996-01-01
 DISCOUNT 0.07
 QUANTITY 25
 Q7 NATION1 PERU
 NATION2 JORDAN
 Q8 NATION JORDAN
 REGION MIDDLE EAST
 TYPE MEDIUM POLISHED NICKEL
 Q9 COLOR lace
 Q10 DATE 1993-04-01
 Q11 NATION CHINA
 FRACTION 0.0000003333
 Q12 SHIPMODE1 SHIP
 SHIPMODE2 RAIL
 DATE 1996-01-01
 Q13 WORD1 pending
 WORD2 accounts
 Q14 DATE 1996-05-01
 Q15 DATE 1993-08-01
 Q16 BRAND Brand#51
 TYPE LARGE BRUSHED
 SIZE1 21
 SIZE2 27
 SIZE3 18
 SIZE4 33
 SIZE5 47
 SIZE6 11
 SIZE7 39
 SIZE8 25
 Q17 BRAND Brand#14
 CONTAINER MED BAG
 Q18 QUANTITY 314
 Q19 BRAND1 Brand#34
 BRAND2 Brand#24

```

BRAND3      Brand#52
QUANTITY1   8
QUANTITY2   17
QUANTITY3   23
Q20 COLOUR   orange
DATE        1994-01-01
NATION      MOZAMBIQUE
Q21 NATION   INDIA
Q22 I1       11
I2          33
I3          15
I4          22
I5          27
I6          18
I7          32

"Throughput Stream = 5  Seed = 403202010"
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 403202010 as a seed to the RNG
Q1 DELTA     94
Q2 SIZE      34
TYPE         NICKEL
REGION       MIDDLE EAST
Q3 SEGMENT   AUTOMOBILE
DATE         1995-03-31
Q4 DATE      1997-11-01
Q5 REGION    AFRICA
DATE         1997-01-01
Q6 DATE      1997-01-01
DISCOUNT    0.04
QUANTITY     24
Q7 NATION1   INDIA
NATION2      ETHIOPIA
Q8 NATION    ETHIOPIA
REGION       AFRICA
TYPE         MEDIUM BURNISHED NICKEL
Q9 COLOR     grey
Q10 DATE     1994-01-01
Q11 NATION   FRANCE
FRACTION     0.0000003333
Q12 SHIPMODE1 FOB
SHIPMODE2    TRUCK
DATE         1996-01-01
Q13 WORD1    pending
WORD2        accounts
Q14 DATE     1996-08-01
Q15 DATE     1996-03-01
Q16 BRAND    Brand#41
TYPE         STANDARD BURNISHED
SIZE1        27
SIZE2        36
SIZE3        6
SIZE4        11
SIZE5        40
SIZE6        1
SIZE7        15
SIZE8        24
Q17 BRAND    Brand#11
CONTAINER    MED PACK
Q18 QUANTITY 312
Q19 BRAND1   Brand#31
BRAND2       Brand#12
BRAND3       Brand#51
QUANTITY1    3
QUANTITY2    18
QUANTITY3    30
Q20 COLOUR   beige
DATE         1993-01-01
NATION       FRANCE
Q21 NATION   ALGERIA

```

```

Q22 I1       16
I2          17
I3          33
I4          26
I5          20
I6          23
I7          11

"Throughput Stream = 6  Seed = 403202011"
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 403202011 as a seed to the RNG
Q1 DELTA     102
Q2 SIZE      22
TYPE         TIN
REGION       ASIA
Q3 SEGMENT   FURNITURE
DATE         1995-03-17
Q4 DATE      1995-08-01
Q5 REGION    AMERICA
DATE         1997-01-01
Q6 DATE      1997-01-01
DISCOUNT    0.02
QUANTITY     25
Q7 NATION1   ALGERIA
NATION2      RUSSIA
Q8 NATION    RUSSIA
REGION       EUROPE
TYPE         SMALL BRUSHED NICKEL
Q9 COLOR     forest
Q10 DATE     1994-10-01
Q11 NATION   ROMANIA
FRACTION     0.0000003333
Q12 SHIPMODE1 MAIL
SHIPMODE2    TRUCK
DATE         1996-01-01
Q13 WORD1    pending
WORD2        deposits
Q14 DATE     1996-11-01
Q15 DATE     1993-12-01
Q16 BRAND    Brand#21
TYPE         MEDIUM PLATED
SIZE1        9
SIZE2        24
SIZE3        21
SIZE4        15
SIZE5        25
SIZE6        45
SIZE7        36
SIZE8        6
Q17 BRAND    Brand#13
CONTAINER    MED DRUM
Q18 QUANTITY 313
Q19 BRAND1   Brand#33
BRAND2       Brand#45
BRAND3       Brand#45
QUANTITY1    8
QUANTITY2    19
QUANTITY3    26
Q20 COLOUR   lawn
DATE         1996-01-01
NATION       SAUDI ARABIA
Q21 NATION   CHINA
Q22 I1       30
I2          15
I3          32
I4          18
I5          33
I6          25
I7          16

```

|

Appendix D: Implementation Specific Layer and Driver Source Code

D.1 tpcdbatch.h

```

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

/** Necessary header files */

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

#include <time.h>
#include <ctype.h>
#if (defined(SQLAIX) || defined(SQLPTX) || defined(LINUX) ||
defined(SQLHP))
#include <unistd.h>        /* SUN */
#include <sys/stat.h>      /* SUN */
#endif
#if ((defined(SQLAIX) || defined(SQLPTX)) && !defined(LINUX))
#include <sys/vnode.h>     /* SUN */
#endif
#ifndef SQLWINT
#include <sys/time.h>      /*@d33143aha*/
#include <sys/ipc.h>
#include <sys/sem.h>
#if (!defined(SQLPTX) && !defined(LINUX)&& !defined(SQLHP))
#include <sys/mode.h>
#endif
#include <sys/timeb.h>
#include <sys/types.h>
#else
#include <windows.h>
#include <sys/timeb.h>
#endif
#include <errno.h>

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

#include "sqlutil.h"
#include "sqlcodes.h"

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

/*****
*****
*
*   Define synonyms here
*****
*****/
#define TPCDBATCH_VERSION "5.6"

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

#define TPCDBATCH_MAX_COLS   100        /* @d30369
tjj */

#define TPCDBATCH_CHAR char

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

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

#define PARALLEL_UPDATES 1

#ifdef PARALLEL_UPDATES
#define UF1OUTSTREAMPATTERN "%s%cuf1.%02d.%d.out"
#ifdef TPCD_NONPARTITIONED
#define UF2OUTSTREAMPATTERN "%s%cuf2.%02d.%d.out"
#else
/* kelly add same as NONPART. */
#define UF2OUTSTREAMPATTERN "%s%cuf2.%02d.%d.out"
/* kelly ... take this out ... should be same name as for non-partitioned
#define UF2OUTSTREAMPATTERN "%s%cuf2.%02d.%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 21 /* WIN NT timestamp fix bbe */
#else
#error Unknown operating system
#endif
/* jen TIME_ACC start */

#define BLANKS  ""
#define READMODE  "r"
#define WRITEMODE  "w"
#define APPENDMODE  "a"
#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 */

#define DEBUG 0 /* to be set to 1 for diagnostic purposes if needed */
/* #define UF1DEBUG 1 */
/* #define UF2DEBUG 1 */

```

D.2 tpcdbatch.sqc

```

/*****
*****
*
*   TPCDBATCH.SQC
*
*   Revision History:
*
*   21 Dec 95 jen   Corrected calculation of geometric mean to include in the
*                   count of statements the update functions.
*   03 Jan 96 jen   Corrected calculation of arithmetic mean to not include the
*                   timings for the update functions. (only want query timings
*                   as part of arithmetic mean)
*   15 Jan 96 jen   Added extra timestamps to the update functions.
*   22 Jan 96 jen   Get rid of checking of short_time....we always use the long
*                   timings.
*                   Fixed timings to print query/uf times rounded up to 0.1 seconds
*                   and uses these rounded time values in subsequent calculations
*                   Fixed bug where last seed in mseedme file wasn't getting read
*                   correctly - EOF processing done too soon.
*
*   22 Feb 96 kbs   port to NT
*   26 Mar 96 kbs   Fix to avoid countig UFs as queries for min max
*   27 Jun 97 wlc   Temporarily fixed deadlock problems when doing UF1,
UF2
*   30 Jul 97 wlc   Add in support for load_update and
TPCD_SPLIT_DELETEs
*   13 Aug 97 wlc   fixed UF1 log file formatting problem,
*                   using TPCD_TMP_DIR for temp files instead of /tmp,
*                   make summary table fit in 80-column,
*                   fixed UF2 # of deleted rows reporting problem
*   18 Aug 97 wlc   added command line support for inlistmax
*   20 Aug 97 wlc   added support for runthroughput without UF
*   27 Aug 97 aph   Replaced hardcoded 'tpcdaudit' with
getenv("TPCD_AUDIT_DIR")
*   05 Sep 97 wlc   fixing free() problem in NT
*   26 Sep 97 kmw   change FLOAT processing in echo_sqlda and
print_headings
*   10 oct 97 jen   add lock table in share mode for staging tables
*   21 oct 97 jen   added explicit rollback on failure of uf1
*   27 oct 97 jen   don't update TPCD.xxxx.update.pair.num if not running UFs
in
*                   throughput run
*   01 nov 97 jen   temp code to do a prep then execute stmt in UFs so we can
*                   get timings
*   03 nov 97 jen   realigned UF code for readability
*                   pushed UF2 commit into loop for inlistmax
*                   fixed UF2 code so rollback performed
*   04 nov 97 jen   Added code to handle vldb
*   06 nov 97 jen   Commented out temp code for prep then execute stmts using
TPCD_PREPARETIME def
*                   Updated version number to 2.2
*                   send all output during update functiosn to output files, not
*                   stderr
*   10 nov 97 jen   jenCI Updated version number to 2.3
*                   Added handling of TPCD_CONCURRENT_INSERTS. Change
control of
*                   chunk processing to use the concurrent_inserts value as the
*                   control. Now the inserts will be run in
TPCD_CONCURRENT_INSERTS
*                   sets, each having concurrent_inserts/
*   13 nov 97 jen   DEADLOCK. Fixed bug that Alex found where
deadlock count
*                   (maxwait) was incremented on every execution of the stmt as
*                   opposed to just when deadlock really happened.
*   14 nov 97 jen   jenSEM - fix up error reporting on semaphore failure
sem_op now returns failure to caller so caller can report where
failure has happened.
*                   Forced dbname to be upper case, an all other parts of update

```

```

*      pair number to be lowercase
* 15 nov 97 jen SEED Reworked code to grab the seed from the seed file.
Now
*      reusing seeds between runs, so power run will always use first
*      seed, throughput will use the 2nd - #stream+1 seeds
*
* 13 jan 98 jen LONG Increase stmt_str to be able to hold inlists with larger
*      order key numbers
* 04 mar 98 jen IMPORT added support for TPCD_UPDATE_IMPORT to
chose whether
*      using import or load api's for loading data into the staging
*      tables
* 04 mar 98 jen TIMER changed from using gettimer to gettimeofday for
unix
* 01 apr 98 jen Fixed IMPORT code to do the proper checking on strcmp (ie
!strcmp)
* 01 apr 98 jen removed code to handle vldb - not needed
*      Upgraded version to 2.4      for ( chunk
* 01 apr 98 jen Fixed up import code on NT so the variable is recognized in
the
*      children
* 25 may 98 sks Reworked some of the environment variable code so
consolidate as
*      much as possible. Not all complete because of differences in
*      the way nt and AIX calls (and starts stuff in background) for UFs
* 29 may 98 jen REUSE_STAGE Changed UF1 so we reuse the same
staging tables
*      instead of having a new set for each update pair
* 06 jul 98 jen Removed locking of staging tables since they are created
with
*      locksize table now
* 06 jul 98 jen 912RETRY - added code to retry query execution on 912 as
well
*      as 911
* 07 jul 98 jen Fixed summary_table() so 1000x adjustment not based on UF
(setting
*      of max and min pointers
*      Added generic SleepSome function to handle NT vs AIX sleep
differences
* 01 apr 98 djd Added change to permit the use of table functions for UF1.
*      to enable this set TPCD_UPDATE_IMPORT to tf in
TPCD.SETUP file.
*      MERGED this into base copy on Jul 07
* 10 jul 98 jen haider's fix for 'outstream' var for error processing in
runUF1_fn and runUF2_fn
*      Updated version to 2.5
* 25 sep 98 jen Added stream number printing into mpqry* files and
increases
*      accuracy of timestamp in mpqry (and mts*qry*) files
* 06 oct 98 jen TIME_ACC Added accuracy of timestamp in mpqry (and
mts*qry*)
*      files. Cleaned up misuse of Sleep and flushed buffers on
*      deadlocks
* 19 oct 98 kbs fix UF2_fn to correctly count rows deleted in case of
deadlock
* 20 oct 98 kbs rewrite UF2 and UF2_fn for static SQL with staging table
* 23 oct 98 jen Cleaned up retrying of order/lineitem on lineitem deadlock in
UF1
* 24 oct 98 jen Used load_uf1 and load_uf2 instead of general load_updates
* 26 oct 98 kbs inject the UF1 with a single staging table
* 02 nov 98 jen Fixed processing of multiple chunks in uf2 so don't
duplicate
* 21 nov 98 kmw Fixed BIGINT
* 05 dec 98 aph Moved runUF1_fn() and runUF2_fn() into a separate file
tpcdUF.sqc
*      so that it can be bound separately with a different isolation level.
* 21 dec 98 aph Integrated Jennifer's QppD calculation (rounding &
adjustment) fixes.
* 22 dec 98 aph For UFs during Throughput run, defer CONNECT until
children launched.

* 28 dec 98 aph Removed error_check() call after CONNECT RESET
* 29 dec 98 aph For UFs do not COMMIT in tpcdbatch.sqc. COMMITs
happen in tpcdUF.sqc.
* 18 jan 99 kal replaced header with #include "tpcdbatch.h"
* 27 may 99 bbeaton from (03 mar 99 jen) Fixed SUN fix that wasn't
compatible with
*      NT (using %D %T instead of %x %X for strftime)
* 16 jun 99 jen Added missing LPCTSTR cast of semaphore file name for
NT
* 17 jun 99 jen SEMA Changes semaphore file for update functions to look
for tpcd.setup
*      not for the orders.*** update data file
* 21 jul 99 bbeaton Added semaphore control that allows runpower to be run
as two
*      separate streams (update and query). This involves the use of
two semaphores to be used as it executes in three different
*      sections. The first is the update inserts. The next is the query
stream which is started with the update stream, but waits until
*      the inserts are complete. The third section is the update deletes
which execute after the queries are complete.
* 21 jul 99 bbeaton Added functions to handle semaphore creation, control,
etc.
* 21 jul 99 bbeaton Modified output to mp*inter files. It now only outputs
intermediate data that will be calculated by calcmetric.pl. This
*      is a result of the runpower being split into two streams and thus
tpcdbatch not having access to all data.
* 21 jul 99 bbeaton The start time for runpower UF2 now does not start until
after
*      the query stream is complete so that its wait time is not included
NOTE: The wait time that the first UF1 in runthroughput still
*      includes the wait period that occurs waiting on queries.
*****
*****/
/* included in tpcdbatch.sqc and tpcdUF.sqc */

#include "tpcdbatch.h"

/*****
*****/
/* global structure containing elements passed between different functions */
/*****
*****/
struct global_struct
{
    struct stmt_info    *s_info_ptr;      /* ptr to stmt_info list */
    struct stmt_info    *s_info_stop_ptr; /* ptr to last struct in list */
    struct comm_line_opt *c_l_opt;        /* ptr to comm_line_opt struct */
    struct ctrl_flags    *c_flags;        /* ptr to ctrl_flags struct */
    Timer_struct         stream_start_time; /* start time for stream
TIME_ACC */
    Timer_struct         stream_end_time;   /* end time for stream
TIME_ACC */
    char                 file_time_stamp[50]; /* time stamp for output files */
    double               scale_factor;      /* scale factor of database */
    char                 run_dir[150];      /* directory for output files */
    int                  copy_on_load;      /* indication of whether or not */
                                /* to do use a copy directory */
                                /* (equiv to COPY YES) on load */
                                /* default is FALSE */
    long                 lSeed;             /* seed used to generate the */
                                /* queries for this particular */
                                /* run. */
    FILE                 *stream_list;      /* ptr to query list file */
    char                 update_num_file[150]; /* name of file that keeps track */
                                /* of which update pairs have run */
    char                 sem_file[150];     /* semaphore name */
    char                 sem_file2[150];    /* semaphore name bbe */
    FILE                 *stream_report_file; /* file to report start stop */
                                /* progress of the stream */

```

```

};

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

struct stmt_info
{
    long        max_rows_fetch;
    long        max_rows_out;
    int         query_block;           /* @d30369 tlg */
    unsigned int stmt_num;             /* @d24993 tlg */
    double      elapse_time;          /* @d24993 tlg */
    double      adjusted_time;
    char        start_stamp[50];      /* start time stamp for block */
    char        end_stamp[50];        /* end time stamp for block */
    char        tag[50];              /* block tag */
    char        qry_description[100];
    struct stmt_info *next;           /* @d24993 tlg */
};

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

    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 tlg */

    int eo_infile;
    int time_stamp;
    int eo_block;                  /* @d30369 tlg */
    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 sqllda *sqllda, int *col_lengths); /* @d22817 tlg */
void echo_sqllda(struct sqllda *sqllda, int *col_lengths);
void allocate_sqllda(struct sqllda *sqllda);

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

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

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

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

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

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

/* Semaphore control functions */
void create_semaphores(struct global_struct *g_struct);
void throughput_wait(struct global_struct *g_struct);
void runpower_wait(struct global_struct *g_struct, int sem_num);
void release_semaphore(struct global_struct *g_struct, int sem_num);
#ifdef SQLWINT
HANDLE open_semaphore(struct global_struct *g_struct, int num);
#else
int open_semaphore(struct global_struct *g_struct);
#endif

EXEC SQL INCLUDE SQLCA;

/*****
*****/
/* Declare the SQL host variables. */

```

```

/*****
*****/
EXEC SQL BEGIN DECLARE SECTION;

char  stmt_str1[4000] = "\0"; /* Assume max SQL statment
                                of 4000 char */
struct {
    short len;
    char data[32700];
} stmt_str; /* jen LONG */

char  dbname[9] = "\0";
char  userid[9] = "\0";
char  passwd[9] = "\0";
char  sourcefile[256]; /* used for semaphores and table functions?*/
sqlint32 chunk = 0; /* jenCI counter for within the set of chunks*/

EXEC SQL END DECLARE SECTION;

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

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

/* Other globals */
FILE *instream, *outstream; /* File pointers */
int verbose = 0; /* Verbose option flag */
int semcontrol = 1; /* allows/disallows smaphores usage */
int updatePairStart; /* update pair to start at */
int currentUpdatePair; /* update pair running */
int updatePairStop; /* update pair to stop before */
char newtime[50] = "\0"; /* Des - moved from get_time_stamp */
char outstreamfilename[256]; /* store filename of outstream
                               wlc 081397 */
int inlistmax = 400; /* define # of keys to delete at a time
                     wlc 081897 */
int sqlda_allocated = 0; /* fixing free() problem in NT
                          wlc 090597 */
int iImportStagingTbl=0; /* IMPORT use import or load (default)
*/
char temp_time_stamp[50]; /* holds end timestamp to be copied
into start_time_stamp of next query bbeaton */
Timer_struct temp_time_struct; /* holds end time value to be copied
into start_time of next query bbeaton */

/* constants for the semaphores used; 1 for throughput and 2 for power */
#define INSERT_POWER_SEM 1
#define QUERY_POWER_SEM 2
#define THROUGHPUT_SEM 1

/*****
*****/
/* Start main program processing. */
/*****
*****/
int main(int argc, char *argv[])
{

```

```

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", FALSE, 0,
  NULL, "\0", "\0", "\0", NULL };

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

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

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

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

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

if ((g_struct.c_l_opt->update == 1) && (semcontrol == 1))
/* runpower: wait for insert function to complete */
/* waiting on the INSERT_POWER_SEM semaphore */
    runpower_wait(&g_struct, INSERT_POWER_SEM);

strcpy(temp_time_stamp, "\0");

/*****
*****/
*
*
* This is the transition from the "driver" to the "SUT"
*
*
*****/
/*****
*****/
/* Read in each statement, prepare, execute, and send output to file. */

/*****
*****/
while (!c_flags.eo_infile) { /* Check to see if there's no more input */

```

```

c_flags.eo_block = 0;

if (c_l_opt.outfile)
    output_file(&g_struct); /* determine appropriate name for output files */
if ((g_struct.c_l_opt->update != 3) && (g_struct.c_l_opt->update != 4))
{
    if (!strcmp(temp_time_stamp, "0")) /* if first query, get timestamp */
    {
        get_start_time(&start_time);
        strcpy(g_struct.s_info_ptr->start_stamp,
            get_time_stamp(T_STAMP_FORM_3,&start_time)); /*
TIME_ACC jen*/
    }
    else /* else get the end timestamp of previous query */
    {
        strcpy(g_struct.s_info_ptr->start_stamp, temp_time_stamp);
        start_time = temp_time_struct;
    }
    /* write the start timestamp to the file...if this is not a qualification */
    /* run, then write the seed used as well */

    fprintf( outstream,"Start timestamp %*.*s\n",
        T_STAMP_3LEN,T_STAMP_3LEN,          /* TIME_ACC
jen*/
        g_struct.s_info_ptr->start_stamp);
    if (c_l_opt.intStreamNum >= 0)
    {
        if (g_struct.lSeed == -1)
        {
            fprintf( outstream,"Using default qgen seed file");
        }
        else
            fprintf( outstream,"Seed used = %ld",g_struct.lSeed);

        fprintf( outstream,"\n");
    }
}
do { /* Loop through these statements as long as we haven't reached
the end of the input file or the end of a block of statements
*/

    /** Read in the next statment **/
    c_flags.select_status=Get_SQL_stmt(&g_struct);

    if (PreSQLprocess(&g_struct, &start_time) == FALSE)
        /* if after reading the next statement we see that we should
        exit this loop (i.e. eof, update functions, etc...), get out
        */
        break;

}

/*****
*
* The SQLprocess function implements the implementation specific
layer.
* It can handle arbitrary SQL statements.
*
*****/

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

/* If we've got up to here then processing
a regular SQL statement */
SQLprocess(&g_struct);

} while ((!c_flags.eo_block) && (!c_flags.eo_infile)); /* @d30369 tjg
*/

```

```

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

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

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

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

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

return(0);

} /* end of main */

/*****
*****/
/* Generic form of Sleep */
int SleepSome( int amount)
{
#ifdef SQLWINT
    sleep (amount);
#else
    Sleep (amount*1000); /* 10x for NT DJD Changed "sleep" to
"Sleep" */
#endif
    return 0;
}

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

/*****
*****/
/* Get environment variables. (sks May 25 98)
*****/

int get_env_vars(void) {
    if (strcpy(env_tpcd_dbname, getenv("TPCD_DBNAME")) == NULL) {
        fprintf(stderr, "\n The environment variable $TPCD_DBNAME is not
setup correctly.\n");
        return -1;
    }
    if (strcpy(env_user, getenv("USER")) == NULL) {
        fprintf(stderr, "\n The environment variable $USER is not setup
correctly.\n");
        return -1;
    }
    if (strcpy(env_tpcd_audit_dir, getenv("TPCD_AUDIT_DIR")) == NULL)
    {
        fprintf(stderr, "\n The environment variable $TPCD_AUDIT_DIR is not
setup correctly.\n");
        return -1;
    }
}

```

```

    }
    if (strcpy(env_tpcd_tmp_dir, getenv("TPCD_TMP_DIR")) == NULL) {
        fprintf(stderr, "\n The environment variable $TPCD_TMP_DIR is not
        setup correctly.\n");
        return -1;
    }
    #if 0
    if (strcpy(env_tpcd_path_delim, getenv("TPCD_PATH_DELIM")) ==
    NULL ||
        (strcmp(env_tpcd_path_delim, "/") && strcmp(env_tpcd_path_delim,
        "\\"))){
        fprintf(stderr, "\n The environment variable $TPCD_PATH_DELIM is
        not setup correctly , env_tpcd_path_delim'%s'.\n", env_tpcd_path_delim);

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

    return 0;
}

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

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

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

    int good_sql = 0; /* good-sql stmt flag @d23684 tlg */
    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") == 0) { /* 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 tlg */
                    if (strstr(input_ln, "--EOBLK") == input_ln) {
                        g_struct->c_flags->eo_block = 1;
                        return TPCDBATCH_EOBLOCK;
                    }
                if (strstr(input_ln, "-- Query") == input_ln)
                    strcpy(g_struct->s_info_ptr->qry_description, input_ln);

                if (strstr(input_ln, "--TAG") == input_ln)
                    strcpy(g_struct->s_info_ptr->tag, (input_ln+sizeof("--TAG")));

                /* if we're using update functions, return that info
                appropriately */
                if (g_struct->c_l_opt->update != 0) {
                    if (strstr(input_ln, "--INSERT") == input_ln)
                        return TPCDBATCH_INSERT;

                    if (strstr(input_ln, "--DELETE") == input_ln)
                        return TPCDBATCH_DELETE;
                }

                if (strstr(input_ln, "--COMMENT") == input_ln) { /* @d25594 tlg
                */
                    temp_ptr = (input_ln + 11); /* User-specified comments go to
                    the outfile */

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

                    eostmt=0;
                }

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

```

```

        fprintf(outstream, "\n%s\n", g_struct->s_info_ptr->qry_description);

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

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

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

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

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

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

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

} while (!eostmt);

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

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

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

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

else /* @d23684 tjg */
    return TPCDBATCH_NONSQL;

} /* Get_SQL_stmt */

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

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

    for (loopvar=0; loopvar<sqlda->sqld; loopvar++)
    {
        switch (sqlda->sqlvar[loopvar].sqltype)
        {
            case SQL_TYP_INTEGER: /* INTEGER */
            case SQL_TYP_NINTEGER:
                if ((sqlda->sqlvar[loopvar].sqldata=
                    (TPCDBATCH_CHAR *)malloc(sizeof(sqlint32))) == NULL)
                    mem_error("allocating INTEGER");
                break;
            case SQL_TYP_BIGINT: /* BIGINT */
            /*kmwBIGINT*/
            case SQL_TYP_NBIGINT:
                //DJD Re activated the IFDef
                #ifdef SQLWINT
                    if ((sqlda->sqlvar[loopvar].sqldata=
                        (TPCDBATCH_CHAR *)malloc(sizeof(__int64))) == NULL)
                        #else
                            if ((sqlda->sqlvar[loopvar].sqldata=
                                (TPCDBATCH_CHAR *)malloc(sizeof(sqlint64))) == NULL)
                                #endif
                                    mem_error("allocating BIGINT");
                    break;
            case SQL_TYP_CHAR: /* CHAR */
            case SQL_TYP_NCHAR:
                if ((sqlda->sqlvar[loopvar].sqldata=
                    (TPCDBATCH_CHAR *)calloc(256, sizeof(char))) == NULL)
                    mem_error("allocating CHAR/VARCHAR");
                break;
            case SQL_TYP_VARCHAR: /* VARCHAR */
            case SQL_TYP_NVARCHAR:
                if ((sqlda->sqlvar[loopvar].sqldata=
                    (TPCDBATCH_CHAR *)calloc(4002, sizeof(char))) == NULL)
                    mem_error("allocating CHAR/VARCHAR");
                break;
            case SQL_TYP_LONG: /* LONG VARCHAR */
            case SQL_TYP_NLONG:
                if ((sqlda->sqlvar[loopvar].sqldata=
                    (TPCDBATCH_CHAR *)calloc(32702, sizeof(char))) ==
                    NULL)
                        mem_error("allocating VARCHAR/LONG VARCHAR");
                break;
            case SQL_TYP_FLOAT: /* FLOAT */
            case SQL_TYP_NFLOAT:
                if ((sqlda->sqlvar[loopvar].sqldata=
                    (TPCDBATCH_CHAR *)malloc(sizeof(double))) == NULL)
                    mem_error("allocating FLOAT");
                break;
            case SQL_TYP_SMALL: /* SMALLINT */
            case SQL_TYP_NSMALL:
                if ((sqlda->sqlvar[loopvar].sqldata=

```

```

        (TPCDBATCH_CHAR *)malloc(sizeof(short))) == NULL)
        mem_error("allocating SMALLINT");
        break;
    case SQL_TYP_DECIMAL:                /* DECIMAL */
    case SQL_TYP_NDECIMAL:
        if ((sqlda->sqlvar[loopvar].sqldata=
            (TPCDBATCH_CHAR *)malloc(20)) == NULL)
            mem_error("allocating DECIMAL");
        break;
    case SQL_TYP_CSTR:                    /* VARCHAR (null terminated) */
    case SQL_TYP_NCSTR:
        if ((sqlda->sqlvar[loopvar].sqldata=
            (TPCDBATCH_CHAR *)calloc(4001,sizeof(char))) == NULL)
            mem_error("allocating CHAR/VARCHAR");
        break;
    case SQL_TYP_DATE:                    /* DATE */
    case SQL_TYP_NDATE:
        if ((sqlda->sqlvar[loopvar].sqldata=
            (TPCDBATCH_CHAR *)calloc(13,sizeof(char))) == NULL)
            mem_error("allocating DATE");
        break;
    case SQL_TYP_TIME:                    /* TIME */
    case SQL_TYP_NTIME:
        if ((sqlda->sqlvar[loopvar].sqldata=
            (TPCDBATCH_CHAR *)calloc(11,sizeof(char))) == NULL)
            mem_error("allocating TIME");
        break;
    case SQL_TYP_STAMP:                    /* TIMESTAMP */
    case SQL_TYP_NSTAMP:
        if ((sqlda->sqlvar[loopvar].sqldata=
            (TPCDBATCH_CHAR *)calloc(29,sizeof(char))) == NULL)
            mem_error("allocating TIMESTAMP");
        break;
    }
    if ((sqlda->sqlvar[loopvar].sqlind=
        (short *)calloc(1,sizeof(short))) == NULL)
        mem_error("allocating indicator");
}
sqlda_allocated = 1; /* fix free() problem on NT
                    wlc 090597 */
return; /* allocate_sqlda */
}

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

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

    int col_type;           /* Type of column */

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

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

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

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

                    fprintf(outstream, "%*ld ",col_lengths[col],
                        *(sqlint32 *) (sqlda->sqlvar[col].sqldata));
                    break;

                case SQL_TYP_BIGINT: /*kmwBIGINT*/
                case SQL_TYP_NBIGINT:
#ifdef SQLWINT
                    fprintf(outstream, "%*I64d ",col_lengths[col],
                        *(__int64 *) (sqlda->sqlvar[col].sqldata));
#else
                    fprintf(outstream, "%*lld ",col_lengths[col],
                        *(sqlint64 *) (sqlda->sqlvar[col].sqldata));
#endif
                    break;

                case SQL_TYP_CHAR:
                case SQL_TYP_NCHAR:

                    fprintf(outstream, "%-*s ",col_lengths[col],sqlda-
>sqlvar[col].sqldata);
                    break;
                case SQL_TYP_VARCHAR:
                case SQL_TYP_NVARCHAR:
                case SQL_TYP_LONG:
                case SQL_TYP_NLONG: /* @d30369 tlg */
                    ((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;

#ifdef (SQLOS2) || defined (SQLWINT) || defined (SQLWIN) || defined (SQLDOS)
    /*@d33143aha*/
    ftime(start_time);
#elif defined (SQLSNI)
    rc = gettimeofday(start_time);
#elif defined (SQLPTX)
    gettimeofday_mapped(start_time);
    rc = 0; /* gettimeofday_mapped returns void */
#elif defined (SQLUNIX) || defined (SQLAIX) /*TIMER jen*/
    rc = gettimeofday(start_time, NULL);
#else
    #error Unknown operating system
#endif

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

/*****/
/* Calculate and return the elapsed time given a starting time. */
/*****/

double get_elapsed_time (Timer_struct *start_time)
{
    int status = 0;
    Timer_struct end_time;
    double result = -1.0;
#ifdef SQLWINT
    long int result_sec;
    long int result_usec;
#endif

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

    /*
     * translate the time to that rounded to the CLOSEST 0.1 seconds as
     * required by the TPC-D spec. ROUNDING
     */
    /* result = (double)((long)((result + 0.099999) * 10))/10.0; */
    result = (double)((long)((result + 0.05) * 10))/10.0;
    return (result);
}

```

```

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

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

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

    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 tjjg */

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

        EXEC SQL CONNECT ; /* @d28763 tjjg */

        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 = %d\n", temp_sqlca.sqlcabc);
            fprintf(outstream, "SQLCODE = %d\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("\ntpcdbatch [-d dbname] [-f file_name] [-l file_name] [-r on/off]");
    printf("\n [-v on/off] [-b on/off] [-u p/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      Qualification - -1");
printf("\n      Power - 0");
printf("\n      Throughput - >= 1 (actual number depends on the
current query stream");
printf("\n      -m Maximum number of keys to delete at a time");
printf("\n      Default - 400");
printf("\n      -h Display this help screen");
printf("\n      -p turns smeaphores on or off");
printf("\n      Default - off");

printf("\n\nControl statements specifying output and performance details");
printf("\ncan be included before SQL statements; they will apply for");
printf("\nthat and subsequent statements until updated.");

printf("\n\nSyntax:  --#SET <control option> <value>");
printf("\n 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--#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 tlg */
                case 'F' :
                    strcpy(g_struct->c_l_opt->infile,argv[++loopvar]);
                    break;

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

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

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

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

                case 'u' : /* @d26350 tlg */
                case 'U' :
                    g_struct->c_l_opt->update=-1; /* init to invalid number */
                    if (!strcmp(uppercase(argv[++loopvar]),"P1"))
                        g_struct->c_l_opt->update=1; /* power query stream */
                    if (!strcmp(uppercase(argv[loopvar]),"P2"))
                        g_struct->c_l_opt->update=3; /* power update with updates */
                    if (!strcmp(uppercase(argv[loopvar]),"P"))
                        g_struct->c_l_opt->update=4; /* power update without updates */
                    if (!strcmp(uppercase(argv[loopvar]),"T1"))
                        g_struct->c_l_opt->update=0; /* throughput query stream */
                    if (!strcmp(uppercase(argv[loopvar]),"T2"))
                        g_struct->c_l_opt->update=2; /* throughput update with updates

                    */

                    if (!strcmp(uppercase(argv[loopvar]),"T"))
                        g_struct->c_l_opt->update=5; /* throughput update without
updates */

```

```

        break;

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

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

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

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

    case 'p' :
    case 'P' :
        if (!strcmp(uppercase(argv[++loopvar]),"ON")) /* bbe 072599 */
            semcontrol = 1;
        else
            semcontrol = 0;
        break;

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

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

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

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

        /* invocation */

```

```

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

    } /* end switch */
} /* end if */

loopvar ++;
} /* end while */

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

#ifdef PARALLEL_UPDATES
    if (running_updates) {
        if (updatePair == -1) {
            fprintf (stderr, "The parameters to tpcdbatch have not been passed
correctly\n");
            exit (-1);
        }
        else {
            /* check to see if we are to use copy on for the load */
            if ((getenv("TPCD_LOG") != NULL ) &&
                (!strcmp(uppercase(getenv("TPCD_LOG")), "YES")))
            {
                /* okay, we have set LOG_RETAIN on so we need to use copy
directory */
                copyOnOrOff = TRUE;
            }
            else
            {
                /* log retain off don't use copy directory */
                copyOnOrOff = FALSE;
            }

            if (function == 1)
                /* runUF1_fn (updatePair, updateStream); aph 981205 */
                runUF1_fn (updatePair, updateStream, dbname, userid, passwd);
            else
                if (function == 2) {
                    /*
                    fprintf(stderr, "A-Calling runUF2_fn %d %d %d ...\n",
                        updatePair, updateStream, deleteChunk);

                    */
                    /* runUF2_fn (updatePair, updateStream, deleteChunk); aph
981205 */
                    runUF2_fn (updatePair, updateStream, deleteChunk, dbname,
userid, passwd);
                }
            else {
                fprintf (stderr, "Wrong function to tpcdbatch\n");
                exit (-1);
            }
            exit (0);
        }
    }
#endif /* PARALLEL_UPDATES */

/* If no database name is given, then use the one specified in the
environment variable DB2DBDFT, otherwise error */

```

```

if (!strcmp(dbname,"\\0")) {
    testptr = getenv("DB2DBDFT");
    if (testptr == NULL) {
        fprintf(stderr, "\\nNo database name has been specified on command ");
        fprintf(stderr, "line\\nnor in environment variable DB2DBDFT.");
        display_usage();
    }
    else
        strcpy(dbname,testptr);
}

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;
    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;
#ifdef 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 */
    }

```

```

/* 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%sseedme", env_tpcd_audit_dir,
env_tpcd_path_delim, env_tpcd_path_delim);

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

/* check to see if we are to use copy on for the load */
if ((getenv("TPCD_LOG") != NULL ) &&
(!strcmp(uppercase(getenv("TPCD_LOG")), "YES")))
{
    /* okay, we have set LOG_RETAIN on so we need to use copy directory
*/
    g_struct->copy_on_load = TRUE;
}
else
{
    /* log retain off don't use copy directory */
    g_struct->copy_on_load = FALSE;
}

/*****
*****/
/* Make sure that DB2 is started. */
/* CONNECT now unless this is a UF stream for a Throughput test. */
/* (aph 98/12/22) */
/*****
*****/

if (g_struct->c_l_opt->update > 1)
{
    /* This is an update function stream in a throughput run. */
    /* Just make sure that DB2 is started. Each UF child will CONNECT
itself. */
    if (verbose) fprintf(stderr, "\nStarting the DB2 Database Manager
Now\n");
    sqlstar ();
}
else
{
    /* In all other cases, CONNECT to the target database. */
    do
    {
        if (!strcmp(userid, "\0")) /* No authentication provided */
            EXEC SQL CONNECT TO :dbname;
        else EXEC SQL CONNECT TO :dbname USER :userid USING
:passwd;
        if (sqlca.sqlcode == SQLE_RC_NOSTARTG) {
            if (verbose)
                fprintf(stderr, "\nStarting the DB2 Database Manager Now\n");
            sqlstar ();
            connect=0;
        }
        else connect=1;
    } while (!connect);
    error_check();
}
}

```

```

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

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

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

/* if we are running a throughput test, then we must report the */
/* stream count information...we will report one file per stream */
/* and amalgamate them after all streams have completed */
/* if the number of streams is greater than 0 then this is a throughput test*/
switch (g_struct->c_l_opt->update)
{
    case (2):
    case (5):
        /* update throughput function stream */
        sprintf(file_name,"%s%sstcntuf.%s",g_struct->run_dir,
            env_tpcd_path_delim, g_struct->file_time_stamp);
        break;
    case (3):
    case (4):
        /* update power function stream */
        sprintf(file_name,"%s%spstcntuf.%s",g_struct->run_dir,
            env_tpcd_path_delim, g_struct->file_time_stamp);
        break;
    case (1):
        /* power query stream */
        sprintf(file_name, "%s%spstcnt%d.%s",g_struct->run_dir,
            env_tpcd_path_delim,
            g_struct->c_l_opt->intStreamNum,g_struct-
>file_time_stamp);
        break;
    case (0):
        /* throughput query stream */
        sprintf(file_name, "%s%ssstcnt%d.%s",g_struct->run_dir,
            env_tpcd_path_delim,
            g_struct->c_l_opt->intStreamNum,g_struct-
>file_time_stamp);
        break;
}

if( (g_struct->stream_report_file = fopen(file_name, WRITEMODE)) ==
NULL )
{
    fprintf(stderr, "\nThe output file for the stream count information\n");
    fprintf(stderr, "could not be opened, make sure the filename is correct\n");
    fprintf(stderr, "filename = %s\n", file_name);
    exit(-1);
}

if (g_struct->c_l_opt->update > 1)
{
    /* update function stream */
    fprintf(g_struct->stream_report_file,
        "Update function stream starting at %*. *s\n",
        T_STAMP_3LEN,T_STAMP_3LEN, /* TIME_ACC jen*/

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

#ifndef LINUX

    fclose(g_struct->stream_report_file);

#endif

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

    sprintf(g_struct->update_num_file, "%s%s%s.%s.update.pair.num",
env_tpcd_audit_dir,
    env_tpcd_path_delim, uppercase(env_tpcd_dbname),
lowercase(env_user));
    sprintf(g_struct->sem_file, "%s.%s.semfile", env_tpcd_dbname, env_user);
    if (g_struct->c_l_opt->intStreamNum == 0)
    {
        sprintf(g_struct->sem_file2, "%s.%s.semfile2", env_tpcd_dbname,
env_user);
    }
    if (verbose) { /* print out the update pair number file for debugging */
        fprintf(stderr, "\n init_setup: strem %d update pair numb file = %s\n",
            g_struct->c_l_opt->intStreamNum,g_struct->update_num_file);
    }

    /* update the
$TPCD_AUDIT_DIR/$TPCD_DBNAME.$USER.update.pair.num file */
    /* update pairs have been run */
    if (( (g_struct->c_l_opt->update >= 1 ) && ( g_struct->c_l_opt->update < 4
))
        /* on or onl, but not */ /* bbe or > 1 */
    {
        updateFP = fopen(g_struct->update_num_file,"r");
        if (updateFP != NULL )
        {
            fscanf(updateFP, "%d",&updatePairStart);
            fclose(updateFP);
            if (g_struct->c_l_opt->intStreamNum == 0) /* on, 1 update pair */
                updatePairStop = updatePairStart + 1;
            else /* only, multiple update pairs, stream number will be total */
                updatePairStop = updatePairStart + g_struct->c_l_opt-
>intStreamNum;
            currentUpdatePair = updatePairStart;

            if (updatePairStart <= 0)
            {
                fprintf(stderr, "updatePairStart is bogus!");
                exit(-1);
            }
        }
        else
        {
            fprintf(stderr, "\n %s not set up, set this \n",g_struct->update_num_file);
            fprintf(stderr, "file to contain the number of the update pair to \n");
            fprintf(stderr, "run and resubmit\n");
            exit(-1);
        }
    }
}

```

```

    }

    return ;

}

/*****
*****/
/* A function to print out the column titles for a returned set */
/*****
*****/
void print_headings (struct sqlda *sqlda, int *col_lengths)
{
    int col = 0;          /* Column number */
    int col_width = 0;     /* width of column */
    int max_col_width = 0; /* maximum column width */
    int col_name_length = 0; /* sizeof column name string */
    int col_type = 0;      /* column type */

    int total_length = 0;  /* accumulator var. for
                           length of column headings */
    int loopvar = 0;

    char col_name[256] = "\0";
    unsigned char m,n;     /* precision and accuracy
                           for decimal conversion */

    fprintf (outstream, "\n");

    /*** loop through for each column in solution set
    and determine the maximum column width ***/

    for (col = 0; col < sqlda->sqlc; 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 tlg */
                col_lengths[col] = TPCDBATCH_MAX (col_name_length, 6);
                break;
            case SQL_TYP_INTEGER:
            case SQL_TYP_NINTEGER:
                col_lengths[col] = TPCDBATCH_MAX (col_name_length, 11);
                break;
            case SQL_TYP_BIGINT: /*kmwBIGINT*/
            case SQL_TYP_NBIGINT:
                col_lengths[col] = TPCDBATCH_MAX (col_name_length, 19);
                break;
            case SQL_TYP_CSTR:
            case SQL_TYP_NCSTR:
            case SQL_TYP_DATE:
            case SQL_TYP_NDATE:
            case SQL_TYP_TIME:
            case SQL_TYP_NTIME:
            case SQL_TYP_STAMP:
            case SQL_TYP_NSTAMP:
            case SQL_TYP_CHAR:
            case SQL_TYP_NCHAR:
            case SQL_TYP_VARCHAR:
            case SQL_TYP_NVARCHAR:
            case SQL_TYP_LONG:
            case SQL_TYP_NLONG:
                col_lengths[col] = TPCDBATCH_MAX (col_name_length, col_width);
                break;

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

            case SQL_TYP_DECIMAL:
            case SQL_TYP_NDECIMAL:

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

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

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

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

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

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

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

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

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

    if ((form == T_STAMP_FORM_1) || (form == T_STAMP_FORM_3))
    {
        /* SUN fix bbe start */
    #if defined (SQLWINT) || defined (SQLWIN) || defined (SQLOS2) ||
defined (SQLDOS)
        timeLength = strftime(newtime, 50, "%x %X", tp);

```

```

#elif defined (SQLUNIX) || defined (SQLAIX)
    timeLength = strftime(newtime,50,"%D %T",tp); /* SUN ...test this */
#else
#error Unknown operating system
#endif
    /* SUN fix bbe stop */
    /* TIME_ACC jen start*/
    if (form == T_STAMP_FORM_3)
    {
        /* concatenate the microsecond/milliseconds on the end of the */
        /*timestamp jen1006 */
    }
    #if defined (SQLUNIX) || defined (SQLAIX)
        sprintf(newtime+timeLength,"%0.6d",temp_stamp.tv_usec);
    #elif defined (SQLOS2) || defined (SQLWINT) || defined (SQLWIN) ||
    defined (SQLDOS))
        sprintf(newtime+timeLength,"%0.3d",temp_stamp.millitm);
    #else
#error Unknown operating system
#endif
    /* TIME_ACC jen stop*/
    }
    }
    else
        if (form == T_STAMP_FORM_2)
            strftime(newtime,50,"%y%m%d-%H%M%S",tp);

    return (newtime);
}

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

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

    double adjusted_a_mean = 0;
    double adjusted_g_mean = 0;
    double adjusted_g_mean_intern;
    double adjusted_max_time = 0;

    double Ts = 0; /* different TPC-D metrics */
    double Ts1;
    double Ts2;
    /* double QppD = 0; MARK
    double QthD = 0;
    double QphD = 0; */

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

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

    /* Determine the size of the database from the scale factor (1 SF = 1GB) */
    if (g_struct->scale_factor < 1.0) {

```

```

        db_size = g_struct->scale_factor * 1000;
        strcpy(db_size_qualifier, "MB");
    } else if (g_struct->scale_factor >= 1000.0) {
        db_size = g_struct->scale_factor / 1000;
        strcpy(db_size_qualifier, "TB");
    } else {
        db_size = g_struct->scale_factor;
        strcpy(db_size_qualifier, "GB");
    }

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

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

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

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

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

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

    fprintf( outstream, "\n\nSummary of
Results\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->elapsed_time > max->elapsed_time)
            max = s_info_ptr;
        else
            if ((s_info_ptr->elapsed_time < min->elapsed_time)
                && (s_info_ptr->elapsed_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->elapsed_time != -1) {
        s_info_ptr->adjusted_time = s_info_ptr->elapsed_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->elapsed_time/min->elapsed_time > 1000)
            {
                /* jmc fix geo_mean calculation...round adjusted time properly
ROUNDING*/
                adjusted_max_time = max->elapsed_time/1000;
                if (s_info_ptr->elapsed_time < adjusted_max_time)
                {
                    s_info_ptr->adjusted_time =
                        (double)((long)((adjusted_max_time + 0.05) * 10))/10.0;
                    if (s_info_ptr->adjusted_time < 0.1)
                        s_info_ptr->adjusted_time = 0.1;
                }
                /*jmc fix geo_mean calculation...round adjusted time properly
ROUNDING end*/
            }
            /* a value was calculated */
            fprintf (outstream,
                "%-5d %-5.5s %15.1f %15.1f %*.*s %*.*s\n",
                s_info_ptr->stmt_num,s_info_ptr->tag,
                s_info_ptr->elapsed_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->elapsed_time;
                adjusted_a_mean += s_info_ptr->adjusted_time;
            }

            if (s_info_ptr->elapsed_time > 0) { /* don't bother finding log of
                numbers < 0 */

```

```

                geo_mean += log(s_info_ptr->elapsed_time);
                adjusted_g_mean += log(s_info_ptr->adjusted_time);
            }

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

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

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

    fprintf(outstream, "\n\nNumber of statements: %d\n\n", s_info_ptr-
>stmt_num - 1);
    /* Calculate the arithmetic and geometric means */

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

    /* print out all the appropriate information including the
different TPC-D metrics */
    /* do not bother with this if we are in an update only stream */
    fprintf (outstream, "\nGeom. mean queries %7.3f %15.3f\n",\
        geo_mean,adjusted_g_mean);
    if (g_struct->c_l_opt->update < 2)
    {
        fprintf (outstream, "Arith. mean queries %7.3f %15.3f\n",\
            arith_mean,adjusted_a_mean);

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

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

        /* Increase the Ts measurement by one second since the accuracy of our
*/
        /* timestamps is only to 1 second and if the start was at 1.01 seconds, */

```

```

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

#if defined (SQLUNIX) || defined (SQLAIX)
    Ts = g_struct->stream_end_time.tv_sec - g_struct-
>stream_start_time.tv_sec + 1;
    Ts1 = (double)g_struct->stream_start_time.tv_sec + ((double)g_struct-
>stream_start_time.tv_usec/1000000);
    Ts2 = (double)g_struct->stream_end_time.tv_sec + ((double)g_struct-
>stream_end_time.tv_usec/1000000);

#elif defined (SQLOS2) || defined (SQLWINT) || defined (SQLWIN) ||
defined (SQLDOS))
    Ts = g_struct->stream_end_time.time - g_struct->stream_start_time.time
+ 1;
    Ts1 = (double)g_struct->stream_start_time.time + ((double)g_struct-
>stream_start_time.millitm/1000);
    Ts2 = (double)g_struct->stream_end_time.time + ((double)g_struct-
>stream_end_time.millitm/1000);

#else
#error Unknown operating system
#endif

/* TIME_ACC jen stop*/

/* MARK
##Now do in calcmetrics.pl##
QppD = (3600 * g_struct->scale_factor) / adjusted_g_mean;
QthD = (num_stmt * 3600 * g_struct->scale_factor) / Ts;
QphD = sqrt(QppD*QthD);
*/
/* if the decimal part has some meaningful value then print the database
size
with decimal part; otherwise just print the integer part */

fprintf (outstream,
        "\nGeometric mean interim value = %10.3f\n\nStream Ts %11 =
%10.0f\n\nStream start int representation %11 = %f\n\nStream stop int
representation %11 = %f",
        adjusted_g_mean_intern,Ts,Ts1,Ts2);
}

}

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

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

    free(sqlda);

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

/*****
**/
/* processing to run the insert update function */
/*****
**/
void runUF1 ( struct global_struct *g_struct, int updatePair )
{
    char statement[3000];
    char sourcedir[256];

    int split_updates = 2; /* no. of ways update records are split */
    int concurrent_inserts = 2; /* jenCI no of concurrent updates to be */
    /* jenCI run at once*/
    int loop_updates = 1; /* jenCI no of updates to be run in one */
    /* jenCI "concurrent" invocation. should*/
    /* jenCI be split_updates / concurrent_inserts*/

    int i;
    int streamNum;
#ifdef SQLWINT
    /* PROCESS_INFORMATION childprocess[100]; */
    char cmdline[256];
    HANDLE su_hSem;
    char UF1_semaphore[256];
#else
    int childpid[100];
    int su_semid; /* semaphore for controlling split updates*/
    key_t su_semkey; /* key to generate semid */
#endif
    if (g_struct->c_l_opt->intStreamNum == 0)
        streamNum = 0;
    else
        streamNum = currentUpdatePair - updatePairStart + 1;

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

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

    /* Load the orderkeys into the staging table */
    /* In SMP environments one could use a load command but by using a */
    /* script we can keep the code common */
#ifdef SQLWINT
    sprintf (statement, "perl %s\\tools\\ploaduf1 %d\n", sourcedir, updatePair);
#else
    sprintf (statement, "perl %s/tools/ploaduf1 %d 1", sourcedir, updatePair);
#endif
    if (system(statement))
    {
        fprintf (stderr, "ploaduf1 failed for UF1, examine UF1.log for cause.
Exiting.\n");
        if (verbose)
            fprintf (stderr,
                    "ploaduf1 failed for UF1, examine UF1.log for cause. Exiting.\n");
        exit (-1);
    }

    fprintf (outstream, "load_update finished for UF1.\n");

    if (getenv ("TPCD_SPLIT_UPDATES") != NULL)
        split_updates = atoi (getenv ("TPCD_SPLIT_UPDATES"));

```

```

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

#ifdef SQLWINT
/* we will use the tpcd.setup file to generate the semaphore key */
if (getenv("TPCD_AUDIT_DIR") != NULL) /*begin SEMA */
{
    /* this is assuming that you will be running this from 0th node */
    sprintf(sourcefile, "%s%ctools%ctpcd.setup",
        getenv("TPCD_AUDIT_DIR"), PATH_DELIM,PATH_DELIM);
}
else
{
    fprintf (stderr, "runUF1 Can't open UF1 semaphore
file,TPCD_AUDIT_DIR is not defined.\n");
    exit (-1);
}
/*end SEMA */
su_semkey = ftok (sourcefile, 'J');
if ( (su_semid = semget (su_semkey, 1,
IPC_CREAT|S_IRUSR|S_IWUSR)) < 0)
{
    fprintf (stderr, "Cannot get semaphore! semget failed: errno =
%d\n",errno);
    exit (-1);
}
#else /* SQLWINT */
sprintf (UF1_semfile, "%s.%s.UF1.semfile", env_tpcd_dbname, env_user);
su_hSem = CreateSemaphore(NULL, 0,
                        concurrent_inserts,          /*jenCI*/
                        (LPCWSTR)(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*/
{
#ifdef 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 */
#ifdef SQLWINT
if (sem_op (su_semid, 0, concurrent_inserts * -1) != 0) /*jenCI*/
{
    /*jenSEM*/
    fprintf(stderr,
        "Failure to wait on insert semaphore with %d of children\n",
        concurrent_inserts);
    exit(1);
} /*jenSEM*/
semctl (su_semid, 0, IPC_RMID, 0);
#else
for (i = 0; i < concurrent_inserts; i++) /*jenCI*/
{
    if (verbose)
    {
        fprintf(stderr, "About to wait again ...Sets to wait for %d\n",
            concurrent_inserts - i); /*jenCI*/
    }
    if (WaitForSingleObject(su_hSem, INFINITE) == WAIT_FAILED)
    {
        fprintf(stderr,
            "WaitForSingleObject (su_hSem) failed in runUF1 on set %d,
error: %d, quitting\n",
            i, GetLastError());
        exit(-1);
    }
}
if (! CloseHandle(su_hSem))
{
    fprintf(stderr,
        "RunUF1 Close Sem failed - Last Error: %d\n", GetLastError());
    /* no exit here */
}
#endif

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

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

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

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

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

    int i;
    int streamNum;
#ifdef SQLWINT
    char commandline[256];
    HANDLE su_hSem;

```

```

char          UF2_semfile[256];
#else
int childpid[100];
char sourcefile[256];
int          su_semid; /* semaphore for controlling split updates*/
key_t       su_semkey; /* key to generate semid */
#endif
if (g_struct->c_l_opt->intStreamNum == 0)
    streamNum = 0;
else
    streamNum = currentUpdatePair - updatePairStart + 1;

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

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

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

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

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

#ifdef SQLWINT
    sprintf (statement, "perl %s\tools\ploaduf2 %d\n", sourcedir, updatePair);
#else
    sprintf (statement, "perl %s/tools/ploaduf2 %d 2", sourcedir, updatePair);
#endif
if (system(statement))
{
    fprintf (stderr, "ploaduf2 failed for UF2, examine UF2.log for cause.
Exiting.\n");
    exit (-1);
}
fprintf (outstream, "ploaduf2 finished for UF2.\n");

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

/* Next we need to get ready to launch a bunch of concurrent processes */
#ifdef SQLWINT
/* we will use the tpcd.setup file to generate the semaphore key begin
SEMA */
if (getenv("TPCD_AUDIT_DIR") != NULL)
{
    sprintf(sourcefile, "%s%ctools%ctpcd.setup",
        getenv("TPCD_AUDIT_DIR"), PATH_DELIM, PATH_DELIM);
}

```

```

else
{
    fprintf (stderr, "runUF2 Can't open UF2 semaphore file,
TPCD_AUDIT_DIR is not defined.\n");
    exit (-1);
}

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

for (i=0; i < concurrent_deletes; i++)
{
#ifdef 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 */
#endif SQLWINT
fprintf(stderr, "About to wait on the semaphore...\n");
if (sem_op (su_semId, 0, concurrent_deletes * -1) != 0)
/*jenSEM*/
{
    fprintf(stderr,
        "Failure to update wait on delete semaphore with %d children\n",
        concurrent_deletes);
    exit(1);
}
semctl (su_semId, 0, IPC_RMID, 0);
/*jenSEM*/
#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(ostream, "UF2 for update pair %d complete\n", updatePair);
}

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

/*-----*/
/*      General semaphore function.                */
/*-----*/
#endif 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 errno = %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: /* qualifying */
            sprintf(file_name,
                "%s%sqryqual%02d.%s", run_dir, delim, qnum, time_stamp);
            break;
        case 0: /* power tests */
            if (qnum < 0) /* update functions */
                sprintf(file_name,
                    "%s%smps00uf%d.%02d.%s", run_dir, delim, abs(qnum), \
                    currentUpdatePair, time_stamp);
            else
                sprintf(file_name,
                    "%s%smpqry%02d.%s", run_dir, delim, qnum, time_stamp);
            break;

        default:
            /* if (qnum < 0) - replaced by berni 96/03/26 */
            if (g_struct->c_l_opt->update == 2 ||
                g_struct->c_l_opt->update == 5)
                sprintf(file_name, "%s%smts%02duf%d.%02d.%s", run_dir, delim, \
                    currentUpdatePair - updatePairStart + 1, abs(qnum),
                    currentUpdatePair, time_stamp);
            else
                sprintf(file_name, "%s%smts%dqry%02d.%s", run_dir, delim, \
                    g_struct->c_l_opt->intStreamNum, qnum, time_stamp);
            break;
    }

    if (g_struct->c_flags->eo_infile)
    if (g_struct->c_l_opt->update == 2 ||

```

```

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

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

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

return;
}

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

    switch (g_struct->c_flags->select_status)
    {
        case TPCDBATCH_NONSQL:
            g_struct->s_info_stop_ptr = g_struct->s_info_ptr;
            /* if we're at the end of the input file, set the stop
            pointer to this structure */
            rc = FALSE;
            break;

        case TPCDBATCH_EOBLOCK:
            rc = FALSE;
            break;
        case TPCDBATCH_INSERT:
            /* we have to check whether or not this is a throughput */
            /* test, and if it is, we have to set up a semaphore to */
            /* control when the update functions are run. We want */
            /* them to be run after all the query streams have finished. */
            /* What we do is set up the semaphore here, decrement it */
            /* in the query streams, and wait for it to get cleared */
            /* before we allow the UFs to run. */
            /* Note: we only set up the semaphore if: */
            /* 1. we are running the throughput test (num of */
            /* streams > 0) */
            /* 2. we are at the first UF1 (i.e. this is the */
            /* case where currentUpdatePair = updatePairStart */
            /* we also want to check the sem_on element in the global */
            /* structure to see if we want to use semaphores or let */
            /* the calling script do the synchronization of the update */
            /* stream */
            if (semcontrol == 1)
            {
                /* yes we are to be using semaphores */
                /* is this the 1st time into update function 1 (uf1)? */
                if (currentUpdatePair == updatePairStart)
                {
                    /* create the semaphores */
                    create_semaphores(g_struct);
                    if (g_struct->c_l_opt->intStreamNum != 0)
                        /* wait period for runthroughput updates */
                        throughput_wait(g_struct);
                }
                /* otherwise continue to run */
            }
            if ((g_struct->c_l_opt->update == 3) || (g_struct->c_l_opt->update == 4))
            {
                get_start_time(start_time);
                strcpy(g_struct->s_info_ptr->start_stamp,
                    get_time_stamp(T_STAMP_FORM_3, start_time)); /*
TIME_ACC jen*/
                /* write the start timestamp to the file...if this is not a qualification */
                /* run, then write the seed used as well */
                fprintf(outstream, "Start timestamp %*. *s \n",
                    T_STAMP_3LEN, T_STAMP_3LEN, /* TIME_ACC
jen*/
                    g_struct->s_info_ptr->start_stamp);
                if (g_struct->c_l_opt->intStreamNum >= 0)
                {
                    if (g_struct->lSeed == -1)
                    {
                        fprintf(outstream, "Using default qgen seed file");
                    }
                    else
                    {
                        fprintf(outstream, "Seed used = %ld", g_struct->lSeed);
                        fprintf(outstream, "\n");
                    }
                }
                if (g_struct->c_l_opt->update < 4){
                    /* run only if updates are enabled */
                    runUF1(g_struct, currentUpdatePair);
                }

                rc = FALSE;
                if ((g_struct->c_l_opt->intStreamNum == 0) && (semcontrol == 1))
                    /* RUNPOWER: release first semaphore so the queries can run */
                    release_semaphore(g_struct, INSERT_POWER_SEM);
                break;
            case TPCDBATCH_DELETE:
                if ((g_struct->c_l_opt->intStreamNum == 0) && (semcontrol == 1))
                {

```

```

/* RUNPOWER: wait for queries to finish */
/* waiting on QUERY_POWER_SEM semaphore */
runpower_wait(g_struct, QUERY_POWER_SEM);
}
if ((g_struct->c_l_opt->update == 3) || (g_struct->c_l_opt->update == 4))
{
    get_start_time(start_time);
    strcpy(g_struct->s_info_ptr->start_stamp,
           get_time_stamp(T_STAMP_FORM_3, start_time)); /*
TIME_ACC jen*/
    /* write the start timestamp to the file...if this is not a qualification */
    /* run, then write the seed used as well */
    fprintf( outstream, "Start timestamp %*.s\n",
             T_STAMP_3LEN, T_STAMP_3LEN,          /* TIME_ACC
jen*/
            g_struct->s_info_ptr->start_stamp);
    if (g_struct->c_l_opt->intStreamNum >= 0)
    {
        if (g_struct->lSeed == -1)
        {
            fprintf( outstream, "Using default qgen seed file");
        }
        else
            fprintf( outstream, "Seed used = %ld", g_struct->lSeed);
        fprintf( outstream, "\n");
    }
}
if (g_struct->c_l_opt->update < 4){
    /* run only if updates are enabled */
    runUF2(g_struct, currentUpdatePair);
    if (g_struct->c_l_opt->intStreamNum == 0)
    { /* RUNPOWER */
        fprintf(stderr, "UF2 completed\n");
    }
}
currentUpdatePair += 1;
/* update the update.pair.num file to reflect the successfully completed */
/* update pair */
if (g_struct->c_l_opt->update < 4)
{ /*jen*/
#ifdef 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 tjj
*/

/* 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 tjj */

            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_sqla(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))
#ifdef SQLPTX
        fprintf (outstream, "\n\nNumber of rows retrieved is: %6d",
            rows_fetch);
    else
        fprintf (outstream, "\n\nNumber of rows retrieved is: %6d",
            s_info_ptr->max_rows_fetch);
}
else
    fprintf (outstream, "\n\nNumber of rows retrieved is: %6d",
        rows_fetch);

}

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

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

} /* end of SQLprocess */

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

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

    s_info_ptr = g_struct->s_info_ptr;

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

    if (g_struct->c_l_opt->update > 1)
    {
        /* This is an update function stream. There is no need to COMMIT. */
        /* Each UF child will COMMIT its own transactions. */
        ;
    }
    else
    {
        /* For non-UF cases, COMMIT now. */
        if (g_struct->c_l_opt->a_commit) {
            EXEC SQL COMMIT WORK;
            error_check(); /* @d22275 tlg */
        }
    }

    fflush(outstream);

    s_info_ptr->elapsed_time = get_elapsed_time(start_time);

    if (g_struct->c_flags->time_stamp == TRUE) /* @d25594 tlg */
        get_start_time(&end_t); /* Get the end time */
        strcpy(s_info_ptr->end_stamp,
            get_time_stamp(T_STAMP_FORM_3,&end_t));
        /*get_time_stamp(T_STAMP_FORM_3,(time_t)NULL) */;

    /* BBE: Pass on time stamp values for the next query */
    temp_time_struct = end_t;
    strcpy(temp_time_stamp, s_info_ptr->end_stamp);

    /* write the start timestamp to the file */
    fprintf( outstream, "\n\nStop timestamp %*.s\n",

```

```

        T_STAMP_3LEN,T_STAMP_3LEN, /* TIME_ACC jen*/
        s_info_ptr->end_stamp);

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

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

    s_info_ptr->next->query_block = s_info_ptr->query_block;
    s_info_ptr->next->elapsed_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)
{
#ifdef SQLWINT
    int      semid;          /* semaphore for controlling UFs*/
    key_t    semkey;        /* key to generate semid */
#endif
    char file_name[256] = "\0";

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

    switch (g_struct->c_l_opt->update)
    {
        case (2):
        case (5):
            /* update throughput function stream */
            sprintf(file_name,"%s%spstrcntuf.%s",g_struct->run_dir,

                env_tpcd_path_delim,g_struct->file_time_stamp);
            break;
        case (3):
        case (4):
            /* update power function stream */
            sprintf(file_name,"%s%spstrcntuf.%s",g_struct->run_dir,
                env_tpcd_path_delim,g_struct->file_time_stamp);
            break;
        case (1):
            /* power query stream */
            sprintf(file_name,"%s%spstrcnt%d.%s",g_struct->run_dir,
                env_tpcd_path_delim,
                g_struct->c_l_opt->intStreamNum,g_struct-
                >file_time_stamp);
            break;
        case (0):
            /* throughput query stream */
            sprintf(file_name,"%s%ssstrcnt%d.%s",g_struct->run_dir,
                env_tpcd_path_delim,
                g_struct->c_l_opt->intStreamNum,g_struct-
                >file_time_stamp);
            break;
    }
}

#ifdef LINUX

    if((g_struct->stream_report_file = fopen(file_name, APPENDMODE)) ==
    NULL )
    {
        fprintf(stderr,"\nThe output file for the stream count information\n");
        fprintf(stderr,"could not be opened, make sure the filename is correct\n");
        fprintf(stderr,"filename = %s\n",file_name);
        exit(-1);
    }

#endif

/* print out the stream stop time in the stream count information file*/
if (g_struct->c_l_opt->update > 1)
{
    /* update function stream */
    fprintf(g_struct->stream_report_file,
        "Update function stream stopping at %*.s\n",
        T_STAMP_3LEN,T_STAMP_3LEN, /* TIME_ACC jen*/
        get_time_stamp(T_STAMP_FORM_3,&(g_struct-
        >stream_end_time))); /* TIME_ACC jen*/
}
else
{
    /* query stream(s) */
    fprintf(g_struct->stream_report_file,
        "Stream number %d stopping at %*.s\n",
        g_struct->c_l_opt->intStreamNum,
        T_STAMP_3LEN,T_STAMP_3LEN, /* TIME_ACC jen*/
        get_time_stamp(T_STAMP_FORM_3,&(g_struct-
        >stream_end_time))); /* TIME_ACC jen*/
}
fclose(g_struct->stream_report_file);

/* No need to check for errors here.
   Also, the UF stream in a Throughput run
   has no connection in tpcdbatch.sqc.      aph 98/12/26
error_check();
*/

/* if we are in a query stream AND this is a throughput test, then need */
/* do to some semaphore stuff (0 implies update functions are off) */
/* AND we are supposed to be using semaphores */

```

```

if ( ( semcontrol == 1 ) &&
    ( g_struct->c_l_opt->update < 2))
/* only queries need to release the semaphore at this point */
{
    if (g_struct->c_l_opt->intStreamNum == 0)
        release_semaphore(g_struct, QUERY_POWER_SEM); /* power stream
*/
    else
        release_semaphore(g_struct, THROUGHPUT_SEM); /* throughput
stream */

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

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

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

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

return (TRUE);
}

void create_semaphores(struct global_struct *g_struct)
{
#ifdef SQLWINT
    int      semid; /* semaphore for controlling UFs*/
    key_t    semkey; /* key to generate semid */
#else
    HANDLE    hSem;
    HANDLE    hSem2;
    int      SemTimeout = 600000; /* Des time out period of 1 minute
*/
#endif
    fprintf(stderr, "numstreams = %d\n", g_struct->c_l_opt->intStreamNum);
    fprintf(stderr, "Update stream creating semaphore(s) for update and
query sequencing\n");
#ifdef SQLWINT
    fprintf(stderr, "semfile = %s\n", g_struct->sem_file);
    if (g_struct->c_l_opt->intStreamNum == 0)
        /*RUNPOWER*/
    {
        fprintf(stderr, "semfile2 = %s\n", g_struct->sem_file2);
        hSem = CreateSemaphore(NULL, 0, 1, (LPCTSTR)(g_struct-
>sem_file));
        hSem2 = CreateSemaphore(NULL, 0, 1, (LPCTSTR)(g_struct-
>sem_file2));
        if ((hSem == NULL) || (hSem2 == NULL))
        {
            fprintf(stderr,
                "CreateSemaphores (ready semaphore) failed, GetLastError:
%d, quitting\n",
                GetLastError());
            exit(-1);
        }
    }

```

```

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

        if (hSem == NULL)
        {
            fprintf(stderr,
                "CreateSemaphore (ready semaphore) failed,
GetLastError: %d, quitting\n",
                GetLastError());
            exit(-1);
        }
        fprintf(stderr, "Semaphore created successfully!\n");
    }
}
#else /* AIX, SUN, etc. */
/* create a semaphore key...use the name of a file that */
/* you know exists */
fprintf(stderr, "semfile = %s\n", g_struct->update_num_file);
semkey = ftok(g_struct->update_num_file, 'J');
if (g_struct->c_l_opt->intStreamNum == 0)
/* RUNPOWER */
{
    if ( ( semid =
semget(semkey, 2, IPC_CREAT | S_IRUSR | S_IWUSR)) < 0)
    {
        fprintf(stderr,
            "Throughput can't get initial semaphore! semget failed
errno = %d\n",
            errno);
        exit(1);
    }
}
else
/* THROUGHPUT */
{
    if ( ( semid =
semget(semkey, 1, IPC_CREAT | S_IRUSR | S_IWUSR)) < 0)
    {
        fprintf(stderr,
            "Throughput can't get initial semaphore! semget failed
errno = %d\n",
            errno);
        exit(1);
    }
    if (verbose)
    {
        fprintf(stderr,
            "insert: semkey = %ld, semid = %d, file = %s, value =
%d\n",
            semkey, semid, g_struct->update_num_file,
            (g_struct->c_l_opt->intStreamNum * -1));
    }
}
#endif
}

/*throughput update */
void throughput_wait(struct global_struct *g_struct)
{

```

```

#ifdef SQLWINT
    int      semid;      /* semaphore for controlling UFs*/
    key_t     semkey;     /* key to generate semid */
#else
    HANDLE     hSem;
    int      j;
    int      SemTimeout = 600000; /* Des time out period of 1 minute
*/
#endif

#ifdef SQLWINT
    hSem = open_semaphore(g_struct, THROUGHPUT_SEM);
    for (j = 0; j < g_struct->c_l_opt->intStreamNum; j++)
    {
        if (verbose)
            fprintf(stderr, "About to wait again ...\n");
        if (WaitForSingleObject(hSem, INFINITE) == WAIT_FAILED)
        {
            fprintf(stderr,
                "WaitForSingleObject (hSem) failed on stream %d, error:
%d, quitting\n",
                j, GetLastError());
            exit(-1);
        }
        if (verbose)
            fprintf(stderr, "Streams to wait for %d\n", j);
    }
    fprintf(stderr, "finished waiting on stream semaphore! Ready to run
updates!\n");
    /* close the semaphore handle */
    if (! CloseHandle(hSem)) {
        fprintf(stderr, "Close Sem failed - Last Error: %d\n", GetLastError());
        /* no exit here */
    }
#else
    semid = open_semaphore(g_struct);
    /* call the sem_op routine to decrement the semaphore by */
    /* however many streams .... by calling this function with*/
    /* a negative number, this stream is forced to wait until */
    /* the semaphore gets back to 0 */
    if (sem_op(semid, 0, (g_struct->c_l_opt->intStreamNum * -1)) != 0)
    {
        /*jenSEM*/
        fprintf(stderr,
            "Failure to wait on throughput semaphore for %d streams\n",
            g_struct->c_l_opt->intStreamNum);
        exit(1);
    }
    /*jenSEM*/
    fprintf(stderr, "finished waiting on stream semaphore! Ready to run
updates!\n");
    semctl(semid, 0, IPC_RMID, 0); /* we've finished waiting, now */
    /* remove the semaphore */

#endif
}

void runpower_wait(struct global_struct *g_struct, int sem_num)
{
    char semfile[150];
#ifdef SQLWINT
    HANDLE hSem;

    if (sem_num == 1)
        strcpy(semfile, g_struct->sem_file);
    else
        strcpy(semfile, g_struct->sem_file2);

#else /* AIX */
    int      semid;      /* semaphore for controlling UFs*/
    key_t     semkey;     /* key to generate semid */

```

```

        strcpy(semfile, g_struct->update_num_file);
#endif

    if (g_struct->c_l_opt->update == 1)
        fprintf(stderr, "querystream waiting for update stream (UF1) to signal
semaphore based on %s\n", semfile);
    else
        fprintf(stderr, "updatestream (UF2) waiting on querystream semaphore to
signal semaphore based on %s\n", semfile);

#ifdef SQLWINT
    hSem = open_semaphore(g_struct, sem_num);
    if (verbose)
        fprintf(stderr, "Runpower queries about to wait ...\n");
    if (WaitForSingleObject(hSem, INFINITE) == WAIT_FAILED)
    {
        fprintf(stderr,
            "WaitForSingleObject (hSem) failed on stream 0, error: %d,
quitting\n",
            GetLastError());
        exit(-1);
    }
    if (! CloseHandle(hSem))
    {
        fprintf(stderr, "Close Sem failed - Last Error: %d\n",
            GetLastError());
        /* no exit here */
    }
#else
    semid = open_semaphore(g_struct);

    /* call the sem_op routine to decrement the semaphore by */
    /* however many streams .... by calling this function with*/
    /* a negative number, this stream is forced to wait until */
    /* the semaphore gets back to 0 */
    /* aix semaphores start at 0, not 1, so sem_num - 1 is used */
    if (sem_op(semid, sem_num - 1, -1) != 0)
    {
        /*jenSEM*/
        fprintf(stderr,
            "Failure to wait on runpower semaphore for %d streams\n",
            g_struct->c_l_opt->intStreamNum);
        exit(1);
    }
    /*jenSEM*/
#endif

    if (g_struct->c_l_opt->update == 1)
        fprintf(stderr, "querystream finished waiting on updatestream
semaphore\n");
    else
        fprintf(stderr, "updatestream finished waiting on querystream
semaphore\n");
}

void release_semaphore(struct global_struct *g_struct, int sem_num)
{
#ifdef SQLWINT
    int      semid;      /* semaphore for controlling UFs*/
    key_t     semkey;     /* key to generate semid */
#else
    HANDLE     hSem;
    int      SemTimeout = 600000; /* Des time out period of 1 minute
*/
#endif

#ifdef SQLWINT
    hSem = open_semaphore(g_struct, sem_num); /* query */

```

```

    if (! ReleaseSemaphore(hSem,
        1,
        (LPLONG)(NULL)))
    {
        fprintf(stderr, "ReleaseSemaphore failed, Sem#: %d LastError: %d,
quit\n",
            sem_num, GetLastError());
        exit(-1);
    }
#else
    semid = open_semaphore(g_struct); /* query */
    /* aix semaphores start at 0, not 1, so sem_num -1 is used */
    if (sem_op(semid, sem_num - 1, 1) != 0) /*jenSEM*/
    {
        /*jenSEM*/
        fprintf(stderr,
            "Failed to increment semaphore %d for throughput stream
%d\n",
            sem_num, g_struct->c_l_opt->intStreamNum);
        fprintf(stderr,
            "file for generation of semaphore is: %s\n",
            g_struct->update_num_file);
        exit(1);
    }
#endif
    if (g_struct->c_l_opt->intStreamNum == 0)
    { /* RUNPOWER */
        if (sem_num == 1)
        {
            fprintf(stderr, "UF1 completed.\n");
        }
        else
        {
            fprintf(stderr, "query stream completed.\n");
        }
    }
}

#ifdef SQLWINT /* Compile only in NT */
HANDLE open_semaphore(struct global_struct *g_struct, int num)
{
    HANDLE hSem;
    LPCTSTR semfile;

    if (num == 1)
        semfile = (LPCTSTR)g_struct->sem_file;
    else
        semfile = (LPCTSTR)g_struct->sem_file2;

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

        Sleep(1000);
    }
    return hSem;
}

#else /* Compile only in non-NT (i.e. AIX) */
int open_semaphore(struct global_struct *g_struct)
{
    int semid; /* semaphore for controlling UFs*/

```

```

    key_t semkey; /* key to generate semid */
    int num;

    if (g_struct->c_l_opt->intStreamNum == 0)
        num = 2;
    else
        num = 1;

    semkey = ftok(g_struct->update_num_file, 'J');
    while ((semid = semget(semkey, num, 0)) < 0)
    {
        if (errno == ENOENT)
        {
            sleep(2);
            fprintf(stderr, "cleanUp: looping for access to semaphore
stream %d ",
                g_struct->c_l_opt->intStreamNum);
            fprintf(stderr, "semkey=%ld semid = %d
file=%s\n", semkey, semid,
                g_struct->update_num_file);
        }
        else
        {
            fprintf(stderr, "query stream %d semget failed errno = %d\n",
                g_struct->c_l_opt->intStreamNum, errno);
            exit(1);
        }
    }
    return semid;
}
#endif

```

D.3 tpcdUF.sqc

```

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

#define UF1DEBUG
#define UF2DEBUG

#ifdef SQLPTX && defined(SQLSUN)
#define exit(rc) _exit(rc)
#else
#define exit(rc) exit(rc)
#endif /* SQLPTX & SQLSUN*/

#include "tpcdbatch.h"

```

```

/** EXEC SQL INCLUDE SQLCA; */

#include "sqlca.h"
extern struct sqlca sqlca;

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

/*****
*****/
/* Declare the SQL host variables. */
/*****
*****/
EXEC SQL BEGIN DECLARE SECTION;
char UF_dbname[9] = "\0";
char UF_userid[9] = "\0";
char UF_passwd[9] = "\0";
sqlint32 UF_chunk = 0;
short month = 0;
EXEC SQL END DECLARE SECTION;

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

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

    long sqlcode;
    int maxwait;

```

```

#ifndef SQLWINT
    int su_sem;
    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(myostream,"Before loop_a startChunk = %d, stopChunk =
%d\n", startChunk, stopChunk);
    fflush(myostream);
#endif
    for ( UF_chunk = startChunk; UF_chunk <= stopChunk; UF_chunk++ )
/*jenCI*/
    {
        /* wlc 062797 */
        sqlcode = SQL_RC_E911;
        month = (short)UF_chunk; /* Cast 'short' added bbe */
        maxwait = 1;
        rc = 0;

#ifdef UF1DEBUG
        fprintf(myostream, "Before While_a Chunk= %d \n",UF_chunk);
        fflush(myostream);
#endif
        /* Loop to handle any deadlocks */
        while (sqlcode == SQL_RC_E911 && maxwait <= MAXWAIT &&
rc==0)
        {
#ifdef UF1DEBUG
            fprintf(myostream, "in loop before orders exec sql %d\n", sqlcode);
            fflush(myostream);
#endif
            sqlcode = 0;
            EXEC SQL INSERT INTO TPCD.ORDERS
            SELECT
            O_ORDERKEY,O_CUSTKEY,O_ORDERSTATUS,O_TOTALPRICE,
            O_ORDERDATE,O_ORDERPRIORITY,O_CLERK,O_SHIPPRIORITY,O
_COMMENT
            FROM TPCDTEMP.ORDERS_NEW
            WHERE APP_ID = :UF_chunk;
            /*AND 12*(YEAR(O_ORDERDATE)-
1992)+MONTH(O_ORDERDATE)-01 = :month;*/

            fprintf(myostream,
            "Insert into orders pair %d chunk %d Return sqlcode=%d\n",
            updatePair,UF_chunk,sqlca.sqlcode);

            if (sqlca.sqlcode < 0)
                sqlcode = error_check();

            if (sqlcode == SQL_RC_E911)
            {
                /* we've hit a deadlock */
                fprintf(myostream,
                "\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(myostream,
                "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(myostream, "in lineitem for update pair number %d set %d
chunk %d\n",
                    updatePair, i,UF_chunk);

```

```

                fflush(myostream);
            }
#endif

            EXEC SQL INSERT INTO TPCD.LINEITEM
            SELECT
            L_ORDERKEY,L_PARTKEY,L_SUPPKEY,L_LINENUMBER,L_QUAN
TITY,
            L_EXTENDEDPRIE,L_DISCOUNT,L_TAX,
            L_RETURNFLAG,L_LINESTATUS,L_SHIPDATE,L_COMMITDATE,L_
RECEIPTDATE,
            L_SHIPINSTRUCT,L_SHIPMODE,L_COMMENT
            FROM TPCDTEMP.LINEITEM_NEW WHERE APP_ID =
:UF_chunk;
            /*(AND L_ORDERKEY IN
            (SELECT O_ORDERKEY FROM TPCD.ORDERS
            WHERE 12*(YEAR(O_ORDERDATE)-
1992)+MONTH(O_ORDERDATE)-01 = :month);*/

            fprintf(myostream,
            "Insert into Lineitem pair %d chunk %d Return
sqlcode=%d\n",
            updatePair,UF_chunk,sqlca.sqlcode);

            if (sqlca.sqlcode < 0)
                sqlcode = error_check();

            if (sqlcode == SQL_RC_E911)
            {
                /* we've hit a deadlock */
                fprintf(myostream,
                "\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(myostream,
                "Insert into lineitem pair %d chunk %d failed sqlcode=%d\n",
                updatePair,UF_chunk,sqlcode);
                dumpCa(&sqlca);
                rc = -1;
            }
            else
            {
#ifdef UF1DEBUG
                fprintf(myostream, "lineitem insert succeeded\n");
                fflush(myostream);
#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(myostream, " interim %ld rows for chunk %d into
TPCD.ORDERS at %*.*s\n",
                    insertedOrders,UF_chunk,T_STAMP_1LEN,T_STAMP_1LEN, /*
TIME_ACC jen*/
                    get_time_stamp(T_STAMP_FORM_1,(Timer_struct
*)NULL)); /* TIME_ACC jen*/
                /* report the number of row deleted *s inserted */
                fprintf(myostream,

```

```

        " 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( myostream,
        " 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(myostream, "%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(myostream, "%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( myostream, "# of deadlocks exceeds %i\n", MAXWAIT);
        }
        rc=-1;
        EXEC SQL ROLLBACK WORK;
        error_check(); /* @d22275 tlg */

        goto UF1_exit;
    }

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

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

/* we used the first flat file to generate the semaphore key */

#ifdef SQLWINT
/* we will use the tpcd.setup file to generate the semaphore key begin
SEMA */
if (getenv("TPCD_AUDIT_DIR") != NULL)
{
    /* this is assuming that you will be running this from 0th node */
    sprintf(sourcefile, "%s%ctools%ctpcd.setup",
        getenv("TPCD_AUDIT_DIR"), PATH_DELIM,PATH_DELIM);
}
else
{

```

```

    fprintf( stderr, "Can't open UF1 semaphore file TPCD_AUDIT_DIR is
not defined.\n");
    exit (-1);
}
/* end SEMA */

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

#else /* SQLWINT */
sprintf( UF1_semaphore, "%s.%s.UF1.semfile",
    getenv("TPCD_DBNAME"), getenv("USER"));
fprintf(stderr,"UF1 semfile = %s\n",UF1_semaphore);
while ((su_hSem = OpenSemaphore(SEMAPHORE_ALL_ACCESS |
    SEMAPHORE_MODIFY_STATE |
    SYNCHRONIZE,
    TRUE,
    UF1_semaphore))
    == (HANDLE)(NULL))
{
    /*
    ** if cannot open the semaphore, wait for 0.1 second
    */
    fprintf(stderr,"Retry Open semaphore %s\n", UF1_semaphore);

    sleep(1);
}

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

/*****
*****/
/* UF2 child */
/*****
*****/
void runUF2_fn ( int updatePair, int thisConcurrentDelete, int numChunks,
char *dbname, char *userid, char *passwd )
{
    int rc = 0;
    long sqlcode;
    int maxwait;

```

```

    int startChunk = thisConcurrentDelete*numChunks; /* where do we start?
*/
    long deletedLineitems = 0; /*kmw*/
    long deletedOrders = 0; /*kmw*/
    long savedDeletedLineitems = 0; /*kbs*/

#ifdef 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 myostreamfile[256];
    FILE *myostream, *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(myostreamfile, UF2OUTSTREAMPATTERN,
env_tpcd_tmp_dir, PATH_DELIM,
        updatePair, thisConcurrentDelete);
    if ( (myostream = fopen(myostreamfile, WRITEMODE)) == NULL)
    {
        fprintf(stderr,
            "\nThe output file '%s' for update pair %d set %d could not be
opened runUF2_fn.\n",
            myostreamfile, updatePair, thisConcurrentDelete);
        rc=-1;
        goto UF2_exit;
    }

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

#ifdef UF2DEBUG
    fprintf(myostream, "RunUF2 Called %d %d %d\n",
        updatePair, thisConcurrentDelete, numChunks );
    fflush(myostream);
#endif
    fprintf( myostream,
        "\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(myostream, "before connect\n");
    fflush(myostream);
#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(myostream, "after connect startchunk= %d, EndChunk = %d\n",
        startChunk, startChunk+numChunks);
    fflush(myostream);
#endif

    /* Start processing each chunk in my range */

```

```

    for ( UF_chunk = startChunk; UF_chunk < startChunk+numChunks;
UF_chunk++)
    {

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

#ifdef UF2DEBUG
        fprintf(myostream, "Chunk = %d\n", UF_chunk);
        fflush(myostream);
#endif
        while (sqlcode == SQL_RC_E911 && maxwait <= MAXWAIT && rc
== 0)
        {

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

            EXEC SQL DELETE FROM TPCD.LINEITEM
                WHERE L_ORDERKEY IN
                (SELECT O_ORDERKEY FROM
TPCDTEMP.ORDERS_DEL
                WHERE APP_ID = :UF_chunk);
            /*AND O_ORDERKEY IN
                (SELECT O_ORDERKEY FROM TPCD.ORDERS
                WHERE 12*(YEAR(O_ORDERDATE)-
1992)+MONTH(O_ORDERDATE)-01 = :month));*/
            if (sqlca.sqlcode < 0)
                sqlcode = error_check();

            if (sqlcode == SQL_RC_E911)
            {
                /* we've hit a deadlock */
                fprintf(myostream,
                    "\nA deadlock detected while deleting from LINEITEM: update
pair %d set %d chunk %d. Retrying.\n",
                    updatePair, thisConcurrentDelete, UF_chunk);
                dumpCa(&sqlca);
                SleepSome(UF_DEADLOCK_SLEEP);
                maxwait++; /* jen DEADLOCK */
            }
            else if (sqlcode < 0)
            {
                fprintf(myostream, "\n%s\n", stmt_str.data);
                fprintf(myostream, "\nsqlcode %d occurred deleting from
TPCD.LINEITEM\n", sqlca.sqlcode);
                dumpCa(&sqlca);
                fprintf(myostream,
                    "for update pair number %d set %d chunk %d..Exiting\n",
                    updatePair, thisConcurrentDelete, UF_chunk);
                rc=-1;
            }
            else
            {
                /* accumulate the number of row deleted */
                savedDeletedLineitems = sqlca.sqlerrd[2]; /*kbs*/

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

                /* delete the orders now */

```

```

EXEC SQL DELETE FROM TPCD.ORDERS
WHERE O_ORDERKEY IN
(SELECT O_ORDERKEY FROM TPCDTEMP.ORDERS_DEL
WHERE APP_ID = :UF_chunk);
/*AND 12*(YEAR(O_ORDERDATE)-
1992)+MONTH(O_ORDERDATE)-01 = :month;*/

if (sqlca.sqlcode < 0)
    sqlcode = error_check();

if (sqlcode == SQL_RC_E911)
{
    /* we've hit a deadlock */
#ifdef UF2DEBUG
    fprintf(myoutstream, "orders deadlocked\n");
    fflush(myoutstream);
#endif
    fprintf(myoutstream,
        "\nA deadlock detected while deleting from ORDERS: update pair
%d set %d chunk %d. Retrying.\n",
        updatePair, thisConcurrentDelete, UF_chunk);
    dumpCa(&sqlca);
    SleepSome(UF_DEADLOCK_SLEEP);
    maxwait++; /* jen DEADLOCK */
}
else if (sqlcode < 0)
{
#ifdef UF2DEBUG
    fprintf(myoutstream, "orders failed\n");
    fflush(myoutstream);
#endif
    fprintf(myoutstream, "\nAn error %d occurred deleting from
TPCD.ORDERS\n", sqlca.sqlcode);
    dumpCa(&sqlca);
    fprintf(myoutstream, "for update pair number %d set %d chunk
%d..Exiting\n",
        updatePair, thisConcurrentDelete, UF_chunk);
    rc=-1;
}
else
{
#ifdef UF2DEBUG
    fprintf(myoutstream, "orders succeeded\n");
    fflush(myoutstream);
#endif
    /* accumulate the number of row deleted */
    /* Order count ONLY updated if both orders and lineitem */
    /* go through */
    deletedLineitems += savedDeletedLineitems; /* kbs */
    deletedOrders += sqlca.sqlerrd[2];
    rc=0;
    EXEC SQL COMMIT WORK;
    error_check();
#ifdef UF2DEBUG
    /* report the number of rows deleted */
    fprintf(myoutstream, "    interim %ld rows for chunk %d from
TPCD.ORDERS at %*.s\n",

deletedOrders, UF_chunk, T_STAMP_1LEN, T_STAMP_1LEN, /*
TIME_ACC jen*/
        get_time_stamp(T_STAMP_FORM_1, (Timer_struct
*)NULL)); /* TIME_ACC jen*/
    fprintf(myoutstream, "    interim %ld rows for chunk %d from
TPCD.LINEITEM at %*.s\n",

deletedLineitems, UF_chunk, T_STAMP_1LEN, T_STAMP_1LEN, /*
TIME_ACC jen*/
        get_time_stamp(T_STAMP_FORM_1, (Timer_struct
*)NULL)); /* TIME_ACC jen*/
    fprintf(myoutstream,

```

```

"    deletes for update pair %d chunk %d complete at
%*.s\n\n",
        updatePair, UF_chunk,
        T_STAMP_1LEN, T_STAMP_1LEN, /* TIME_ACC jen*/
        get_time_stamp(T_STAMP_FORM_1,
            (Timer_struct *)NULL)); /* TIME_ACC jen*/
#endif
    }
    /* process orders deletes */
    /* while trying to delete one chunk loop */
    /* while there are more chunks */

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

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

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

UF2_exit:
    fclose(myoutstream);

    /* exiting, increment the semaphore */
#ifdef SQLWINT
    /* we used the tpcd.setup file to generate the semaphore key begin
SEMA */
    if (getenv("TPCD_AUDIT_DIR") != NULL)
    {
        sprintf(sourcefile, "%s%ctools%ctpcd.setup",
            getenv("TPCD_AUDIT_DIR"), PATH_DELIM, PATH_DELIM);
    }
    else
    {
        fprintf(stderr, "Can't open UF2 semaphore file TPCD_AUDIT_DIR is
not defined.\n");
        exit(-1);
    }

    su_semkey = ftok(sourcefile, 'D'); /* use D for deletes */
    /* end SEMA */
    while ((su_semid = semget(su_semkey, 1, 0)) < 0)
    {
        if (errno == EWOULDBLOCK)
            sleep(2);
        else {
            fprintf(stderr, "UF2 update stream %d: semget failed errno = %d\n",

```

```

        updatePair, errno);
    exit(1);
}
}
if (sem_op (su_sem, 0, 1) != 0) /*jenSEM*/
{
    /*jenSEM*/
    fprintf(stderr, "Failure to increment semaphore 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 */
}

```

D.4 runpower

```

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

# usage runpower [UF]
# where UF is the optional parameter that says to run the power test
# with the update functions. By default, the update functions are not
# run

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

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

# Use the macros in here so that they can handle the platform differences.
# macro.pl should be sourced from cmvc, other people wrote and maintain it.
require "macro.pl";
require "tpcdmacro.pl";

# Make output unbuffered.
select(STDOUT);
$| = 1 ;

if (@ARGV > 0)
{

```

```

    $runUF=$ARGV[0];
}
else
{
    $runUF="no";
}

if (length($ENV{"TPCD_AUDIT_DIR"}) <= 0)
{
    die "TPCD_AUDIT_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_RUN_DIR"}) <= 0)
{
    die "TPCD_RUN_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_DBNAME"}) <= 0)
{
    die "TPCD_DBNAME environment variable not set\n";
}
if (length($ENV{"TPCD_RUNNUMBER"}) <= 0)
{
    die "TPCD_RUNNUMBER environment variable not set\n";
}
if (length($ENV{"TPCD_SF"}) <= 0)
{
    die "TPCD_SF environment variable not set\n";
}
if (length($ENV{"TPCD_PLATFORM"}) <= 0)
{
    die "TPCD_PLATFORM environment variable not set\n";
}
if (length($ENV{"TPCD_PATH_DELIM"}) <= 0)
{
    die "TPCD_PATH_DELIM environment variable not set\n";
}
if (length($ENV{"TPCD_PRODUCT"}) <= 0)
{
    die "TPCD_PRODUCT environment variable not set\n";
}
if (length($ENV{"TPCD_AUDIT"}) <= 0)
{
    die "Must set TPCD_AUDIT env't var. Real audit timing sequence run if
yes\n";
}
if (length($ENV{"TPCD_PHYS_NODE"}) <= 0)
{
    die "TPCD_PHYS_NODE env't var not set\n";
}
if (length($ENV{"TPCD_LOG_DIR"}) <= 0)
{
    $ENV{"TPCD_LOG_DIR"} = "NULL";
}
if (length($ENV{"TPCD_MODE"}) <= 0)
{
    die "TPCD_MODE environment variable not set - uni/smp/mln \n";
}
if (length($ENV{"TPCD_ROOTPRIV"}) <= 0)
{
    die "TPCD_ROOTPRIV environment variable not set - yes/no \n";
}

#set up local variables
$runNum=$ENV{"TPCD_RUNNUMBER"};
$runDir=$ENV{"TPCD_RUN_DIR"};
$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$dbname=$ENV{"TPCD_DBNAME"};
$sf=$ENV{"TPCD_SF"};
$platform=$ENV{"TPCD_PLATFORM"};
$delim=$ENV{"TPCD_PATH_DELIM"};
$gatherstats=$ENV{"TPCD_GATHER_STATS"};

```

```

$product=$ENV{"TPCD_PRODUCT"};
$RealAudit=$ENV{"TPCD_AUDIT"};
$inlistmax=$ENV{"TPCD_INLISTMAX"};
$pn=$ENV{"TPCD_PHYS_NODE"};
$logDir=$ENV{"TPCD_LOG_DIR"};
$rootPriv=$ENV{"TPCD_ROOTPRIV"};
$mode=$ENV{"TPCD_MODE"};
if (( $mode eq "uni" ) || ( $mode eq "smp" ))
{
    $all_ln="once";
    $all_pn="once";
    $once="once";
}
else
{
    $all_ln="all_ln";
    $all_pn="all_pn";
    $once="once";
}

if ($inlistmax eq "default")
{
    $inlistmax = 400;
}

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

# append isolation level information about tpcdbatch to the miso file
# the miso file is created here but appended to for power and throughput
#information

$misofile="$RunDir${delim}miso$RunNum";
if ( -e $misofile )
{
    &rm("$misofile");
}
# if we are in real audit mode then we must start the db manager now since
# there must be no activity on the database between the time the build script
# has finished and the time the power test is started
if ( $RealAudit eq "yes" )
{
    system("db2start");
    system("db2 activate database $dbname");
}

# do not activate the database
#if ( $RealAudit ne "yes" )
#{
#    system("db2 activate database $dbname");
#}

#Report current log info to the run# directory in a file called startLog.Info
system("perl getLogInfo.pl startLog");

open(MISO, ">$misofile") || die "Can't open $misofile: $!\n";
$curTs = `perl gettimestamp "long"`;
print MISO "Timestamp and isolation level of tpcdbatch before power run at
: $curTs\n";
close(MISO);
if ( $product eq "pe" )
{
    system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_id,isolation from sysibm.sysplan where name like
'TPCD%\"; db2 connect reset; db2 terminate >>
$RunDir${delim}miso$RunNum ");
}
else
{
    &verifyTPCDBatch("$misofile","$dbname");
}

if ($platform eq "aix")
{
    # Create the sysunused file. This reports what disks are attached, and which
    # ones are being used. Its use spans both the runpower and runthroughput
    tests
    system("echo \"The following disks are assigned to the indicated volume
groups\" > $RunDir/sysunused$RunNum") && die "cannot create
$RunDir/sysunused$RunNum";

    system("lspv >> $RunDir/sysunused$RunNum");
    system("echo \"The following volume groups are currently online\" >>
$RunDir/sysunused$RunNum");
    $curTs = `perl gettimestamp "long"`;
    system("echo \"$curTs\" >> $RunDir/sysunused$RunNum");
    system("lsvg -o >> $RunDir/sysunused$RunNum");
    # show the disks that are used/unused
    system("getdisks \"Before the start of the Power Test\"");
}
else
{
    # for all other platforms
    system("echo Assume that all portions of the system are used >>
$RunDir${delim}sysunused$RunNum");
}

&getConfig("p");
if ( $rootPriv eq "yes" )
{
    # get the o/s tuning parameters...currently AIX only and only if your
    # user has root privileges to run this
    &getOSTune("p");
}
if ($gatherstats eq "on")
{
    # gather vm io and net stats
    if ($platform eq "aix" || $platform eq "sun" || $platform eq "ptx" ||
        $platform eq "hp" || $platform eq "linux")
    {
        # gather vmstats and iostats (and net stats if in mpp mode)
        system("perl getstats p &");
    }
    else
    {
        print "Stats gather not set up for current platform $platform\n";
    }
}

# print to screen what type of run is running and set variables to run
# the query and update streams in parallel
if ($runUF ne "UF")
{
    $semcontrol = "off";
    print "Beginning power stream....no update functions\n";

    $streamEx = "";
    $streamExNT = "";
}
else
{
    $semcontrol = "on";
    print "Beginning power stream....with update functions\n";
    if ( $platform eq "nt" )

```

```

{
    $streamExNT = "start ";
    $streamEx = "";
}
else
{
    $streamExNT = "";
    $streamEx = "&";
}
}

# bbe This new line (below) runs queries for power test

print "Starting tpcdbatch...\n";
$ret=system("$streamExNT $auditDir${delim}auditruns${delim}tpcdbatch -
d $dbname -f $runDir${delim}qtextpow.sql -r on -b on -s $sf -u p1 -m
$inlistmax -n 0 -l
$auditDir${delim}auditruns${delim}querytext${delim}streampow.list -p
$semcontrol $streamEx");

if ( $runUF eq "UF" )
{
    $ret2 = system("$auditDir${delim}auditruns${delim}tpcdbatch -d
$dbname -f $runDir${delim}qtextqf.sql -r on -b on -s $sf -u p2 -m
$inlistmax -n 0 -l
$auditDir${delim}auditruns${delim}querytext${delim}streampuf.list");
}
else
{
    $ret2 = 0; # If UFs were not running, then the stream cannot fail
}

if (($ret2 == 0) && ($ret == 0))
{
    print "Power stream completed succesfully.\n";
}
else
{
    print "Power stream failed. ret=$ret\n";
}

if ($platform eq "aix")
{
    # show that the same disks are still used or unused
    system("getdisks \"After completion of the Power Test\"");

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

```

```

    }
    $rc= `perl5 zap "-f" "getstats"`;
}
}

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

if ( $product eq "pe" )
{
    system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_id,isolation from sysibm.sysplan where name like
'TPCD%\"";db2 connect reset;db2 terminate >>
$runDir${delim}miso$runNum");
}
else
{
    &verifyTPCDBatch("$misofile", "$dbname");
}
if ( $RealAudit ne "yes" )
{
    $curTs = `perl gettimestamp "short"`;
    # grab the db and dbm snapshot before we deactivate
    system("db2 get snapshot for all on $dbname >
$runDir${delim}dbrun$runNum.snap.$curTs");
    system("db2 get snapshot for database manager >>
$runDir${delim}dbrun$runNum.snap.$curTs");
}

#####

# now copy the reports from the count of streams files into one final file
&cat("$runDir${delim}pstrcnt*", "$runDir${delim}mpstrcnt$runNum");
#(NOTE: there is a dependancy that this mpstrcnt file exist before the
# calcmetrics.pl script is called, both because it is used as input for
# calcmetrics.pl, and because the output from calcmetrics is used as
# the trigger for watchstreams to complete, and watchstreams cats its
# output at the end of the mstrcnt file.

# generate the mpinter?.metrics file in the run directory
#require 'calcmetricsp.pl';
if ( $runUF eq "UF" )
{
    system("perl calcmetricsp.pl UF");
}
else
{
    system("perl calcmetricsp.pl");
}

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

if ($runUF eq "UF") {

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

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

#####

```

```

# no longer activate/deactivate the database
# if ( $RealAudit ne "yes" )
#{
#   # deactivate the database
#   system("db2 deactivate database $dbname");
#}

# do not stop the database after the power test
# if ( $RealAudit ne "yes" )
#{
#   system("db2stop");
#}

1;

sub getConfig
{
    $testtype=$_[0];
    print "Getting database configuration.\n";
    $dbtunefile="$runDir${delim}m${testtype}dbtune${runNum}";
    open(DBTUNE, ">$dbtunefile") || die "Can't open $dbtunefile: $!\n";
    $timestamp=`perl gettimestamp "long"`;
    print DBTUNE "Database and Database manager configuration taken at : $timestamp";
    close(DBTUNE);
    system("db2level >> $dbtunefile");
    system("db2 get database configuration for $dbname >> $dbtunefile");
    system("db2 get database manager configuration >> $dbtunefile");
    system("db2set >> $dbtunefile");
    if (( $mode eq "mln" ) || ( $mode eq "mpp" ))
    {
        $cfgfile="$runDir${delim}dbtune${runNum}.";
        #removed by Alex due to hang
        #system("db2_all `||` typeset -i ln=##; db2 get db cfg for $dbname > $cfgfile${ln}; db2 get dbm cfg >> $cfgfile${ln}; db2set >> $cfgfile${ln}; db2 terminate """);
    }
}

sub getOSTune
{
    $testtype=$_[0];
    if ( $platform eq "aix" )
    {
        print "Getting OS and VMdatabase configuration.\n";
        $ostunefile="$runDir${delim}m${testtype}ostune${runNum}";
        open(OSTUNE, ">$ostunefile") || die "Can't open $ostunefile: $!\n";
        $timestamp=`perl gettimestamp "long"`;
        print OSTUNE "Operating System and Virtual Memory configuration taken at : $timestamp";
        close(OSTUNE);
        system("${delim}usr${delim}samples${delim}kernel${delim}schedtune >> $ostunefile");
        system("${delim}usr${delim}samples${delim}kernel${delim}vmtune >> $ostunefile");
    }
    else
    {
        print "OS parameters retrieval not supported for $platform \n";
    }
}

sub verifyTPCDBatch
{
    $logfile=$_[0];
    $dbname=$_[1];
    $file="verifytpcdbatch.clp";

```

```

    open(VERTBL, ">$file") || die "Can't open $file: $!\n";
    print VERTBL "connect to $dbname;\n";
    print VERTBL "select name,creator,valid,last_bind_time,isolation from sysibm.sysplan where name like 'TPCD%';\n";
    print VERTBL "connect reset;\n";
    print VERTBL "terminate;\n";
    close(VERTBL);
    system("db2 -vtf $file >> $logfile");
}

```

D.5 runthroughput

```

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

# usage runthroughput [UF]
# where UF is the optional parameter that says to run the throughput test
# with the update functions. By default, the update functions are not
# run
# If UF is not supplied and a number is supplied, then that number is taken
# as the number of concurrent throughput streams to run. This is also
# optional

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

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

# Use the macros in here so that they can handle the platform differences.
# macro.pl should be sourced from cmvc, other people wrote and maintain it.
require "macro.pl";
require "tpcdmacro.pl";

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

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

```

```

die "TPCD_RUN_ON_MULTIPLE_NODES environment variable not
set\n";
}
if (length($ENV{"TPCD_SF"}) <= 0)
{
die "TPCD_SF environment variable not set\n";
}
if (length($ENV{"TPCD_PRODUCT"}) <= 0)
{
die "TPCD_PRODUCT environment variable not set\n";
}
if (length($ENV{"TPCD_PLATFORM"}) <= 0)
{
die "TPCD_PLATFORM environment variable not set\n";
}
if (length($ENV{"TPCD_AUDIT"}) <= 0)
{
die "Must set TPCD_AUDIT env't var. Real audit timing sequence run if
yes\n";
}
if (length($ENV{"TPCD_LOG_DIR"}) <= 0)
{
$ENV{"TPCD_LOG_DIR"} = "NULL";
}
if (length($ENV{"TPCD_MODE"}) <= 0)
{
die "TPCD_MODE environment variable not set - uni/smp/mln \n";
}
if (length($ENV{"TPCD_ROOTPRIV"}) <= 0)
{
die "TPCD_ROOTPRIV environment variable not set - yes/no \n";
}

#set up local variables
$runNum=$ENV{"TPCD_RUNNUMBER"};
$numStream=$ENV{"TPCD_NUMSTREAM"};
$runDir=$ENV{"TPCD_RUN_DIR"};
$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$dbname=$ENV{"TPCD_DBNAME"};
$sf=$ENV{"TPCD_SF"};
$product=$ENV{"TPCD_PRODUCT"};
$multinode=$ENV{"TPCD_RUN_ON_MULTIPLE_NODES"};
$platform=$ENV{"TPCD_PLATFORM"};
$delim=$ENV{"TPCD_PATH_DELIM"};
$RealAudit=$ENV{"TPCD_AUDIT"};
$inlistmax=$ENV{"TPCD_INLISTMAX"};
$gatherstats=$ENV{"TPCD_GATHER_STATS"};
$logDir=$ENV{"TPCD_LOG_DIR"};
$rootPriv=$ENV{"TPCD_ROOTPRIV"};
$mode=$ENV{"TPCD_MODE"};

$path="$auditDir${delim}auditruns";

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

# return 1 if the given pattern(parameter $_[0]) matches any file
sub existfile {
if ($platform eq "aix" || $platform eq "sun" || $platform eq "ptx" || $platform
eq "linux")
{
ls $_[0] 2> /dev/null | wc -l` + 0 != 0;
}
else
{
dir /b $_[0] 2> NUL | wc -l` + 0 != 0;
}
}

if ($inlistmax eq "default")
{
$inlistmax = 400;
}

# no longer stop and start the dbm between runs when not in realaudit mode
#if ( $RealAudit ne "yes" )
#{
# # if we are not in real audit mode then we must start the db manager now
# system("db2start");
# # activate the database
# system("db2 activate database $dbname");
#}

$misofile="$runDir${delim}miso$runNum";
# append isolation level information about tpcdbatch to the miso file
open(MISO, ">>$misofile") || die "Can't open $misofile: $!\n";
$curTs = `perl gettimestamp "long"`;
print MISO "Timestamp and isolation level of tpcdbatch before throughput
run at : $curTs\n";
close(MISO);

if ( $product eq "pe" )
{
system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_id,isolation from sysibm.sysplan where name like
'TPCD%'\n >> $runDir${delim}miso$runNum ");
}
else
{
&verifyTPCdbatch("$misofile","$dbname");
}

# kick off the script that will monitor for the database applications during
# the running of the throughput tests. This will quit when the
mtinterX.metrics
# (where X=runnumber) file has been created.

# set variables to run streams in parallel
if ( $platform eq "nt" )
{
$streamExNT = "start /b";
$streamEx = "";
}
else
{
$streamExNT = "";
$streamEx = "&";
}

if ( $platform eq "aix" || $platform eq "sun" || $platform eq "nt" || $platform
eq "hp" || $platform eq "linux")
{
system("$streamExNT perl watchstreams $streamEx");
}
else
{
die "platform not supported, can't start watchstreams in background";
}

# show the disks that are used/unused
if ($platform eq "aix")

```

```

{
    system("getdisks \"Before the start of the Throughput Test\");
}
if ($gatherstats eq "on")
{
    # gather vm io and net stats
    if ($platform eq "aix" || $platform eq "sun" || $platform eq "ptx" ||
    $platform eq "hp" || $platform eq "linux")
    {
        # gather vmstats and iostats (and net stats if in mpp mode)
        system("perl getstats t &");
    }
    else
    {
        print "Stats gather not set up for current platform $platform\n";
    }
}

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

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

    $loopStream=1;

    for ( $loopStream = 1; $loopStream <= $numStream; $loopStream++)
    {
        print "starting stream $loopStream\n";
        system("echo Executing stream $loopStream out of $numStream.");
        # run the queries
        if ( $platform eq "aix" || $platform eq "sun" || $platform eq "nt" ||
        $platform eq "ptx" ||
        $platform eq "hp" || $platform eq "linux")
        {
            system("$streamExNT $path${delim}tpcdbatch -d $dbname -f
$runDir${delim}qtextt$loopStream.sql -r on -b on -s $sf -u t1 -m $inlistmax
-n $loopStream -l $path${delim}querytext${delim}stream$loopStream.list
$streamEx");
        }
        else
        {
            die "platform $platform not supported yet";
        }
    }

    # run the update function stream....this will wait until the queries have
    # completed to kick off the updates
    print "starting update stream\n";

    if ($runUF eq "no") {
        $ret=system("$auditDir${delim}auditruns${delim}tpcdbatch -d $dbname
-f $runDir${delim}quft.sql -r on -b on -s $sf -u t -m $inlistmax -n
$numStream -l $runDir${delim}streamuf.list");
    }
    else {
        $ret=system("$auditDir${delim}auditruns${delim}tpcdbatch -d $dbname
-f $runDir${delim}quft.sql -r on -b on -s $sf -u t2 -m $inlistmax -n
$numStream -l $runDir${delim}streamuf.list");
    }
    print "update stream done\n";

    &getConfig("t");
    if ( $rootPriv eq "yes" )
    {
        # get the o/s tuning parameters...currently AIX only and only if your
        # user has root privileges to run this
        &getOSTune("t");
    }
}
else
{
    # we are running the query streams and update stream from different nodes,
    use
    # files and rksh to control the update stream
    system("runthru.pe");
}

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

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

if ( $product eq "pe" )
{
    system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_id,isolation from sysibm.sysplan where name like
'TPCD%\" >> $runDir${delim}miso$runNum\"");
}
else
{
    &verifyTPCDBatch("$misofile", "$dbname");
}

if ( $RealAudit ne "yes" )
{
    $curTs = `perl gettimestamp "short"`;
    # grab the db and dbm snapshot before we deactivate
    system("db2 get snapshot for all on $dbname >
$runDir${delim}dbTrun$runNum.snap.$curTs");
    system("db2 get snapshot for database manager >>
$runDir${delim}dbTrun$runNum.snap.$curTs");
}

# now copy the reports from the count of streams files into one final file

```

```
&cat("$runDir${delim}strcnt*", "$runDir${delim}mstrcnt$runNum");
#(NOTE: there is a dependancy that this mstrcnt file exist before the
# calcmetrics.pl script is called, both because it is used as input for
# calcmetrics.pl, and because the output from calcmetrics is used as
# the trigger for watchstreams to complete, and watchstreams cats its
# output at the end of the mstrcnt file.

# generate the mtinter?.metrics file in the run directory
#require 'calcmetrics.pl';

if ( $runUF ne "no")
{
    system("perl calcmetrics.pl $numStream UF");
}
else
{
    system("perl calcmetrics.pl $numStream");
}

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

if ($runUF ne "no") {

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

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

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

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

# deactivate the database this needs to remain at the end of run throughput
so
# asynchronous writing of the log files completes.
system("db2 deactivate database $dbname");
$rc=&doddb_noconn("db2 get db cfg for $dbname | grep -i log >>
$runDir${delim}endLog.Info", $all_ln);
if ( $logDir ne "NULL" )
{
    $rc=&doddb_noconn("$dircmd $logDir >>
$runDir${delim}endLog.Info", $all_ln);
}

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

#Create Catalog info
$rc = system("perl catinfo.pl p");

if ( $rc != 0 )
{
    warn "catinfo failed!!!\n";
}

#Report current log info to the run# directory in a file called endLog.Info
```

```
system("perl getLogInfo.pl endLog");

# if we are in audit mode we must do a db2stop at the end of the
power/throughput run
if ( $RealAudit eq "yes" )
{
    system("db2stop");
}

1;

sub getConfig
{
    $testtype=$_[0];
    print "Getting database configuration.\n";
    $dbtunefile="$runDir${delim}m${testtype}dbtune${runNum}";
    open(DBTUNE, ">$dbtunefile") || die "Can't open $dbtunefile: $!\n";
    $timestamp=`perl gettimestamp "long"`;
    print DBTUNE "Database and Database manager configuration taken at :
$timestamp";
    close(DBTUNE);
    system("db2level >> $dbtunefile");
    system("db2 get database configuration for $dbname >> $dbtunefile");
    system("db2 get database manager configuration >> $dbtunefile");
    system("db2set >> $dbtunefile");
}

sub getOSTune
{
    $testtype=$_[0];
    if ( $platform eq "aix" || $platform eq "linux")
    {
        print "Getting OS and VMdatabase configuration.\n";
        $ostunefile="$runDir${delim}m${testtype}ostune${runNum}";
        open(OSTUNE, ">$ostunefile") || die "Can't open $ostunefile: $!\n";
        $timestamp=`perl gettimestamp "long"`;
        print OSTUNE "Operating System and Virtual Memory configuration
taken at : $timestamp";
        close(OSTUNE);
        system("${delim}usr${delim}samples${delim}kernel${delim}schedtune
>> $ostunefile");
        system("${delim}usr${delim}samples${delim}kernel${delim}vmtune
>> $ostunefile");
    }
    else
    {
        print "OS parameters retrieval not supported for $platform \n";
    }
}

sub verifyTPCDBatch
{
    $logfile=$_[0];
    $dbname=$_[1];
    $file="verifytpcdbatch.clp";
    open(VERTBL, ">$file") || die "Can't open $file: $!\n";
    print VERTBL "connect to $dbname;\n";
    print VERTBL "select name,creator,valid,last_bind_time,isolation from
sysibm.sysplan where name like 'TPCD%';\n";
    print VERTBL "connect reset;\n";
    print VERTBL "terminate;\n";
    close(VERTBL);
    system("db2 -vtf $file >> $logfile");
}
```

D.6 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";
require "tpcdmacro.pl";

# Make output unbuffered.
select(STDOUT);
$| = 1;

if (@ARGV > 0)
{
    $PairNum=$ARGV[0];
}
else
{
    die "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";
}
if (length($ENV{"TPCD_MODE"}) <= 0)
{
    die "TPCD_MODE env't var not set\n";
}
if (length($ENV{"TPCD_DDLPATH"}) <= 0)
{
    die "TPCD_DDLPATH environment variable 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"};
$ddlpath=$ENV{"TPCD_DDLPATH"};
$delim=$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"};
$mode=$ENV{"TPCD_MODE"};
$coldel="|";

if ( $platform eq "nt" )
{
    $sep="&";
}
else
{
    $sep=",";
}

if (( $mode eq "uni" ) || ( $mode eq "smp" ))
{
    print "Beginning ....Load of Update Function Data for pair $PairNum \n";
    system("db2 connect to tpcd\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}orders.tbl.u" ;
    $str="$str$PairNum.new of del modified by coldel| fastparse messages
orders.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 -tf $auditDir${delim}tools${delim}set_UF1_app_id.sql");
system("db2 connect reset\n");
}
else
{
    # load_update_on_node loads for each node (update function 1)
    #DJD $function=1;
    #DJD &dodb_noconn("cd $auditDir${delim}tools $sep
    $auditDir${delim}tools${delim}load_update_on_node $PairNum $function
    $inlistmax", "all_in");
    #DJD Added this line to do the loading....
    print "Calling system $ddlpath${delim}doUFLoad.bat 1 $PairNum";
    system("$ddlpath${delim}doUFLoad.bat 1 $PairNum");
}
exit (0);

```

D.7 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";
require "tpcdmacro.pl";

# Make output unbuffered.
select(STDOUT);
$| = 1;

if (@ARGV > 0)
{
    $PairNum=$ARGV[0];
}
else
{
    die "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($ENV{"TPCD_PHYS_NODE"}) <= 0)
{
    die "TPCD_PHYS_NODE env't var not set\n";
}
if (length($ENV{"TPCD_MODE"}) <= 0)
{
    die "TPCD_MODE env't var not set\n";
}
if (length($ENV{"TPCD_DDLPATH"}) <= 0)
{
    die "TPCD_DDLPATH environment variable 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"};
$ddlpath=$ENV{"TPCD_DDLPATH"};
$delim=$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"};
$mode=$ENV{"TPCD_MODE"};
$coldel="|";
$dblquote="";

if ( $platform eq "nt" )
{
    $sep="&";
}
else
{
    $sep=";";
}

if (( $mode eq "uni" ) || ( $mode eq "smp" ))
{

```

```

print "Beginning ....Load of Update Function Data for pair $PairNum \n";
system("db2 connect to tpcd\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;
}

#system("db2 -tf $auditDir${delim}tools${delim}set_UF2_app_id.sql");
}
else
{
    # load for each node
    #DJD $function=2;
    #DJD &dobd_noconn("cd $auditDir${delim}tools $sep
    $auditDir${delim}tools${delim}load_update_on_node $PairNum $function
    $inlistmax", "all_ln");
    print "Calling system $ddlpath${delim}doUfload.bat 2 $PairNum";
    system("$ddlpath${delim}doUfload.bat 2 $PairNum");
}
exit (0);

```

D.8 doufload.bat

REM Takes UfType and update_pair as parameters.
 set RAHSLEEPTIME=999999
 db2_all "||\" z: & cd \tpch\tools\ & doit_3params Ufload.bat %1 %2 ##"
 REM db2_all "||\" call z:\tpch\tools\doit_3params Ufload.bat %1 %2 ##"

D.9 ufload.bat

REM Takes UfType, update_pair and nodenum as parameters.
 z:
 cd \tpch\tools
 echo "UFLOAD PARS" %1 %2 %3
 perl load_UF%1_data %2 %3

D.10 load_uf1_data

```

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

# usage perl load_UF1_data [update pair number] [nodenum]

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"};
$lastpair = $ENV{"TPCD_GEN_UPDATEPAIRS"};
$sf=$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];
$Current_node=$ARGV[1];
print("PairNum= $PairNum, Current_Node = $Current_node\n");
for ($i=0, $nodestr="00000"; $i<$Current_node; $i++, $nodestr++) {}

print "Beginning ....Preload of Update Function Data. Lineitem \n";
system("db2 connect to tpcd\n");
$str="db2 \" load from ";
$str="$str$flatfilepath${delim}lineitem.tbl.u ";
$str="$str$PairNum.";
$str="$str$nodestr.new of del modified by coldel| fastparse messages
c:\temp\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 tpcd\n");
$str="db2 \" load from ";
$str="$str$flatfilepath${delim}order.tbl.u" ;
$str="$str$PairNum.";
$str="$str$nodestr.new of del modified by coldel| fastparse messages
c:\\temp\\order.msg.u";
if ($PairNum == $lastpair) {
    $str="$str$PairNum.new replace into TPCDTEMP.ORDERS_NEW
statistics yes nonrecoverable using c:\\temp_tables \" ";
}
else {
    $str="$str$PairNum.new replace into TPCDTEMP.ORDERS_NEW
statistics no nonrecoverable using c:\\temp_tables \" ";
}
print "$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;
}

```

D.11 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]

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];
$current_node=$ARGV[1];
print("current_node=$current_node\n");
for ($i=0, $nodestr="00000"; $i<$current_node; $i++, $nodestr++) { };

print "Beginning ....Preload of Update Function Data. Deletes \n";
system("db2 connect to tpcd\n");
$str="db2 \" load from ";
$str="$str$flatfilepath${delim}delete." ;
$str="$str$PairNum.";
$str="$str$nodestr.new of del modified by coldel| fastparse messages
c:\\temp\\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;
}

```

Appendix E: ACID Transaction Source Code

E.1 acid.sqc

```

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

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

#include "acid.h"

#if (defined(SQLPTX) || defined(SQLWINT) || defined(SQLSUN) ||
defined(Linux))
double nearest(double);
#endif /* SQLPTX */

#define DEADLOCK -911

/*
#define TRUNC2(d) ((floor((d)*100.0))/100.0)
*/
/*
#define TRUNC2(d) ((floor(nearest((d)*100.0)))*0.01)
*/
/*
#define TRUNC2(d) ((floor(nearest((d)*1000.0)/10.0)/100.0)
*/
#define TRUNC2(d) ((floor(nearest((d)*100000.0)/1000.0)/100.0))

void sqlerror(char *, struct sqlca *);

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
char dbname[8]; /* = "tpcd"; */
EXEC SQL END DECLARE SECTION;

#ifdef SQLWINT

/*
** redefine gettimeofday so I don't have to
** change too much aix-specific code
*/
/*#typedef struct timeval { unsigned tv_sec; unsigned tv_usec; }; */
typedef struct timezone { int dummy; };
struct timeb timer;

void gettimeofday( struct timeval *tv, struct timezone *tz)
{
ftime(&timer);
tv->tv_sec = timer.time;

```

```

tv->tv_usec = timer.millitm * 1000;
tz->dummy = 0;
}
#endif

/*-----*/
/* acidQ */
/*-----*/

int acidQ (struct acidQ_struct *acid)
{
time_t timeT;
FILE *out;
char out_fn[50];
struct timeval tv;
struct timezone tz;
int mypid;
int rc = 0;

EXEC SQL BEGIN DECLARE SECTION;
sqlint32 okey;
sqlint32 lEprice;
double eprice;
EXEC SQL END DECLARE SECTION;

okey = acid->o_key;

/* mypid = getpid(); */
mypid = acid->tag;

sprintf(out_fn,
"%s%cacidQ.out.%d",getenv("TPCD_TMP_DIR"),del(),mypid);
out=fopen(out_fn,"a");
if (out == NULL)
{
fprintf(stderr, "ERROR input file %s could not be appended
to!!\n",out_fn);
}

gettimeofday(&tv, &tz);
time(&timeT);
fprintf(out,"n----- START of acidQ tag: %d -----n\n",mypid);
fprintf(out, "acidQ tag: %d, begin transaction time: (%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: (%06uu) %s",
mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));

/*
** use the same sql code as used in the consistsql.pl to
** run the consistency acid queries. Note we assign an long int
** to lEprice (we make it 10s of pennies by * 1000). Then divide
** by 1000.0 and cast it to a double (eprice) for printing
*/

EXEC SQL
SELECT

INTEGER(DECIMAL(SUM(DECIMAL(INTEGER(INTEGER(DECIMAL
(INTEGER(100*DECIMAL(L_EXTENDEDPRI,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 specific okey.
    ** this ensures that the ast will be used instead of the base table
    ** Have to use ast_lEprice as double since this sum is so big
    */
    EXEC SQL
    SELECT
        SUM ( L_EXTENDEDPRICE*(1-L_DISCOUNT)*(1 + L_TAX))
    into :ast_lEprice
    FROM
        TPCD.LINEITEM;

    if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        fprintf(out,"ast_acidQ **ERROR** sqlcode = %d\n",sqlca.sqlcode);
        sqlerror("ast_acidQ: select sum(l_extendedprice)", &sqlca);
        goto Qerror;
    }
    ast_eprice = ast_lEprice; /* use ast_eprice for printout to be consistent*/

    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"AST_ACID tag: %d, after read of LINEITEM: (%us %06uu)
%s",
        mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    fprintf(out, "sum(l_extendedprice): %0.3f\n",
        ast_eprice);

    EXEC SQL COMMIT;
    if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        fprintf(out,"ast_acidQ **ERROR** sqlcode = %d\n",sqlca.sqlcode);
        sqlerror("ast_acidQ: COMMIT", &sqlca);
        goto Qerror;
    }
    acid->l_extendedprice = ast_eprice;

    rc = 0;
    goto Qexit;

Qerror:
EXEC SQL rollback work;
if (sqlca.sqlcode != 0) sqlerror("ast_acidQ: ROLLBACK FAILED",
&sqlca);

Qexit:
fprintf(out,"\n----- END of ast_acidQ tag: %d ----- \n\n",mypid);
fflush(out);fclose(out);
return(rc);
}

/*-----*/
/*      acidT                               */
/*-----*/
int acidT (struct acidT_struct *acid)
{
    time_t timeT;
    FILE *out;
    char out_fn[50];
    struct timeval tv;

```

```

struct timezone tz;
int mypid;
int rc = 0;

EXEC SQL BEGIN DECLARE SECTION;
sqlint32   o_key, l_key, delta;
sqlint32   l_partkey, l_supkey;
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_supkey, l_quantity,
         l_tax, l_discount,
         l_extendedprice
  FROM tpcd.lineitem
  WHERE l_orderkey = :o_key
  AND l_linenumber = :l_key
  FOR UPDATE OF l_extendedprice, l_quantity;

EXEC SQL DECLARE o_cursor CURSOR FOR
  SELECT o_totalprice
  FROM tpcd.orders
  WHERE o_orderkey = :o_key
  FOR UPDATE OF o_totalprice;

if (acid->termination < 0 || acid->termination > 3) acid->termination = 0;
o_key = acid->o_key;
l_key = acid->l_key;
delta = acid->delta;

if (acid->logging) {
  /* mypid = getpid(); */
  mypid = acid->tag;
  sprintf(out_fn,
"%s%cacidT.out.%d",getenv("TPCD_TMP_DIR"),del(),mypid);
  out=fopen(out_fn,"a");
  gettimeofday(&tv, &tz);
  time(&timeT);
  fprintf(out,"n----- START of acidT tag: %d -----n",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_supkey, :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_supkey: %d l_quantity: %0.3f\nl_tax:
%0.3f l_discount: %0.3f l_extendedprice: %0.3f\n",
        l_partkey, l_supkey, 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);
}

#ifdef DEBUG

```

```

{
    double zeroone= l_extendedprice * (1.0- l_discount);
    double zeroonetimes= (l_extendedprice * (1.0- l_discount))*100.0;
    double firstone = TRUNC2(l_extendedprice * (1.0-l_discount));
    double notone= TRUNC2 ( l_extendedprice * (1.0-l_discount)) *
(1.0+l_tax);
    double secondone= TRUNC2( TRUNC2( l_extendedprice * (1.0-
l_discount) ) * (1.0+l_tax) );
    printf("firstone= %f\n", firstone);
    printf("zeroone= %f\n", zeroone);
    printf("zeroonetimes= %f\n", zeroonetimes);
    printf("notone= %f\n", notone);
    printf("secondone= %f\n", secondone);
}
#endif
ototal = o_totalprice -
    TRUNC2( TRUNC2( l_extendedprice * (1-l_discount) ) *
(1+l_tax) );
new_ototal = TRUNC2( new_extprice * (1.0-l_discount) );
new_ototal = TRUNC2( new_ototal * (1.0+l_tax) );
new_ototal = ototal + new_ototal;

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

EXEC SQL UPDATE tpcd.orders
    SET o_totalprice = :new_ototal
    WHERE CURRENT OF o_cursor;
if (sqlca.sqlcode != 0) {
    if (sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
        fprintf(out, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } else {
        fprintf(stderr, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: UPDATE o_cursor", &sqlca);
    goto Terror;
}

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

/*
** why is this code in here? we don't want to
** commit until the history table has been updated as well
if (acid->termination == 0) {
    EXEC SQL CLOSE L_CURSOR;
    EXEC SQL CLOSE O_CURSOR;
    EXEC SQL COMMIT;
    if (sqlca.sqlcode != 0) {
        if (sqlca.sqlcode == DEADLOCK) goto retry_tran;
        rc = sqlca.sqlcode;
        if (acid->logging) {
            fprintf(out, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
        } else {
            fprintf(stderr, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
        }
        sqlerror("acidT: COMMIT", &sqlca);
        goto Terror;
    }
}

```

```

}
*/

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

EXEC SQL INSERT INTO tpcd.history values
(:l_partkey, :l_supkey, :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:
    // gjc@011214
    if (acid->logging) {
        gettimeofday(&tv, &tz);
        time(&timeT);
        fprintf(out, "acidT tag: %d, immediately before closing cursors: (%us
%06uu) %s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    EXEC SQL CLOSE L_CURSOR;
    // gjc@011214.
    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;
// gjc@011214.
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;
}

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 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 closing cursors: (%us
%06uu) %s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    EXEC SQL CLOSE L_CURSOR;
    // gjc@011214.
    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;
// gjc@011214.
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;
}

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 rollback work;
if (sqlca.sqlcode != 0) {
    if (sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
        fprintf(out, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } else {
        fprintf(stderr, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: ROLLBACK", &sqlca);
    goto Terror;
}
if (acid->logging) {
    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out, "acidT tag: %d, after ROLLBACK: (%us %06uu) %s",
        mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
}
break;
}

acid->l_partkey = l_partkey;
acid->l_suppkey = l_suppkey;
acid->l_quantity = l_quantity;
acid->l_tax = l_tax;
acid->l_discount = l_discount;
acid->l_extendedprice = l_extendedprice;
acid->o_totalprice = o_totalprice;

rc = 0;
goto Texit;

Terror:
    EXEC SQL CLOSE L_CURSOR;
    EXEC SQL CLOSE O_CURSOR;
    EXEC SQL rollback work;
    if (sqlca.sqlcode != 0) sqlerror("acidT: ROLLBACK FAILED", &sqlca);

Texit:
    if (acid->logging) {
        fprintf(out, "n----- END of acidT tag: %d -----n\n", mypid);
        fflush(out); fclose(out);
    }

```

```

    return(rc);
}

/*-----*/
/*      updateQ      */
/*-----*/
int updateQ (struct update_struct *us)
{
    FILE *out;
    time_t timeT;
    struct timeval tv;
    struct timezone tz;
    int qnum;
    int rc = 0;
    int i;
    int secs2sleep;
    char buff[256];
    struct acidtype {int logging;} a, *acid;

    EXEC SQL BEGIN DECLARE SECTION;
    double  acctbal;
    double  discount;
    double  price;
    sqlint32 availqty;
    sqlint32 size;
    EXEC SQL END DECLARE SECTION;

    qnum = us->qnum;

    acid = &a;
    acid->logging = 1;

    sprintf(buff, "%s%cupdate.out", getenv("TPCD_TMP_DIR"), del());
    out = fopen(buff, "a");

    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out, "n----- START of update -----n\n");
    fprintf(out, "update query number: %d, begin transaction time: (%us
%06uu) %s",
        qnum, tv.tv_sec, tv.tv_usec, ctime(&timeT));

    sqlca.sqlcode = 0;
    discount = 0.25;
    price = 5000.50;
    acctbal = 1000.00;
    availqty = 10;
    size = 5;

    for (i=1; i <= 2; i++) {
        gettimeofday(&tv, &tz);
        time(&timeT);
        fprintf(out, "update query number: %d, pass %d, immediately before
UPDATE: (%us %06uu) %s",
            qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));

        switch (qnum)
        {
            case 1:
            {
                EXEC SQL
                    UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT +
:discount
                    WHERE L_ORDERKEY IN (326,512,928,995);
                if (sqlca.sqlcode != 0) {
                    rc = sqlca.sqlcode;
                    if (acid->logging)
                    {
                        fprintf(out, "update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",

```

```

        qnum, i, sqlca.sqlcode);
    }
    else
    {
        fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
        qnum, i, sqlca.sqlcode);
    }
    sqlerror("update query number 1", &sqlca);
    goto Uerror;
}
discount = discount * (-1);
secs2sleep = 300;
break;
}
case 2:
{
    EXEC SQL
        UPDATE TPCD.SUPPLIER set S_ACCTBAL = S_ACCTBAL +
:acctbal
        WHERE S_NAME in
('Supplier#000000647','Supplier#000000070','Supplier#000000802');
    if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
            qnum, i, sqlca.sqlcode);
        }
        else
        {
            fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
            qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 2", &sqlca);
        goto Uerror;
    }
    acctbal = acctbal * (-1);
    secs2sleep = 90;
    break;
}
case 3:
{
    EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT +
:discount
        WHERE L_ORDERKEY IN (260930, 402497, 457859, 509889,
58117,
        538311, 588421, 416167, 97830, 90276);
    if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
            qnum, i, sqlca.sqlcode);
        }
        else
        {
            fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
            qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 3", &sqlca);
        goto Uerror;
    }
    discount = discount * (-1);
    secs2sleep = 300;

```

```

        break;
    }
    case 4:
    {
        if ( i ==1 ) {
            EXEC SQL
                UPDATE TPCD.ORDERS set O_ORDERDATE =
O_ORDERDATE - 6 MONTHS
                WHERE O_ORDERKEY = 67461;
            /* WHERE O_ORDERKEY IN
(22400,28515,34338,46596,67461,92644,98307);*/
        } else {
            EXEC SQL
                UPDATE TPCD.ORDERS set O_ORDERDATE =
O_ORDERDATE + 6 MONTHS
                WHERE O_ORDERKEY = 67461;
        }
        if (sqlca.sqlcode != 0) {
            rc = sqlca.sqlcode;
            if (acid->logging)
            {
                fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
            }
            else
            {
                fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
            }
            sqlerror("update query number 4", &sqlca);
            goto Uerror;
        }
        secs2sleep = 300;
        break;
    }
    case 5:
    {
        EXEC SQL
            UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT +
:discount
            WHERE L_ORDERKEY IN
(70976,566279,152897,84226,232483);
        if (sqlca.sqlcode != 0) {
            rc = sqlca.sqlcode;
            if (acid->logging)
            {
                fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
            }
            else
            {
                fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
            }
            sqlerror("update query number 5", &sqlca);
            goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 300;
        break;
    }
    case 6:
    {
        EXEC SQL
            UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT +
:discount

```

```

WHERE L_ORDERKEY IN
(33,131,161,195,229,230,231,323,353,356);
if (sqlca.sqlcode != 0) {
    rc = sqlca.sqlcode;
    if (acid->logging)
    {
        fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
            qnum, i, sqlca.sqlcode);
    }
    else
    {
        fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
            qnum, i, sqlca.sqlcode);
    }
    sqlerror("update query number 6", &sqlca);
    goto Uerror;
}
discount = discount * (-1);
secs2sleep = 300;
break;
}
case 7:
{
    EXEC SQL
    UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT +
:discount
    WHERE L_ORDERKEY IN
(562917,410659,16550,398401,157634,429920,45411);
    if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
            fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 7", &sqlca);
        goto Uerror;
    }
    discount = discount * (-1);
    secs2sleep = 300;
    break;
}
case 8:
{
    EXEC SQL
    UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT +
:discount
    WHERE L_ORDERKEY IN
(129569,343591,270242,254983,98500,28963);
    if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
            fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",

```

```

            qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 8", &sqlca);
        goto Uerror;
    }
    discount = discount * (-1);
    secs2sleep = 300;
    break;
}
case 9:
{
    EXEC SQL
    UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT +
:discount
    WHERE L_ORDERKEY IN
(113509,232997,246691,379233,448162,32134);
    if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
            fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 9", &sqlca);
        goto Uerror;
    }
    discount = discount * (-1);
    secs2sleep = 300;
    break;
}
case 10:
{
    EXEC SQL
    UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT +
:discount
    WHERE L_ORDERKEY IN
(516487,245411,265799,253025,6914,562020);
    if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
            fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 10", &sqlca);
        goto Uerror;
    }
    discount = discount * (-1);
    secs2sleep = 300;
    break;
}
case 11:
{
    EXEC SQL
    UPDATE TPCD.PARTSUPP set PS_AVAILQTY =
PS_AVAILQTY + :availqty

```

```

WHERE PS_PARTKEY IN
(12098,5134,13334,17052,3452,12552,1084,5797);
if (sqlca.sqlcode != 0) {
rc = sqlca.sqlcode;
if (acid->logging)
{
fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
qnum, i, sqlca.sqlcode);
}
else
{
fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
qnum, i, sqlca.sqlcode);
}
sqlerror("update query number 11", &sqlca);
goto Uerror;
}
availqty = availqty * (-1);
secs2sleep = 180;
break;
}
case 12:
{
if (i == 1) {
EXEC SQL
UPDATE TPCD.LINEITEM set L_RECEIPTDATE =
L_RECEIPTDATE - 3 YEARS
WHERE L_ORDERKEY IN
(33,70,195,355,677,837,960,962,1028);
} else {
EXEC SQL
UPDATE TPCD.LINEITEM set L_RECEIPTDATE =
L_RECEIPTDATE + 3 YEARS
WHERE L_ORDERKEY IN
(33,70,195,355,677,837,960,962,1028);
}
if (sqlca.sqlcode != 0) {
rc = sqlca.sqlcode;
if (acid->logging)
{
fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
qnum, i, sqlca.sqlcode);
}
else
{
fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
qnum, i, sqlca.sqlcode);
}
sqlerror("update query number 12", &sqlca);
goto Uerror;
}
secs2sleep = 300;
break;
}
case 13:
{
EXEC SQL
UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT +
:discount
WHERE L_ORDERKEY IN (263,9476,32355,34854,53445,56901);
if (sqlca.sqlcode != 0) {
rc = sqlca.sqlcode;
if (acid->logging)
{
fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",

```

```

qnum, i, sqlca.sqlcode);
}
else
{
fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
qnum, i, sqlca.sqlcode);
}
sqlerror("update query number 13", &sqlca);
goto Uerror;
}
discount = discount * (-1);
secs2sleep = 90;
break;
}
case 14:
{
EXEC SQL
UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT +
:discount
WHERE L_ORDERKEY IN (32,225,326,448,449,483,512);
if (sqlca.sqlcode != 0) {
rc = sqlca.sqlcode;
if (acid->logging)
{
fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
qnum, i, sqlca.sqlcode);
}
else
{
fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
qnum, i, sqlca.sqlcode);
}
sqlerror("update query number 14", &sqlca);
goto Uerror;
}
discount = discount * (-1);
secs2sleep = 180;
break;
}
case 15:
{
EXEC SQL
UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT +
:discount
WHERE L_ORDERKEY IN (1,4,7,35,135,131300);
if (sqlca.sqlcode != 0) {
rc = sqlca.sqlcode;
if (acid->logging)
{
fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
qnum, i, sqlca.sqlcode);
}
else
{
fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
qnum, i, sqlca.sqlcode);
}
sqlerror("update query number 15", &sqlca);
goto Uerror;
}
discount = discount * (-1);
secs2sleep = 180;
break;
}
case 16:

```

```

{
    EXEC SQL
        UPDATE TPCD.PART set P_SIZE = P_SIZE + :size
        WHERE P_PARTKEY IN (4,7,15,1313);
    if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
            fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 16", &sqlca);
        goto Uerror;
    }
    size = size * (-1);
    secs2sleep = 180;
    break;
}
case 17:
{
    EXEC SQL
        UPDATE TPCD.LINEITEM set L_EXTENDEDPRICE =
L_EXTENDEDPRICE + :price
        WHERE L_ORDERKEY IN (4065,110372,165061,265702,87138);
    if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
            fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 17", &sqlca);
        goto Uerror;
    }
    price = price * (-1);
    secs2sleep = 90;
    break;
}
default:
{
    fprintf(out,"ERROR: Invalid query number specified %d\n", qnum);
    rc = 1;
    goto Uexit;
}
}

gettimeofday(&tv, &tz);
time(&timeT);

if (acid->logging)
    fprintf(out,"update query number: %d, pass %d, after UPDATE: (%us
%06uu) %s",
        qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));
else
    fprintf(stderr,"update query number: %d, pass %d, after UPDATE:
(%us %06uu) %s",

```

```

        qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));

if ( i == 2 ) {
    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"update query number: %d, pass %d, sleeping for %d
seconds: (%us %06uu) %s",
        qnum, i, secs2sleep, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    fflush(out);
    system("touch /tmp/tpcd/update.sync.sleep");
    sleep(secs2sleep);
}

gettimeofday(&tv, &tz);
time(&timeT);
    fprintf(out,"update query number: %d, pass %d, immediately before
COMMIT: (%us %06uu) %s",
        qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));

    EXEC SQL COMMIT;
    if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        fprintf(out,"update pass %d, **ERROR** sqlcode = %d\n", i,
sqlca.sqlcode);
        sqlerror("update: COMMIT", &sqlca);
        goto Uerror;
    }
    gettimeofday(&tv, &tz);
    time(&timeT);
    if (acid->logging)
        fprintf(out,"update query number: %d, pass %d, after COMMIT: (%us
%06uu) %s",
            qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    else
        fprintf(stderr,"update query number: %d, pass %d, after COMMIT:
(%us %06uu) %s",
            qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));
}

rc = 0;
goto Uexit;

Uerror:
    EXEC SQL rollback work;
    if (sqlca.sqlcode != 0) sqlerror("update: ROLLBACK FAILED", &sqlca);
    system("touch /tmp/tpcd/update.sync.sleep");

Uexit:
    fprintf(out,"n----- END of update -----n\n");
    fflush(out);fclose(out);
    return(rc);
}

/*-----*/
/*    connect_to_TM                                */
/*-----*/
void connect_to_TM( void )
{
    char *dbname_ptr;
    if ((dbname_ptr = getenv("TPCD_QUAL_DBNAME")) != NULL) {
        fprintf(stderr,"***** %s *****n",dbname_ptr);
        strcpy (dbname, dbname_ptr);
    }

    EXEC SQL CONNECT TO :dbname IN SHARE MODE;
    if (sqlca.sqlcode < 0) {
        fprintf(stderr, "CONNECT TO %s failed SQLCODE = %d\n", dbname,
sqlca.sqlcode);
        exit(-1);
    }
}

```

```

    return;
}

/*-----*/
/*    disconnect_from_TM    */
/*-----*/
void disconnect_from_TM ( void )
{
    EXEC SQL CONNECT RESET;
    if (sqlca.sqlcode < 0) {
        fprintf(stderr, "DISCONNECT failed SQLCODE = %d\n",
            sqlca.sqlcode);
        exit(-1);
    }
    return;
}

/*-----*/
/*    sqlerror    */
/*-----*/
void sqlerror(char *msg, struct sqlca *psqlca)
{
    FILE *err_fp;

    char err_fn[256];

    int j,k;

    sprintf(err_fn, "%s%cacid.sqlerrors", getenv("TPCD_TMP_DIR"), del());
    err_fp=fopen(err_fn, "a");
    fprintf(err_fp, "acid: sqlcode: %4d %s\n", psqlca->sqlcode, msg);
    fprintf(stderr, "acid: sqlcode: %4d %s\n", psqlca->sqlcode, msg);
    fflush(stderr);
    if (psqlca->sqlerrmc[0] != ' ' || psqlca->sqlerrmc[1] != ' ') {
        fprintf(err_fp, "acid: slerrmc: ");
        for(j = 0; j < 5; j++)
        {
            for(k = 0; k < 14; k++) fprintf(err_fp, "%x ", psqlca-
>sqlerrmc[j*10+k]);
            fprintf(err_fp, " ");
            for(k = 0; k < 14; k++) fprintf(err_fp, "%c", psqlca-
>sqlerrmc[j*10+k]);
            fprintf(err_fp, "\n");
            if (j < 4) fprintf(err_fp, " ");
        }

        fprintf(err_fp, "acid: sqlerrp: ");
        for(j = 0; j < 8; j++) fprintf(err_fp, "%c", psqlca->sqlerrp[j]);
        fprintf(err_fp, "\n");

        fprintf(err_fp, "acid: sqlerrd: ");
        for(j = 0; j < 6; j++) fprintf(err_fp, "%d", psqlca->sqlerrd[j]);
        fprintf(err_fp, "\n");

        if (psqlca->sqlwarn[0] != ' ') {
            fprintf(err_fp, "acid: sqlwarn: ");
            for(j = 0; j < 8; j++) fprintf(err_fp, "%c ", psqlca->sqlwarn[j]);
            fprintf(err_fp, "\n");
        }

        fprintf(err_fp, "\n");
        fflush(err_fp); fclose(err_fp);
    }

#ifdef SQLWINT
void sleep(int sec)
{

```

```

    Sleep(sec * 1000);
}
#endif

char del(void)
{
#ifdef SQLWINT
    return '\\';
#else
    return '/';
#endif
}

#ifdef defined(SQLPTX) || defined(SQLWINT) || defined(SQLSUN) ||
defined(Linux)
/* added for PTX as this one is not there in libm */
double nearest(double x)
{
    double y, z;

    y = x;
    if (x < 0)
        y = -x;
    z = y - (int)y;
    if (z == 0.5) {
        if ((int)floor(y) % 2) {
            return((x < 0) ? -ceil(y) : ceil(y));
        } else {
            return((x < 0) ? -floor(y) : floor(y));
        }
    } else if (z < 0.5)
        return((x < 0) ? -floor(y) : floor(y));
    else
        return((x < 0) ? -ceil(y) : ceil(y));
}
#endif /* SQLPTX */

```

E.2 acid.h

```

#include <math.h>

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

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

struct update_struct {
    int qnum;
};

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

struct acidT_struct {
    int termination;
    int tag;
    int logging;
    long o_key;

```

```
long  l_key;
long  delta;
long  l_partkey;
long  l_suppkey;
double l_quantity;
double l_tax;
double l_discount;
double l_extendedprice;
double o_totalprice;
};

/*
}
```

```
** in acid.sqc
*/

int updateQ (struct update_struct *us);

char del(void);

#ifdef SQLWINT
void sleep (int sec);
#endif
```

Appendix F: Price Quotations



Route 100
Somers, NY 10589

January 15, 2003

Mr. Michael V. Nikolaiev
Director, ISS, Database Engineering
H P Company
20555 SH-249 - MS-150402
Houston, TX 77070

Via Facsimile 281-514-8375

Dear Mr. Nikolaiev

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

All prices shown are in U.S. Dollars:

DESCRIPTION	PRICE
DB2 Enterprise Extended Edition License With 1 Year Support (Per Processor) 128 Processors @ \$22,153 Each	\$2,835,584.00
DB2 Enterprise Extended Edition Support Renewal (Per Processor/Per Year) 2 Years for 128 Processors @\$1,081 Each	\$276,736.00

This quote is valid for 90 days.

If I can be of any further assistance, please contact me at 914/766-1325 or send me a note at privot@us.ibm.com.

Yours Truly,

Paul K. Rivot
Director, Database Servers &
Business Intelligence Software

CC: D. Daly



[SIGN IN](#)
[POLICIES/HELP](#)
[CONTACT US](#)
[WISH LIST](#)



Ultra-Compact

Click for Advanced Search

SUMMIT7I 28PT 100/1000BT SWCH 4PT GBIC 1F

► PRODUCTS

[Accessories](#)
[Audio/Video](#)
[Cases](#)
[CD-R/RW/DVD](#)
[Computer Systems](#)
[Digital Cameras](#)
[Education/Training](#)
[Internal Components](#)
[Memory](#)
[Monitors/Displays](#)
[MP3/PDA](#)
[Network/Connectivity](#)
[Power/Surge Devices](#)
[Printers/Scanners](#)
[Software/Books](#)
[Storage Devices](#)

\$15,123.49
MFG # 11701

CHECK
STOCK

ADD TO
WISH LIST

NOTIFY ME
WHEN IN STOCK

Technical Specifications are currently unavailable for this product.

► TOOLS/RESOURCES

[Sign In](#)
[Order Tracking](#)
[View Previous Carts](#)
[View Wish List](#)
[Hot Deals](#)
[About EMS](#)

At this time, the manufacturer of this product has not supplied EMS Computing specifications. For more information on this item, we encourage you to visit the manufacturer's website. Please be aware that EMS Computing offers only brand new, factory direct merchandise to our customers. As such, you may rest assured that when you make your product choice, you may buy with confidence.

[Contact Us](#) | [Policies/Help](#) | [Sign In](#) | [View Cart](#) | [Site Map](#)

Please direct all comments or questions about this website to info@emscomputing.com
 Copyright © 2001, EMS Computing, Inc., All Rights Reserved.
[Terms and Conditions](#)

.../specs.asp?pid="069283"&sid="3c9f422b002f93a22741d8b0e20c06b3"&price="\$15,123.493/25/2002