

TPC Benchmark™ H Full Disclosure Report
for
IBM® @server® xSeries® 346
using
IBM DB2® UDB 8.2

Submitted for Review

December 14, 2004



First Edition - December 2004

THE INFORMATION CONTAINED IN THIS DOCUMENT IS DISTRIBUTED ON AN AS IS BASIS WITHOUT ANY WARRANTY EITHER EXPRESSED OR IMPLIED. The use of this information or the implementation of any of these techniques is the customer's responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

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

This publication was produced in the United States. IBM may not offer the products, services, or features discussed in this document in other countries, and the information is subject to change without notice. Consult your local IBM representative for information on products and services available in your area.

© Copyright International Business Machines Corporation 2004. All rights reserved.

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

U.S. Government Users - Documentation related to restricted rights: Use, duplication, or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Trademarks

IBM, the IBM eServer logo, DB2, DB2 Universal Database, and xSeries are trademarks or registered trademarks of International Business Machines Corporation.

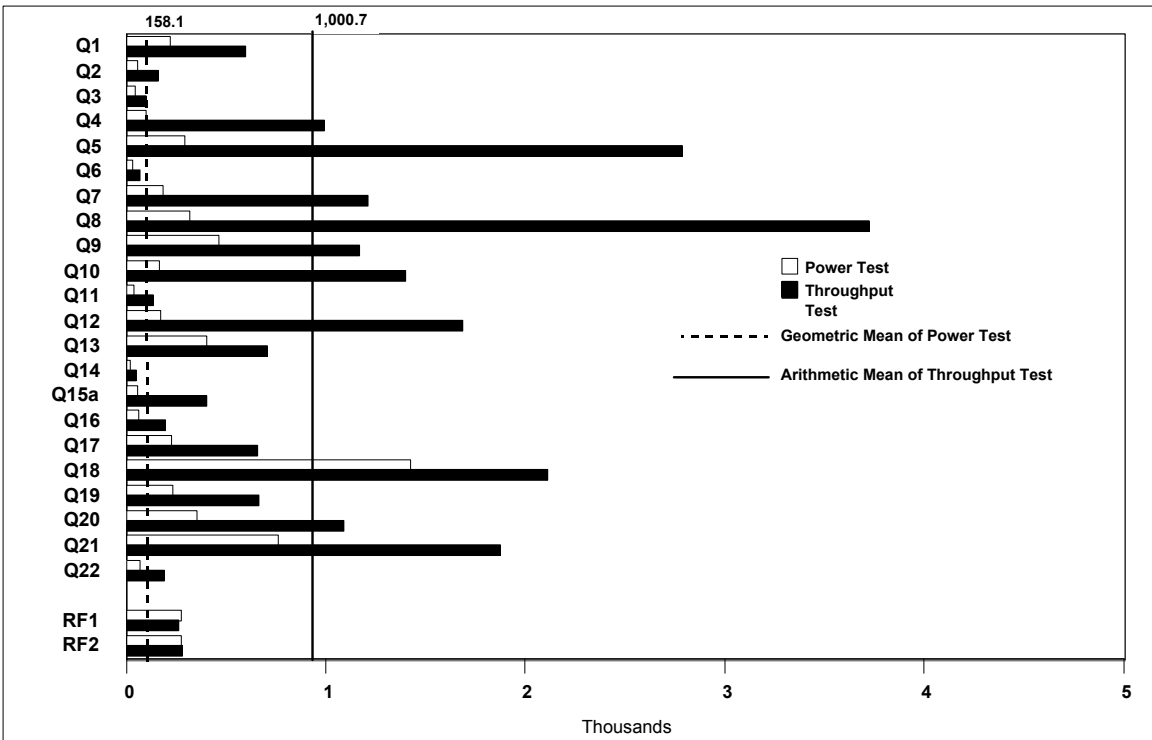
The following terms used in this publication are trademarks of other companies as follows: TPC Benchmark, TPC-H, QppH QthH and QphH are trademarks of Transaction Processing Performance Council; Intel and Xeon are trademarks or registered trademarks of Intel Corporation; Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Other company, product, or service names, which may be denoted by two asterisks (**), may be trademarks or service marks of others.

Notes

¹ GHz only measures microprocessor internal clock speed, not application performance. Many factors affect application performance.

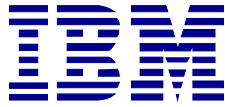
² When referring to hard disk capacity, one GB equals one billion bytes. Total user-accessible capacity may be less.

		IBM® @server® xSeries® 346 with IBM DB2® UDB 8.2		TPC-H Rev 2.1.0
				Report Date: Dec. 14, 2004
Total System Cost		Composite Query-per-Hour Metric		Price/Performance
\$26,536 USD		1,894.2 QphH @ 100GB		\$14 USD per QphH @ 100GB
Database Size	Database Manager	Operating System	Other Software	Availability Date
100GB	IBM DB2 UDB 8.2	SUSE® Linux® Enterprise Server 9	None	Dec. 14, 2004



Query	Power Test (Thousands)	Throughput Test (Thousands)
Q1	158.1	1,000.7
Q2	0.2	0.2
Q3	0.1	0.1
Q4	0.1	1.0
Q5	0.2	2.8
Q6	0.1	0.1
Q7	0.1	1.2
Q8	0.1	3.7
Q9	0.1	1.2
Q10	0.1	1.4
Q11	0.1	0.2
Q12	0.1	1.7
Q13	0.2	0.7
Q14	0.1	0.1
Q15a	0.1	0.4
Q16	0.1	0.2
Q17	0.1	0.7
Q18	0.1	2.1
Q19	0.1	0.7
Q20	0.1	1.1
Q21	0.1	1.9
Q22	0.1	0.2
RF1	0.2	0.2
RF2	0.2	0.2

Database Load Time: 04:07:28		Load Included Backup: Y	Total Data Storage / Database Size: 6.78
RAID (Base Tables Only): N		RAID (Base Tables and Auxiliary Data Structures): N	RAID (All): N
Configuration			
Processors	2	Intel Xeon 3.6GHz / 1MB L2 Cache	
Memory	4	1GB (2x512MB) PC-3200 DDR ECC SDRAM RDIMM	
Disk Controllers	1	ServeRAID-7k Ultra320 SCSI Embedded Interface	
Disk Drives	20	36.4GB 15K Ultra320 SCSI Drive	
Total Disk Storage		678GB	



IBM @server xSeries 346 with IBM DB2 UDB 8.2

TPC-H 2.1.0 Executive Summary

Report Date: December 14, 2004

Description	Part Number	Third Party Brand	Pricing	Unit Price	Quantity	Extended Price	3-Yr. Maint. Price
Server Hardware							
IBM e(logo)server xSeries 346 w/3.6GHz Xeon w/1 x 3.6GHz/1MB Xeon MP Processors w/2 x 512MB ECC SDRAM RDIMMs Integrated Dual-Channel Gigabit Ethernet	8840-41U	IBM	1	3,559	1	3,559	
ServicePac for 3-Year 24x7x4 Support	21P2078	IBM	1	600	1		600
3.6GHz Xeon Processor Upgrade	13N0685	IBM	1	1,429	1	1,429	
1GB (2x512MB) PC-3200 DDR ECC SDRAM RDIMM	73P3522	IBM	1	399	3	1,197	
IBM ServeRAID-7K SCSI Controller	71P8642	IBM	1	449	1	449	
IBM Preferred Pro Full-Size Keyboard	31P7415	IBM	1	29	1	29	
IBM Sleek 2-Button Mouse	28L3673	IBM	1	15	1	15	
E54 15" (13.8" Viewable) Color Monitor	633147N	IBM	1	149	1	149	
ServicePac for 3-Year 24x7x4 Support	30L9183	IBM	1	90	1		90
					Subtotal	6,827	690
Server Storage							
IBM EXP400 Rack Storage Enclosure	17331RU	IBM	1	3,099	1	3,099	
ServicePac for 3-Year 24x7x4 Support	41L2768	IBM	1	760	1		760
36.4GB 15K U320 SCSI Drive	90P1318	IBM	1	349	20	6,980	
					Subtotal	10,079	760
Server Software							
DB2 Universal Database Express Edition Server Lic+SW Maint 12 Mo	D52BILL	IBM	1	624	1	624	
Server License Maintenance 12 Months		IBM	1	125	2		250
DB2 Universal Database Express Edition Server User Lic+SW Maint 12 Mo	D52BKLL	IBM	1	124	50	6,200	
User License Maintenance 12 Months		IBM	1	25	100		2,500
SUSE Linux Enterprise Server 9 2 CPU with 3 Yr Maint.		SUSE	2	870	1	870	
					Subtotal	7,694	2,750
Compsat Technology discount of 12.33%					Discount	2,090	174
					Total	22,510	4,026

Three-Year Cost of Ownership:	26,536
-------------------------------	--------

Pricing: 1 - Compsat Technology; 2 - Novell

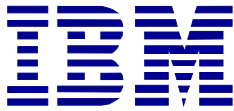
QphH@100GB:	1,894.2
-------------	---------

Warranty and Maintenance: The standard warranty has been upgraded to 3 years of 24x7x4 coverage.

\$USD/QphH@100GB: \$14

Audited by Francois Raab, InfoSizing, Inc.

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



IBM @server xSeries 346
with
IBM DB2 UDB 8.2

TPC-H Rev 2.1.0

Report Date: Dec. 14, 2004

Measurement Results:

Database Scale Factor	100
Total Data Storage/Database Size	6.78
Start of Database Load	06:28:50
End of Database Load	10:36:18
Database Load Time	04:07:28
Query Streams for Throughput Test	5
TPC-H Power	2,277.7
TPC-H Throughput	1,575.2
TPC-H Composite Query-per-Hour (QphH@300GB)	1,894.2
Total System Price over 3 Years	\$26,536 USD
TPC-H Price/Performance Metric (\$/QphH@300GB)	\$14 USD

Measurement Interval:

Measurement Interval in Throughput Test (Ts) = 25,139

Duration of Stream Execution:

	Seed	Query Start Date/Time Query End Date/Time	RF1 Start Date/Time RF1 End Date/Time	RF2 Start Date/Time RF2 End Date/Time	Duration
Stream 00	1205103618	12/05/04 14:11:47 12/05/04 15:47:06	12/05/04 14:07:10 12/05/04 14:11:47	12/05/04 15:47:06 12/05/04 15:51:43	1:35:19 0:09:14
Stream 01	1205103619	12/05/04 15:51:47 12/05/04 21:58:53	12/05/04 15:51:47 12/05/04 22:09:08	12/05/04 22:09:08 12/05/04 22:13:53	6:13:04 0:09:02
Stream 02	1205103620	12/05/04 15:51:47 12/05/04 22:02:31	12/05/04 22:13:53 12/05/04 22:18:23	12/05/04 22:18:23 12/05/04 22:23:09	6:10:44 0:09:16
Stream 03	1205103621	12/05/04 15:51:47 12/05/04 22:04:50	12/05/04 22:23:09 12/05/04 22:27:36	12/05/04 22:27:36 12/05/04 22:32:21	6:13:04 0:09:12
Stream 04	1205103622	12/05/04 15:51:47 12/05/04 21:58:02	12/05/04 22:32:21 12/05/04 22:36:50	12/05/04 22:36:50 12/05/04 22:41:32	6:06:14 0:09:11
Stream 05	1205103623	12/05/04 15:51:48 12/05/04 21:49:15	12/05/04 22:41:32 12/05/04 22:46:01	12/05/04 22:46:01 12/05/04 22:50:46	5:57:27 0:09:14



IBM @server xSeries 346
with
IBM DB2 UDB 8.2

TPC-H Rev 2.1.0

Report Date: Dec. 14, 2004

TPC-H Timing Intervals (in seconds):

Query	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Stream 00	221.7	56.5	46.8	98.5	296.1	33.4	188.4	316.7	465.6	167.3	40.9	173.9
Stream 01	557.1	163.9	130.1	1,338.3	3,661.4	73.7	522.1	4,330.1	1,190.3	893.5	70.8	1,527.6
Stream 02	880.5	148.9	172.9	1,771.2	1,902.0	104.6	866.4	3,518.8	1,355.3	1,717.1	120.5	1,719.2
Stream 03	318.1	277.8	45.9	734.6	2,537.3	37.7	842.5	5,600.2	821.6	929.3	335.7	1,286.4
Stream 04	842.5	142.1	102.0	562.7	3,988.3	49.6	2,125.0	2,491.9	866.8	1,432.9	95.9	1,199.1
Stream 05	410.8	85.5	59.9	578.4	1,847.8	74.0	1,715.1	2,680.2	1,617.5	2,047.0	47.0	2,719.9
Minimum	318.1	85.5	45.9	562.7	1,847.8	37.7	522.1	2,491.9	821.6	893.5	47.0	1,199.1
Average	601.8	163.6	102.2	997.0	2,787.4	67.9	1,214.2	3,724.2	1,170.3	1,404.0	134.0	1,690.4
Maximum	880.5	277.8	172.9	1,771.2	3,988.3	104.6	2,125.0	5,600.2	1,617.5	2,047.0	335.7	2,719.9

Stream ID	Q13	Q14	Q15a	Q16	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream 00	402.0	19.8	57.5	63.4	228.3	1,425.4	232.3	356.1	760.0	68.7	276.7	277.3
Stream 01	738.5	75.3	265.8	385.0	386.3	1,611.3	678.0	810.1	2,319.4	298.1	258.0	284.6
Stream 02	599.5	52.0	116.2	132.9	819.6	2,862.1	793.9	1,521.0	916.9	152.7	270.3	286.0
Stream 03	871.5	37.8	1,375.1	124.6	254.8	1,809.3	782.8	1,278.0	1,973.8	108.6	267.1	285.0
Stream 04	826.0	55.8	179.8	124.9	924.6	2,495.1	459.9	853.0	1,857.8	298.7	268.4	282.1
Stream 05	497.8	35.9	74.2	216.8	902.4	1,778.8	623.5	1,005.0	2,327.9	102.0	269.4	285.0
Minimum	497.8	39.9	74.2	124.6	254.8	1,611.3	459.9	810.1	916.9	102.0	258.0	282.1
Average	706.7	51.4	402.2	196.8	657.5	2,111.3	667.6	1,093.4	1,879.2	192.0	266.6	284.5
Maximum	871.5	75.3	1,375.1	385.0	924.6	2,862.1	793.9	1,521.0	2,327.9	298.7	270.3	286.0

Table of Contents

Preface	10
1 General Items	12
1.1 Benchmark Sponsor	12
1.2 Parameter Settings	12
1.3 Configuration Diagrams	12
1.3.1 Priced and Measured Configurations	13
2 Clause 1: Logical Database Design Related Items	14
2.1 Database Table Definitions	14
2.2 Database Organization	14
2.3 Horizontal Partitioning	14
2.4 Replication	14
3 Clause 2: Queries and Update Functions Related Items	15
3.1 Query Language	15
3.2 Random Number Generation	15
3.3 Substitution Parameters Generation	15
3.4 Query Text and Output Data from Database	15
3.5 Query Substitution Parameters and Seeds Used	15
3.6 Query Isolation Level	15
3.7 Refresh Function Implementation	16
4 Clause 3: Database System Properties Related Items	17
4.1 Atomicity Requirements	17
4.1.1 Atomicity of Completed Transactions	17
4.1.2 Atomicity of Aborted Transactions	17
4.2 Consistency Requirements	17
4.2.1 Consistency Condition	17
4.2.2 Consistency Tests	18
4.3 Isolation Requirements	18
4.3.1 Isolation Test 1	18
4.3.2 Isolation Test 2	18
4.3.3 Isolation Test 3	19
4.3.4 Isolation Test 4	19
4.3.5 Isolation Test 5	19
4.3.6 Isolation Test 6	20
4.4 Durability Requirements	20
4.4.1 Failure of a Durable Medium	20
4.4.2 Loss of Log and System Crash	20
4.4.3 System Crash	21
4.4.4 Memory Failure	21
5 Clause 4: Scaling and Database Population Related Items	22
5.1 Cardinality of Tables	22
5.2 Distribution of Tables and Logs	22
5.3 Database Partition / Replication Mapping	24
5.4 RAID Implementation	24
5.5 DBGEN Modifications	25
5.6 Database Load Time	25
5.7 Data Storage Ratio	25
5.8 Database Load Mechanism Details and Illustration	25
5.9 Qualification Database Configuration	26
6 Clause 5: Performance Metrics and Execution Rules Related Items	27
6.1 System Activity between Load and Performance Tests	27
6.2 Steps in the Power Test	27

6.3 Timing Intervals for Each Query and Refresh Function	27
6.4 Number of Streams for the Throughput Test	27
6.5 Start and End Date/Times for Each Query Stream	27
6.6 Total Elapsed Time for the Measurement Interval	27
6.7 Refresh Function Start Date/Time and Finish Date/Time	27
6.8 Timing Intervals for Each Query and Each Refresh Function for Each Stream	28
6.9 Performance Metrics	28
6.10 Performance Metric and Numerical Quantities from Both Runs	28
6.11 System Activity between Tests	28
7 Clause 6: SUT and Driver Implementation Related Items	29
7.1 Driver	29
7.2 Implementation-Specific Layer	29
7.3 Profile-Directed Optimization	29
8 Clause 7: Pricing Related Items	30
8.1 Hardware and Software Components	30
8.2 Three-Year Cost of System Configuration	30
8.3 Availability Dates	30
8.4 Country-Specific Pricing	30
Clause 9: Audit Related Items	31
9.1 Auditor's Report	31
Appendix A: Tunable Parameters and System Configuration	34
DB2 UDB 8.2 Database Manager Configuration	34
DB2 UDB 8.2 Database Configuration	34
DB2 Version	35
DB2 Registry Variables	35
SUSE Linux Version	36
SUSE Linux Configuration Parameters	36
Appendix B: Database Build Scripts	37
bpvars	37
buildtpcd	37
createmseedme.pl	45
createUFTables	45
dss.bp	46
dss.db2set	46
dss.db2set.load	46
dss.dbcfg	46
dss.dbcfg.load	46
dss.dbmcfg	46
dss.dbmcfg.load	46
dss.index	46
dss.load	47
dss.load.qual	48
dss.runstats	49
dss.tbl	50
dss.tbsp	50
ploaduf1	51
ploaduf2	51
tpcd.setup	51
Appendix C: Qualification Query Output	54
Qualification Queries	54
First 10 Rows of the Database	64
Query Substitution Parameters	67
Appendix D: Driver Source Code	70
load_line_uf	70

load_orders_uf	70
makefile	70
ploadUF1	70
ploadUF2	70
preloadUF	70
runpower	70
runthroughput	74
tpcdbatch.h	76
tpcdbatch.sqc	78
tpcdUF.sqc	104
Appendix E: ACID Transaction Source Code	110
acid.h	110
acid.sqc	110
makefile	120
Appendix F: Price Quotations	121

Preface

TPC Benchmark H Standard Specification was developed by the Transaction Processing Performance Council (TPC). It was released on February 26, 1999, and most recently revised (Revision 2.0) October 29, 2002. This is the full disclosure report for benchmark testing of the IBM *@server* xSeries 346 according to the TPC Benchmark H Standard Specification.

The TPC Benchmark H is a decision support benchmark. It consists of a suite of business-oriented ad hoc queries and concurrent data modifications. The queries and the data populating the database have been chosen to have broad industrywide relevance while maintaining a sufficient degree of ease of implementation. This benchmark illustrates decision support systems that:

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

TPC-H evaluates the performance of various decision support systems by the execution of set of queries against a standard database under controlled conditions. The TPC-H queries:

- Give answers to real-world business questions;
- Simulate generated ad-hoc queries (e.g., via a point-and-click GUI interface);
- Are far more complex than most OLTP transactions;
- Include a rich breadth of operators and selectivity constraints;
- Generate intensive activity on the part of the database server component of the system under test;
- Are executed against a database complying with specific population and scaling requirements;
- Are implemented with constraints derived from staying closely synchronized with an on-line production database.

The TPC-H operations are modeled as follows:

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

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

The performance metrics reported by TPC-H is called the TPC-H Composite Query-per-Hour Performance Metric (QphH@Size), and reflects multiple aspects of the capability of the system to process queries. These aspects include the selected database size against which the queries are executed, the query processing power when queries are submitted by a single stream, and the query throughput when queries are submitted by multiple concurrent users. The TPC-H Price/Performance metric is expressed as \$/QphH@Size. To be compliant with the TPC-H standard, all references to TPC-H results for a given configuration must include all required reporting components (see Clause 5.4.6). The TPC believes that comparisons of TPC-H results measured against different database sizes are misleading and discourages such comparisons.

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

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

1 General Items

1.1 Benchmark Sponsor

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

IBM Corporation sponsored this TPC-H benchmark.

1.2 Parameter Settings

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

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

Appendix A, “Tunable Parameters,” contains a list of all DB2 parameters and operating system parameters. Session initialization parameters can be set during or immediately after establishing the connection to the database within the tpcdbatch program documented in Appendix D, “Driver Source Code.” This result uses the default session initialization parameters established during preprocessing/binding of the tpcdbatch program.

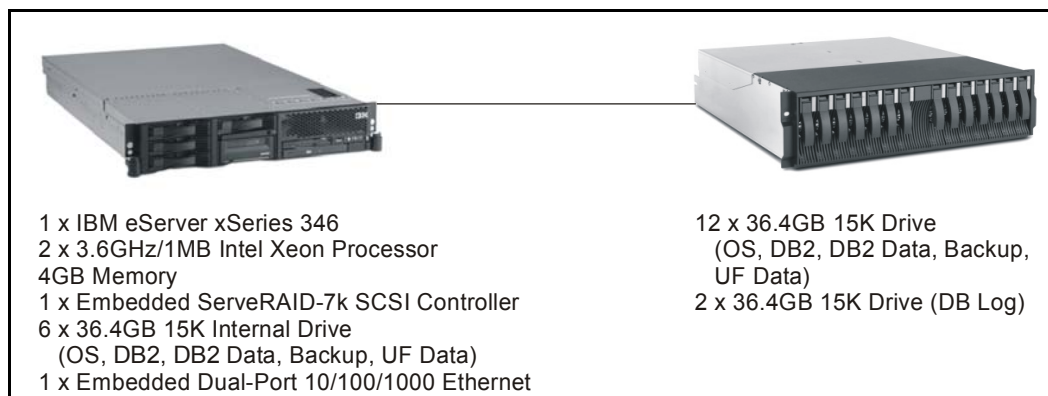
1.3 Configuration Diagrams

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

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

The configuration diagram for the tested and priced system is provided on the following page.

1.3.1 Priced and Measured Configurations



The priced configuration for the xSeries 346 contained:

- Two Intel Xeon 3.6GHz processors, each with 1MB of L2 cache
- Eight (8) 512MB PC-3200 ECC SDRAM RDIMMs
- One embedded dual-port 10/100/1000 Ethernet interface
- One ServeRAID-7k Ultra320 SCSI controllers
- Twenty (20) 36.4GB 15K Ultra320 SCSI disk drives
- One (1) EXP400 Storage Expansion Enclosure

The measured and priced configurations were identical with the exception that the priced configuration contained two Qlogic 2342 controllers that were attached to Fibre disk storage. One controller and its disks were unused. The other Fibre controller was attached to disks that contained the flat file data. These components were not priced.

2 Clause 1: Logical Database Design Related Items

2.1 Database 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 contains the scripts that were used to set up the TPC-H test and qualification databases.

2.2 Database Organization

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

Appendix B contains the scripts that were used to create the indexes on the test and qualification databases.

2.3 Horizontal Partitioning

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

Partitioning was not used.

2.4 Replication

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

Replication was not used.

3 Clause 2: Queries and Update Functions Related Items

3.1 Query Language

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

SQL was the query language used.

3.2 Random Number Generation

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

The TPC-supplied DBGEN version 1.3.0 and QGEN version 1.3.0 were used to generate all database populations.

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 contains the output for each of the qualification queries. The functional query definitions and variants used in this disclosure use the following minor query modifications:

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

3.5 Query Substitution Parameters and Seeds Used

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

Appendix C contains the seed and query substitution parameters used.

3.6 Query 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 Function Implementation

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, “Driver Source Code.”

4 Clause 3: Database System Properties Related Items

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

All ACID tests were conducted according to specifications. The Atomicity, Isolation, Consistency and Durability tests were performed on the xSeries 365 server. Appendix E contains the ACID transaction source code.

4.1 Atomicity Requirements

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

4.1.1 Atomicity of Completed Transactions

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

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

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

4.1.2 Atomicity of Aborted Transactions

Perform the ACID transactions 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 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 order key. The number of records in the HISTORY table was also retrieved.
3. The ACID transaction was executed for the orderkey used in step 2. The transaction was rolled back.
4. The total price and the extended price were retrieved for the same orderkey used in step 2 and step 3. It was verified that the extended price and the total price were the same as in step 2.

4.2 Consistency Requirements

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

4.2.1 Consistency Condition

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

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

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

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

```
SELECT DECIMAL(SUM(DECIMAL(INTEGER(INTEGER(DECIMAL
(INTEGER(100*DECIMAL(L_EXTENDEDPRICE,20,2)),20,3)*
(1-L_DISCOUNT))*(1+L_TAX)),20,3)/100.0),20,3)
FROM TPCD.LINEITEM WHERE L_ORDEYKEY=okey
SELECT DECIMAL(SUM(O_TOTALPRICE,20,3)) from TPCD.ORDERS WHERE O_ORDERKEY =
okey
```

4.2.2 Consistency Tests

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

The queries defined in 4.2.1, “Consistency Condition,” were run after initial database build and prior to executing the ACID transaction. The queries showed that the database was in a consistent condition.

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

4.3 Isolation Requirements

4.3.1 Isolation Test 1

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

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

1. 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

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

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

1. 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 out by the ACID transaction.
3. First session: The ACID transaction is released and the Rollback is executed, releasing the read.

4. Second session: With the LINEITEM record now released, the ACID query completes.

4.3.3 Isolation Test 3

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

The following steps were performed to verify isolation of two refresh 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 1st 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:

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

4.3.4 Isolation Test 4

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

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

1. 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 1st session. This transaction is forced to wait while the 1st session holds a lock on the LINEITEM record requested by the second session.
3. First session: Rollback the ACID transaction T1. With the LINEITEM record now released, the ACID transaction T2 completes.
4. Verify that $T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE$

4.3.5 Isolation Test 5

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

1. 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. First session: Start a second ACID transaction, T2, which selects random values of PS_PARTKEY and PS_SUPPKEY and returns all columns of the PARTSUPP table for which PS_PARTKEY and PS_SUPPKEY are equal to the selected values.
3. T2 completed.
4. T1 was allowed to complete.
5. It was verified that the appropriate rows in the ORDERS, LINEITEM and HISTORY tables have been changed.

4.3.6 Isolation Test 6

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

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

4.4 Durability Requirements

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

4.4.1 Failure of a Durable Medium

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

The database log was stored on RAID-1 protected storage. The tables for the database were stored on RAID-0 storage, with the exception of the Nation and Region tables, which were stored on an internal drive. A backup of the database was taken to a separate array for drives than those used for the database.

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

1. The complete database was backed up once to a software RAID-5 partition on the data drives.
2. Six streams of ACID transactions were started. Each stream executed a minimum of 100 transactions.
3. One physical drive of a RAID-0 data volume was removed.
4. The six streams of ACID transactions failed and recorded their number of committed transactions in success files.
5. The failed disk was replaced with a new drive.
6. The system was rebooted. The database data partitions containing the failed disk were recreated.
7. The RAID-5 array containing the backup was resynchronized.
8. A database restore was issued using the backup taken at the beginning of this test.
9. A command was issued causing the database to run through its roll-forward recovery.
10. The counts in the success files and the HISTORY table count were compared and were found to match.

4.4.2 Loss of Log and System Crash

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.

1. Six streams of ACID transactions were started. Each stream executed a minimum of 100 transactions.
2. While the test was running, one of the disks from the database RAID-1 log was removed.
3. The test continued running for approximately an additional 30 transactions per stream.
4. Then the system was powered off.
5. When the power was restored, the system rebooted.
6. The database log disk was replaced and a rebuild function was initiated to restore the RAID-1 log array to its protected status. The rebuild completed successfully.

7. When the database was started, it went through a recovery period.
8. The success file and the HISTORY table counts were compared. One more transaction was in the History table than was in the success file. This is known as an “in-flight” transaction.

4.4.3 System Crash

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

This test was combined with the Loss of Log test. See the previous section.

4.4.4 Memory Failure

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

See the previous section.

5 Clause 4: Scaling and Database Population Related Items

5.1 Cardinality of Tables

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

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

5.2 Distribution of Tables and Logs

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

The following series of tables shows the distribution of tables and logs across all media.

Controller	Drives	Logical Node/Partition	Size	Use
Internal Serve-RAID 7k	2 - 36.4GB RAID 1	/dev/sda1	100MB	/boot swap Linux Operating System DB2, TPC-H Kit DB Log
		/dev/sda2	4GB	
		/dev/sda3	6GB	
		/dev/sda4	23.8GB	
	3 - 36.4GB RAID-0	/dev/sdb1	15.8GB	Temp Tables DB Data Temp Tables Backup (SW RAID-5) RF Data; DB Path
		/dev/sdb2	24.2GB	
		/dev/sdb3	15.8GB	
		/dev/sdb4	45.8GB	
	3 - 36.4GB RAID-0	/dev/sdc1	15.8GB	Temp Data DB Data Temp Data Backup (SW RAID-5) RF Data; DB Path
		/dev/sdc2	24.2GB	
		/dev/sdc3	15.8GB	
		/dev/sdc4	45.8GB	
	3 - 36.4GB RAID-0	/dev/sdd1	15.8GB	Temp Data DB Data Temp Data Backup (SW RAID-5) RF Data; DB Path
		/dev/sdd2	24.2GB	
		/dev/sdd3	15.8GB	
		/dev/sdd4	45.8GB	
	3 - 36.4GB RAID-0	dev/sde1	15.8GB	Temp Data DB Data Temp Data Backup(SW RAID-5) RF Data; DB Path
		/dev/sde2	24.2GB	
		/dev/sde3	15.8GB	
		/dev/sde4	45.8GB	
	3 - 36.4GB RAID-0	/dev/sdf1	15.8GB	Temp Data DB Data Temp Data Backup(SW RAID-5) RF Data; DB Path
		/dev/sdf2	24.2GB	
		/dev/sdf3	15.8GB	
		/dev/sdf4	45.8GB	
	3 - 36.4GB RAID-0	dev/sdg1	15.8GB	Temp Data DB Data Temp Data Backup (SW RAID-5) RF Data; DB Path
		/dev/sdg2	24.2GB	
		/dev/sdg3	15.8GB	
		/dev/sdg4	45.8GB	
QLogic 2342	12 - 36.4GB RAID-5	/dev/sdh1	Unused	Flatfile Data. Not Priced
		/dev/sdh2	Unused	
		/dev/sdh3	125GB	

The priced configuration used 20 disks. An additional 12 disks were used to store the 100GB database files and were not priced.

5.3 Database Partition / Replication Mapping

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

The database was not replicated or partitioned.

5.4 RAID Implementation

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

RAID-1 was used for log disks. RAID-0 was used for all other database disks and the temporary tablespace.

5.5 DBGEN Modifications

Any modifications to the DBGEN (see Clause 4.2.1) source code must be disclosed. In the event that a program other than DBGEN was used to populate the database, it must be disclosed in its entirety.

The standard distribution DBGEN version 1.3.0 was used for database population. No modifications were made.

5.6 Database Load Time

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

See the Executive Summary at the beginning of this report.

5.7 Data Storage Ratio

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

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

Disk Type	Number of Disks	Formatted Space per Disk	Total Disk Space	Scale Factor	Storage Ratio
36.4GB 15K Ultra320 SCSI Drive	20	33.9GB	678GB	100GB	6.78

The data storage ratio is 6.78, derived by dividing 678GB by the database size of 100GB.

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 tables were loaded as depicted in Figure 4-1.

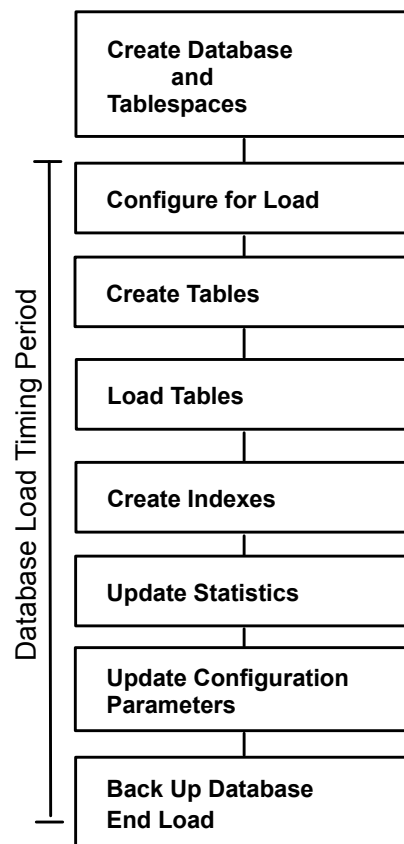


Figure 4-1. Database Load Procedure

5.9 Qualification Database Configuration

Any differences between the configuration of the qualification database and the test database must be disclosed.

The qualification database used identical scripts and disk structure to create and load the data with the exception of the database load script that was modified to point to the Qualification database flatfile data.

6 Clause 5: Performance Metrics and Execution Rules Related Items

6.1 System Activity between Load and Performance Tests

Any system activity on the SUT that takes place between the conclusion of the load test and the beginning of the performance test must be fully disclosed.

The auditor requested that queries be run against the database to verify the correctness of the database load.

6.2 Steps in the Power Test

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

The following steps were used to implement the power test:

1. RF1 Refresh Transaction
2. Stream 00 Execution
3. RF2 Refresh Transaction

6.3 Timing Intervals for Each Query and Refresh Function

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

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report.

6.4 Number of Streams for the Throughput Test

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

Six streams were used for the throughput test.

6.5 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.

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report.

6.6 Total Elapsed Time for the Measurement Interval

The total elapsed time for the measurement interval must be reported for the throughput test.

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report..

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

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

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report.

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

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

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report.

6.9 Performance Metrics

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

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report.

6.10 Performance Metric and Numerical Quantities from Both Runs

The performance metric and numerical quantities from both runs must be disclosed.

Two consecutive runs of the TPC-H benchmark were performed. The following table contains the results for both runs.

	QppH @ 100GB	QthH @ 100GB	QphH @ 100GB
Run1	2,277.7	1,575.2	1,894.2
Run2	2,284.0	1,595.2	1,908.8

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.

DB2 was restarted between runs.

7 Clause 6: SUT and Driver Implementation Related Items

7.1 Driver

A detailed textual description of how the driver performs its functions, how its various components interact and any product functionality or environmental setting on which it relies must be provided. All related source code, scripts and configurations must be disclosed. The information provided should be sufficient for an independent reconstruction of the driver.

Appendix D, “Driver Source Code,” contains the source code used for the driver and all scripts used in connection with it.

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

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

7.2 Implementation-Specific Layer

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

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

A separate instance of tpcdbatch is invoked for each stream. Each instance establishes a distinct connection to the database server through which the EQT is transmitted to the database and the results are returned through the implementation specific layer to the driver. When an instance of tpcdbatch is invoked, it is provided with a context of whether it is running a power test, query stream or refresh stream, as well as an input file containing the 22 queries and/or refresh functions. tpcdbatch then connects to the database, performs any session initialization as well as preparing output files required by the auditor. Then it proceeds to read from the input file and processes each query or refresh function in turn.

For queries, each query is prepared, described, and a cursor is opened and used to fetch the required number of rows. After the last row has been retrieved a commit is issued. For the refresh functions, during the database build all data is first split for each node using the db2split utility. 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 tpcdbatch encounters a call to execute RF1, it first calls a shell script which loads these n sets of data into n sets of temporary tables (one each for lineitem and orders). Then tpcdbatch forks off n children to do an insert with subselect into the original lineitem and orders tables. When tpcdbatch encounters a call to execute RF2, it calls a shell script that loads these data into a single staging table. Then tpcdbatch forks off p children (where $p * x = m$) to do x sets of deletes from the orders and lineitem tables with a subselect from the staging table.

7.3 Profile-Directed Optimization

Profile-directed optimization was not used.

8 Clause 7: Pricing Related Items

8.1 Hardware and Software Components

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

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.

8.2 Three-Year Cost of System Configuration

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 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.

8.3 Availability Dates

The committed delivery date for general availability (availability date) of products used in the price calculations must be reported. When the priced system includes products with different availability dates, availability date reported on the Executive Summary must be the date by which all components are committed to being available. The Full Disclosure Report must report availability dates individually for at least each of the categories for which a pricing subtotal must be provided (see Clause 7.3.1.3).

The total solution as priced is available December 14, 2004.

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.

Clause 9: Audit Related Items

9.1 Auditor's 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 Francois Raab of InfoSizing, Inc. For a copy of this disclosure, go to www.tpc.org.

Benchmark Sponsor:	Haider Rizvi Mgr., DB2 Data Warehouse Performance IBM Canada Ltd; 8200 Warden Avenue Markham, Ontario L6G 1C7	Celia Schreiber Manager, xSeries Performance Benchmarks and Analysis IBM Systems Group 3039 Cornwallis Road Research Triangle Park, NC 27709
--------------------	--	---

December 13, 2004

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

Platform:	IBM® @server® xSeries® 346
Database Manager:	IBM DB2® UDB 8.2 Express Edition
Operating System:	SUSE Linux Enterprise Server 9

The results were:

CPU (Speed)	Memory	Disks	QphH@100GB
IBM @server xSeries 346			
2 x Intel Xeon (3.6 GHz)	1 MB L2 Cache 4 GB Main	20 x 36.4 GB uSCSI	1,894.2

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 100GB and populated accordingly
- The compliance of the database auxiliary data structures was verified
- The database load time was correctly measured and reported

- The required ACID properties were verified and met
- The query input variables were generated by QGEN
- The query text was produced using minor modifications and an approved query variant
- 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 5 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

Additional Audit Notes:

None.

Respectfully Yours,

A handwritten signature in black ink, appearing to read "François Raab", with a long horizontal flourish extending to the right.

François Raab
President

Appendix A: Tunable Parameters and System Configuration

DB2 UDB 8.2 Database Manager Configuration

get database manager configuration

Database Manager Configuration

Node type = Enterprise Server Edition with local and remote clients

Database manager configuration release level = 0x0a00

CPU speed (millisec/instruction) (CPUSPEED) = 2.676617e-07
Communications bandwidth (MB/sec) (COMM_BANDWIDTH) = 1.000000e+02

Max number of concurrently active databases (NUMDB) = 1
Data Links support (DATA LINKS) = NO
Federated Database System Support (FEDERATED) = NO
Transaction processor monitor name (TP_MON_NAME) =

Default charge-back account (DFT_ACCOUNT_STR) =

Java Development Kit installation path (JDK_PATH) = /opt/IBMJava2-141

Diagnostic error capture level (DIAGLEVEL) = 0
Notify Level (NOTIFYLEVEL) = 0
Diagnostic data directory path (DIAGPATH) =

Default database monitor switches
Buffer pool (DFT_MON_BUFPOOL) = OFF
Lock (DFT_MON_LOCK) = OFF
Sort (DFT_MON_SORT) = OFF
Statement (DFT_MON_STMT) = OFF
Table (DFT_MON_TABLE) = OFF
Timestamp (DFT_MON_TIMESTAMP) = ON
Unit of work (DFT_MON_UOW) = OFF
Monitor health of instance and databases (HEALTH_MON) = OFF

SYSADM group name (SYSADM_GROUP) =
SYSCTRL group name (SYSCTRL_GROUP) =
SYSMAINT group name (SYSMAINT_GROUP) =
SYSMON group name (SYSMON_GROUP) =

Client Userid-Password Plugin (CLNT_PW_PLUGIN) =
Client Kerberos Plugin (CLNT_KRB_PLUGIN) =
Group Plugin (GROUP_PLUGIN) =
GSS Plugin for Local Authorization (LOCAL_GSSPLUGIN) =
Server Plugin Mode (SRV_PLUGIN_MODE) = UNFENCED
Server List of GSS Plugins (SRVCON_GSSPLUGIN_LIST) =
Server Userid-Password Plugin (SRVCON_PW_PLUGIN) =
Server Connection Authentication (SRVCON_AUTH) = NOT_SPECIFIED
Database manager authentication (AUTHENTICATION) = SERVER
Cataloging allowed without authority (CATALOG_NOAUTH) = NO
Trust all clients (TRUST_ALLCLNTS) = YES
Trusted client authentication (TRUST_CLNTAUTH) = CLIENT
Bypass federated authentication (FED_NOAUTH) = NO

Default database path (DFTDBPATH) = /home/tpch

Database monitor heap size (4KB) (MON_HEAP_SZ) = 90
Java Virtual Machine heap size (4KB) (JAVA_HEAP_SZ) = 1024
Audit buffer size (4KB) (AUDIT_BUF_SZ) = 0
Size of instance shared memory (4KB) (INSTANCE_MEMORY) = AUTOMATIC
Backup buffer default size (4KB) (BACKBUFSZ) = 1024
Restore buffer default size (4KB) (RESTBUFSZ) = 1024

Sort heap threshold (4KB) (SHEAPTHRES) = 144000

Directory cache support (DIR_CACHE) = YES

Application support layer heap size (4KB) (ASLHEAPSZ) = 15
Max requester I/O block size (bytes) (RQIOBLK) = 32767
Query heap size (4KB) (QUERY_HEAP_SZ) = 1000

Workload impact by throttled utilities (UTIL_IMPACT_LIM) = 10

Priority of agents (AGENTPRI) = SYSTEM
Max number of existing agents (MAXAGENTS) = 400
Agent pool size (NUM_POOLAGENTS) = 8
Initial number of agents in pool (NUM_INITAGENTS) = 4
Max number of coordinating agents (MAX_COORDAGENTS) = (MAXAGENTS - NUM_INITAGENTS)
Max no. of concurrent coordinating agents (MAXCAGENTS) = MAX_COORDAGENTS
Max number of client connections (MAX_CONNECTIONS) = MAX_COORDAGENTS

Keep fenced process (KEEPFENCED) = YES
Number of pooled fenced processes (FENCED_POOL) = MAX_COORDAGENTS
Initial number of fenced processes (NUM_INITFENCED) = 0

Index re-creation time and redo index build (INDEXREC) = RESTART

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

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

TCP/IP Service name (SVCENAME) =
Discovery mode (DISCOVER) = SEARCH
Discover server instance (DISCOVER_INST) = ENABLE

Maximum query degree of parallelism (MAX_QUERYDEGREE) = 8
Enable intra-partition parallelism (INTRA_PARALLEL) = YES

No. of int. communication buffers (4KB) (FCM_NUM_BUFFERS) = 512
Number of FCM request blocks (FCM_NUM_RQB) = AUTOMATIC
Number of FCM connection entries (FCM_NUM_CONNECT) = AUTOMATIC
Number of FCM message anchors (FCM_NUM_ANCHORS) = AUTOMATIC

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

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

DB2 UDB 8.2 Database Configuration

get database configuration for TPCD

Database Configuration for Database TPCD

Database configuration release level = 0x0a00
Database release level = 0x0a00

Database territory = US
Database code page = 819
Database code set = ISO8859-1
Database country/region code = 1
Database collating sequence = BINARY
Alternate collating sequence (ALT_COLLATE) =

Dynamic SQL Query management (DYN_QUERY_MGMT) = DISABLE

Discovery support for this database (DISCOVER_DB) = ENABLE

Default query optimization class (DFT_QUERYOPT) = 7
Degree of parallelism (DFT_DEGREE) = 6

Continue upon arithmetic exceptions (DFT_SQLMATHWARN) = NO
 Default refresh age (DFT_REFRESH_AGE) = 0
 Default maintained table types for opt (DFT_MTTB_TYPES) = SYSTEM
 Number of frequent values retained (NUM_FREQVALUES) = 0
 Number of quantiles retained (NUM_QUANTILES) = 300

Backup pending = NO

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

Multi-page file allocation enabled = YES

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

Data Links Token Expiry Interval (sec) (DL_EXPINT) = 60
 Data Links Write Token Init Expiry Intvl(DL_WT_IEXPINT) = 60
 Data Links Number of Copies (DL_NUM_COPIES) = 1
 Data Links Time after Drop (days) (DL_TIME_DROP) = 1
 Data Links Token in Uppercase (DL_UPPER) = NO
 Data Links Token Algorithm (DL_TOKEN) = MAC0

Database heap (4KB) (DBHEAP) = 6000
 Size of database shared memory (4KB) (DATABASE_MEMORY) = AUTOMATIC
 Catalog cache size (4KB) (CATALOGCACHE_SZ) = 64
 Log buffer size (4KB) (LOGBUFSZ) = 64
 Utilities heap size (4KB) (UTIL_HEAP_SZ) = 5000
 Buffer pool size (pages) (BUFFPAGE) = 160000
 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) = 100000

Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 30000
 Percent of mem for appl. group heap (GROUPHEAP_RATIO) = 70
 Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 2048

Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = (SHEAPTHRES)
 Sort list heap (4KB) (SORTHEAP) = 14000
 SQL statement heap (4KB) (STMTHEAP) = 4096
 Default application heap (4KB) (APPLHEAPSZ) = 256
 Package cache size (4KB) (PCKCACHESZ) = (MAXAPPLS*8)
 Statistics heap size (4KB) (STAT_HEAP_SZ) = 4384

Interval for checking deadlock (ms) (DLCHKTIME) = 10000
 Percent. of lock lists per application (MAXLOCKS) = 60
 Lock timeout (sec) (LOCKTIMEOUT) = -1

Changed pages threshold (CHNGPGS_THRESH) = 60
 Number of asynchronous page cleaners (NUM_IOCLEANERS) = 6
 Number of I/O servers (NUM_IOSERVERS) = 12
 Index sort flag (INDEXSORT) = YES
 Sequential detect flag (SEQDETECT) = YES
 Default prefetch size (pages) (DFT_PREFETCH_SZ) = AUTOMATIC

Track modified pages (TRACKMOD) = OFF

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

Max number of active applications (MAXAPPLS) = 192
 Average number of active applications (AVG_APPLS) = 1
 Max DB files open per application (MAXFILOP) = 1024

Log file size (4KB) (LOGFILSIZ) = 100000
 Number of primary log files (LOGPRIMARY) = 20
 Number of secondary log files (LOGSECOND) = 2
 Changed path to log files (NEWLOGPATH) =
 Path to log files = /dev/raw/raw1
 Overflow log path (OVERFLOWLOGPATH) =
 Mirror log path (MIRRORLOGPATH) =
 First active log file = S00000005.LOG
 Block log on disk full (BLK_LOG_DSK_FUL) = NO

Percent of max active log space by transaction(MAX_LOG) = 0
 Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0

Group commit count (MINCOMMIT) = 1
 Percent log file reclaimed before soft ckcpt (SOFTMAX) = 1600
 Log retain for recovery enabled (LOGRETAIN) = RECOVERY
 User exit for logging enabled (USEREXIT) = OFF

HADR database role = STANDARD
 HADR local host name (HADR_LOCAL_HOST) =
 HADR local service name (HADR_LOCAL_SVC) =
 HADR remote host name (HADR_REMOTE_HOST) =
 HADR remote service name (HADR_REMOTE_SVC) =
 HADR instance name of remote server (HADR_REMOTE_INST) =
 HADR timeout value (HADR_TIMEOUT) = 120
 HADR log write synchronization mode (HADR_SYNCMODE) = NEARSYNC

First log archive method (LOGARCHMETH1) = LOGRETAIN
 Options for logarchmeth1 (LOGARCHOPT1) =
 Second log archive method (LOGARCHMETH2) = OFF
 Options for logarchmeth2 (LOGARCHOPT2) =
 Failover log archive path (FAILARCHPATH) =
 Number of log archive retries on error (NUMARCHRETRY) = 5
 Log archive retry Delay (secs) (ARCHRETRYDELAY) = 20
 Vendor options (VENDOROPT) =

Auto restart enabled (AUTORESTART) = ON
 Index re-creation time and redo index build (INDEXREC) = SYSTEM (RESTART)
 Log pages during index build (LOGINDEXBUILD) = OFF
 Default number of loadrec sessions (DFT_LOADREC_SES) = 1
 Number of database backups to retain (NUM_DB_BACKUPS) = 12
 Recovery history retention (days) (REC_HIS_RETENTN) = 366

TSM management class (TSM_MGMTCLASS) =
 TSM node name (TSM_NODENAME) =
 TSM owner (TSM_OWNER) =
 TSM password (TSM_PASSWORD) =

Automatic maintenance (AUTO_MAINT) = OFF
 Automatic database backup (AUTO_DB_BACKUP) = OFF
 Automatic table maintenance (AUTO_TBL_MAINT) = OFF
 Automatic runstats (AUTO_RUNSTATS) = OFF
 Automatic statistics profiling (AUTO_STATS_PROF) = OFF
 Automatic profile updates (AUTO_PROF_UPD) = OFF
 Automatic reorganization (AUTO_REORG) = OFF

DB2 Version

Database and Database manager configuration taken at : Sun Dec 5 14:07:08 EST 2004
 DB21085I Instance "tpch" uses "64" bits and DB2 code release "SQL08020" with level identifier "03010106".
 Informational tokens are "DB2 v8.1.1.64", "s040812", "MI00089", and FixPak "7".
 Product is installed at "/opt/IBM/db2/V8.1".

DB2 Registry Variables

DB2_ARAM_DISCOVERY=-1
 DB2NOLIOAIO=no
 DB2_EXTENDED_OPTIMIZATION=y
 DB2_ANTIJOIN=Y
 DB2_LIKE_VARCHAR=Y,Y
 DB2_CORRELATED_PREDICATES=Y
 DB2BPVARS=/home/tpch/tpcd/custom/diskio/bpvars
 DB2_VECTOR=Y
 DB2_HASH_JOIN=Y
 DB2OPTIONS=-t -v +c
 DB2_PARALLEL_IO=*

SUSE Linux Version

Linux x346 2.6.5-7.97-smp #1 SMP Fri Jul 2 14:21:59 UTC 2004 x86_64 x86_64
x86_64 GNU/Linux

SUSE Linux Configuration Parameters

/etc/SuSEconfig/profile

Line 34 LANG="en_US"

Appendix B: Database Build Scripts

bpvars

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

buildtpcd

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

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

# get TPC-D specific environment variables
require "getvars";
require "macro.pl";
require "tpcdmacro.pl";
require "version";
$timestamp=`perl gettimestamp "short"`;

# Make output unbuffered.
open(STDOUT, "| tee buildtpcd.out.${timestamp}");
select(STDOUT);
$|= 1 ;

#-----#
# verify that necessary environment variables for building the database  #
# are present.  Default those that aren't necessary                      #
#-----#

# variables that must be specified for script to run
@reqVars = ("TPCD_PLATFORM",
            "TPCD_PRODUCT",
            "TPCD_VERSION",
            "TPCD_DBNAME",
            "TPCD_MODE",
            "TPCD_SF",
            "TPCD_DDLPATH",
            "TPCD_AUDIT",
            "TPCD_AUDIT_DIR",
            "TPCD_BUILD_STAGE");

# variables default to 'NULL' if unspecified
@defNullVars = ("TPCD_LOAD_SCRIPT",
               "TPCD_LOAD_SCRIPT_QUAL",
               "TPCD_INPUT",
               "TPCD_QUAL_INPUT",
               "TPCD_DBGEN",
               "TPCD_LOGPRIMARY",
               "TPCD_LOGSECOND",
               "TPCD_LOGFILSIZ",
               "TPCD_LOG_DIR",
               "TPCD_MACHINE",
               "TPCD_AGENTPRI",
               "TPCD_STAGING_TABLE_DDL",
               "TPCD_PRELOAD_STAGING_TABLE_SCRIPT",
               "TPCD_DELETE_STAGING_TABLE_SQL",
               "TPCD_RUNSTATSHORT",
               "TPCD_ADD_RI",
               "TPCD_AST",
               "TPCD_DBM_CONFIG",
               "TPCD_EXPLAIN_DDL",
```

```
"TPCD_NODEGROUP_DEF",
"TPCD_BUFFERPOOL_DEF",
"TPCD_LOAD_DB2SET_SCRIPT",
"TPCD_DB2SET_SCRIPT",
"TPCD_LOG_DIR_SETUP_SCRIPT",
"TPCD_LOAD_CONFIGFILE",
"TPCD_LOAD_DBM_CONFIGFILE",
"TPCD_TEMP");
```

```
&setVar(@reqVars, "ERROR");
&setVar(@defNullVars, "NULL");
```

```
if ( $qual eq "QUAL" ){
    @reqQualVars = ("TPCD_QUAL_DBNAME",
                   "TPCD_QUAL_DDL",
                   "TPCD_QUAL_TBSP_DDL",
                   "TPCD_QUALCONFIGFILE",
                   "TPCD_DBM_QUALCONFIG",
                   "TPCD_LOAD_QUALCONFIGFILE",
                   "TPCD_LOAD_DBM_QUALCONFIGFILE");
```

```
&setVar(@reqQualVars, "ERROR");
```

```
if ( ($ENV{"TPCD_QUAL_INPUT"}) eq "NULL" ){
    if (((($ENV{"TPCD_DBGEN"}) eq "NULL") ||
          (($ENV{"TPCD_TEMP"}) eq "NULL"))){
        die "TPCD_DBGEN and TPCD_TEMP must be set if flatfiles are not
provided.\n";
    }
}
}
```

```
$platform=$ENV{"TPCD_PLATFORM"};
```

```
if (length($ENV{"TPCD_DBPATH"}) <= 0){
    # if no db pathname specified, build the db in the home directory
    if ( $platform eq "aix" ||
        $platform eq "sun" ||
        $platform eq "ptx" ||
        $platform eq "hp" ||
        $platform eq "linux"){
        $ENV{"TPCD_DBPATH"} = $ENV{"HOME"};
    }
    elsif ( $platform eq "nt" ){
        $ENV{"TPCD_DBPATH"} = $ENV{"HOMEDRIVE"};
    }
    else{
        die "platform '$platform' not supported yet\n";
    }
}
if ( ($ENV{"TPCD_INPUT"}) eq "NULL" ){
    if (((($ENV{"TPCD_DBGEN"}) eq "NULL") ||
          (($ENV{"TPCD_TEMP"}) eq "NULL"))){
        die "TPCD_DBGEN and TPCD_TEMP must be set if flatfiles are not
provided.\n";
    }
}
#-----#
# ddl script files found under custom directory                        #
#-----#

if (length($ENV{"TPCD_DDL"}) <= 0){
    $ENV{"TPCD_DDL"} = "dss.ddl";
}
if (length($ENV{"TPCD_TBSP_DDL"}) <= 0){
    $ENV{"TPCD_TBSP_DDL"} = "dss.tbsp.ddl";
}
if (length($ENV{"TPCD_INDEXDDL"}) <= 0){
    $ENV{"TPCD_INDEXDDL"} = "dss.index";
}
if (length($ENV{"TPCD_RUNSTATS"}) <= 0){
```

```

$ENV{"TPCD_RUNSTATS"} = "dss.runstats";
}
if (length($ENV{"TPCD_CONFIGFILE"}) <= 0){
    $ENV{"TPCD_CONFIGFILE"} = "dss.dbconfig";
}

#-----#
# other settings                                     #
#-----#

if (length($ENV{"TPCD_BACKUP_DIR"}) <= 0){
    $ENV{"TPCD_BACKUP_DIR"} = "${delim}dev${delim}null";
}
if (length($ENV{"TPCD_COPY_DIR"}) <= 0){
    $ENV{"TPCD_COPY_DIR"} = "${delim}dev${delim}null";
}
if (length($ENV{"TPCD_TEMP"}) <= 0){
    $ENV{"TPCD_TEMP"} = "/u/$instance/sqllib/tmp";
}
if (length($ENV{"TPCD_PHYS_NODE"}) <= 0){
    $ENV{"TPCD_NODEGROUP_DEF"}="NULL"
}
if (length($ENV{"TPCD_GENERATE_SEED_FILE"}) <= 0){
    $ENV{"TPCD_GENERATE_SEED_FILE"} = "no";
}
if (length($ENV{"TPCD_SORTBUF"}) <= 0){
    $ENV{"TPCD_SORTBUF"} = 4096;
}
if (length($ENV{"TPCD_LOAD_PARALLELISM"}) <= 0){
    $ENV{"TPCD_LOAD_PARALLELISM"} = 0;
}
if (length($ENV{"TPCD_LOADSTATS"}) <= 0){
    $ENV{"TPCD_LOADSTATS"} = "no";
}
if (length($ENV{"TPCD_FASTPARSE"}) <= 0){
    $ENV{"TPCD_FASTPARSE"} = "no";
}
if (length($ENV{"TPCD_LOG"}) <= 0){
    $ENV{"TPCD_LOG"} = "no";
}
if (length($ENV{"TPCD_SMPDEGREE"}) <= 0){
    $ENV{"TPCD_SMPDEGREE"} = 1;
}
if (length($ENV{"TPCD_ACTIVATE"}) <= 0){
    $ENV{"TPCD_ACTIVATE"} = "no";
}
if (length($ENV{"TPCD_APPEND_ON"}) <= 0){
    $ENV{"TPCD_APPEND_ON"}="yes"
}
if (length($ENV{"TPCD_GENERATE_SEED_FILE"}) <= 0){
    $ENV{"TPCD_GENERATE_SEED_FILE"}="no";
}

#setup global variables
$tpcdVersion=      $ENV{"TPCD_VERSION"};
$buildStage=      $ENV{"TPCD_BUILD_STAGE"};
$mode=            $ENV{"TPCD_MODE"};
$delim =          $ENV{"TPCD_PATH_DELIM"};
$sep =            $ENV{"COMMAND_SEP"};
$dddlpath=        $ENV{"TPCD_DDLPATH"};
$extraindex=      $ENV{"TPCD_EXTRAINDEX"};
$earlyindex=      $ENV{"TPCD_EARLYINDEX"};
$loadstats=       $ENV{"TPCD_LOADSTATS"};
$addRI=           $ENV{"TPCD_ADD_RI"};
$astFile=         $ENV{"TPCD_AST"};
$genSeed=         $ENV{"TPCD_GENERATE_SEED_FILE"};
$log=             $ENV{"TPCD_LOG"};
$activate=        $ENV{"TPCD_ACTIVATE"};
$RealAudit=       $ENV{"TPCD_AUDIT"};
$auditDir=        $ENV{"TPCD_AUDIT_DIR"};
$loadsetScript=   $ENV{"TPCD_LOAD_LOAD_DB2SET_SCRIPT"};
$user=            $ENV{"USER"};

```

```

$logDirScript=     $ENV{"TPCD_LOG_DIR_SETUP_SCRIPT"};
$logprimary=       $ENV{"TPCD_LOGPRIMARY"};
$logsecond=        $ENV{"TPCD_LOGSECOND"};
$logfilsize=       $ENV{"TPCD_LOGFILSIZE"};
$dbpath =          $ENV{"TPCD_DBPATH"};
$explainDDL=       $ENV{"TPCD_EXPLAIN_DDL"};
$platform=         $ENV{"TPCD_PLATFORM"};
$buffpooldef=      $ENV{"TPCD_BUFFERPOOL_DEF"};
$stagingTbl =      $ENV{"TPCD_STAGING_TABLE_DDL"};
$preloadSampleUF=  $ENV{"TPCD_PRELOAD_STAGING_TABLE_SCRIPT"};
$deleteSampleUF=   $ENV{"TPCD_DELETE_STAGING_TABLE_SQL"};
$machine=          $ENV{"TPCD_MACHINE"};
$runstatShort =    $ENV{"TPCD_RUNSTATSHORT"};
$runstats =        $ENV{"TPCD_RUNSTATS"};
$smpdegree =       $ENV{"TPCD_SMPDEGREE"};
$agentpri =        $ENV{"TPCD_AGENTPRI"};
$setScript =       $ENV{"TPCD_DB2SET_SCRIPT"};
$backupdir =       $ENV{"TPCD_BACKUP_DIR"};
$nodegroupdef=     $ENV{"TPCD_NODEGROUP_DEF"};
$dbgen=            $ENV{"TPCD_DBGEN"};
$appendOn=         $ENV{"TPCD_APPEND_ON"};
$indexddl=         $ENV{"TPCD_INDEXDDL"};

if($qual eq "QUAL"){
    $logDir= $ENV{"TPCD_LOG_QUAL_DIR"};
    $dbname= $ENV{"TPCD_QUAL_DBNAME"};
    $input=  $ENV{"TPCD_QUAL_INPUT"};
    $sf=     $ENV{"TPCD_QUAL_SF"};
    $loadconfigfile=$ENV{"TPCD_LOAD_QUALCONFIGFILE"};
    $loadDBMconfig=
$ENV{"TPCD_LOAD_DBM_QUALCONFIGFILE"};
    $loadscript = $ENV{"TPCD_LOAD_SCRIPT_QUAL"};
    $configfile = $ENV{"TPCD_QUALCONFIGFILE"};
    $dbmconfig = $ENV{"TPCD_DBM_QUALCONFIG"};
    $ddl=        $ENV{"TPCD_QUAL_DDL"};
    $tbspddl=    $ENV{"TPCD_QUAL_TBSP_DDL"};
}
else{
    $logDir= $ENV{"TPCD_LOG_DIR"};
    $dbname= $ENV{"TPCD_DBNAME"};
    $input=  $ENV{"TPCD_INPUT"};
    $sf=     $ENV{"TPCD_SF"};
    $loadconfigfile=$ENV{"TPCD_LOAD_CONFIGFILE"};
    $loadDBMconfig=
$ENV{"TPCD_LOAD_DBM_CONFIGFILE"};
    $loadscript = $ENV{"TPCD_LOAD_SCRIPT"};
    $configfile = $ENV{"TPCD_CONFIGFILE"};
    $dbmconfig = $ENV{"TPCD_DBM_CONFIG"};
    $ddl=        $ENV{"TPCD_DDL"};
    $tbspddl=    $ENV{"TPCD_TBSP_DDL"};
}

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

#-----#
# echo parameter settings to acknowledge what is being built      #
# and set db2set options for database load                        #
#-----#

&printSummary;

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

```

```

if ( $platform eq "nt" ){
    if (( $mode eq "uni" ) || ( $mode eq "smp" )){
        #spaces required for NT
        $src=&dodb_noconn("db2set DB2OPTIONS=\\" -t -v +c\\";db2set
DB2NTNOCACHE=ON",$all_ln);
    }
    else{
        $src=&dodb_noconn("db2set DB2OPTIONS=\\\\" -t -v +c\\\\";db2set
DB2NTNOCACHE=ON",$all_ln);
    }
}
else{
    if (( $mode eq "uni" ) || ( $mode eq "smp" )){
        $src=&dodb_noconn("db2set DB2OPTIONS=\\" -t -v +c\\"",$all_ln);
    }
    else{
        $src=&dodb_noconn("db2set DB2OPTIONS=\\\\" -t -v +c\\\\"",$all_ln);
    }
}
if ( $rc != 0 ){
    die "failure setting db2 environment variable : rc = $rc\n";
}

#-----#
# set the db2 env vars for loading, from the TPCD_LOAD_DB2SET_SCRIPT
script #
#-----#

if ( $loadsetScript ne "NULL" )
{
    if ( $platform eq "nt" ){
        if (( $mode eq "uni" ) || ( $mode eq "smp" )){
            $src=system("$ {ddlpath} $ {delim} $loadsetScript");
        }
        else{
            $src=system(" rah \\" cd $ {ddlpath} & $loadsetScript" ");
        }
    }
    else{
        $src=system("$ {ddlpath} $ {delim} $loadsetScript");
    }
    ($src == 0) || die "failure loading db2set parms from $loadsetScript \n";
}

!&stopStart || die;
#-----#
# Begin complete build: TPCD_BUILDSTAGE = ALL #
#-----#

if($buildStage eq "ALL") {
    #create the database
    $src = &createDb;
    ($src == 0) || die "ERROR: create database failed. rc = $rc\n ";
    &setLog;
};

$src = &setLoadConfig;

#-----#
# Begin build from CreateTablespace or early Indexes #
#-----#

if( $buildStage eq "ALL" ||
    $buildStage eq "CRTTBSP" ||
    ($buildStage eq "INDEX" && $earlyindex eq "yes")){
    !&createNodegroups || print "ERROR: create nodegroups failed.\n";
    !&createBufferPools || print "ERROR: create bufferpools failed.\n";
    &outtime("**** Start of audited Load Time - starting to create tables");
    !&createTablespaces || print "WARNING: create tablespaces error.\n";
    !&createExplainTbls || print "ERROR: create EXPLAIN tables
failed.\n";
    !&createTables || print "ERROR: create tables failed.\n";

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

    # if earlyindex requested, create indexes
    if ( $earlyindex eq "yes" ){
        !&createIndexes("early") || die "ERROR: create early
indexes failed.\n";
    }
    # start the dbgen and load.....call the specific mode for loading
    (uni,smp,mIn)
    !&loadData || die "ERROR: failure during load data\n";

    # remove the update.pair.num file so when setupDir runs, it doesn't
    # hang waiting for an answer on nt
    &rm("$auditDir$ {delim} $dbname.$user.update.pair.num");
    # verify that the audit directory exists
    $filename="$auditDir";
    if (-e $filename){
        # set up the $auditDir/$dbname.$user.update.pair.num file
        # to start at update pair 1

        $filename="$auditDir$ {delim} $dbname.$user.update.pair.num";
    }
    else{
        mkdir (" $auditDir", 0775) || die "cannot mkdir $auditDir";
    }
    print "setting update pair num to 1\n";
    system("echo 1 > $filename");
};

#-----#
# Begin build from Index or Load #
#-----#

if( $buildStage eq "ALL" ||
    $buildStage eq "CRTTBSP" ||
    $buildStage eq "LOAD" ||
    $buildStage eq "INDEX"){

    # if indexes haven't been created, do so now
    if ( $earlyindex ne "yes" ){
        !&createIndexes("normal") || die "ERROR: create indexes
failed.\n";
    }
    if ( $extraindex ne "no" ){
        !&createIndexes("extra") || die "ERROR: create extra
indexes failed.\n";
    }
}; # end create/load/index phase of the build

#-----#
# Begin build from runstats #
#-----#

if( $buildStage eq "ALL" ||
    $buildStage eq "CRTTBSP" ||
    $buildStage eq "LOAD" ||
    $buildStage eq "INDEX" ||
    $buildStage eq "RUNSTATS"){
    # if statistics not gathered on the load, run runstats (we have to run the
    # stats at the same time as the index creation whether it be both during
    load,
    # or after load)
    # We need to run the runstats as well if we have specified an extra
    index file
    # for "after load" indexes
    if (( $loadstats eq "no" ) || ( $earlyindex eq "no" ) || ( $extraindex ne
    "no" )){
        &doRunStats;
    }
};

#-----#
# End build phase: all/load/index/runstats #
#-----#
# Add RI/AST, set run configuration #
#-----#

```

```

if ( $addRI ne "NULL" ){
    &outtime("*** Adding RI constraints started");
    &dodb2file($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");

&setConfiguration;
# if logging is enabled, we must take a backup of the database
if ( $log eq "yes" ){
    &createBackup;
}

# stop and restart the database to get config parms recognized
!&stopStart || die;

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

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

#-----#
# Call buildtpcdbatch to compile tpcdbatch #
#-----#
# - if we are in real audit mode then we have to do a number of things #
#   set up the audit directory structure and the run directory structure #
#   so that once we have completed the buildtpcd, we are ready to run. #
#   first remove any old "update pair number" file so we won't be prompted #
#   doing setupDir. #
# - before we stop the database for the final time #
#   if we are in the real audit mode then run dbtables and dbcheck before #
#   we print out the notice that we're ready to run performance tests #
#   if we are building the qualification database then we'll bind to both #
#   the dbname database and the qualification database #
#-----#

$rc = system("perl buildtpcdbatch $qual");
($rc == 0) || die "buildtpcdbatch failed rc=$rc\n";

if ( $RealAudit eq "yes" ){
    &rm("$auditDir${delim}tools${delim}tpcd.runsetup");
    system("perl setupRun");
    if ( $qual eq "QUAL" ){
        $verifyType="q";
    }
    else {
        $verifyType="t";
    }
    system("perl tablesdb $verifyType");
}

&dodb2file($dbname,"$auditDir${delim}tools${delim}first10rows.sql", $once);
}

#-----#

```

```

# Create Catalog info #
#-----#
$rc = system("perl catinfo.pl b");
($rc == 0) || warn "catinfo failed!!! rc = $rc\n";

$rc=system("db2stop");
($rc == 0) || die "failure during db2stop rc = $rc\n";

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

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

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

#-----#
# Function: setLog #
#-----#
sub setLog {
    # update the log information first
    # set up the log directory before we do any index creation
    my $rc;
    my $setLogs;
    my $setLogString;

    if ($logDirScript ne "NULL"){
        system ("perl $ddlpath${delim}$logDirScript");
    }
    elsif ( $logDir ne "NULL" ){
        &dodb_noconn("db2 update database configuration for
$dbname using newlogpath $logDir", $all_in);
    }
    $setLogs=0;
    $setLogString="";
    if ( $logprimary ne "NULL" ){
        $setLogString.="db2 update db cfg for $dbname using
logprimary $logprimary";
        $setLogs=1;
    }
    if ( $logsecond ne "NULL" ){
        if ( $setLogs != 0 ){
            $setLogString.=" $sep ";
        }
        $setLogString.="db2 update db cfg for $dbname using
logsecond $logsecond";
        $setLogs=1;
    }
    if ( $logfilsiz ne "NULL" ){
        if ( $setLogs != 0 ){
            $setLogString.=" $sep ";
        }
        $setLogString.="db2 update db cfg for $dbname using
logfilsiz $logfilsiz";
        $setLogs=1;
    }
    if ( $setLogs != 0 ){
        $setLogString.=" $sep ";
    }
    $setLogString.="db2 update db cfg for $dbname using logbufsz 128";
    $rc = &dodb_noconn("$setLogString", $all_in);
}

```



```

}

#-----#
# Function: createDb                                #
#-----#
sub createDb{
    &outtime("*** Starting to create the database");
    # setup required variables
    my $src;
    $src = &dodb_noconn("db2 \create database $dbname on $dbpath
collate using identity with 'TPC-D $sf GB'",$once);
    ($src == 0) || return($src);
    # reset the db and dbm configuration before we start
    &dodb_noconn("db2 reset database configuration for
$dbname",$all_in);
    &dodb_conn($dbname,"db2 alter bufferpool ibmdefaultbp size -1 $sep
\
        db2 grant connect on database to public $sep \
        db2 grant dbadm on database to $dbname $sep \
        db2 commit",$once);
    &dodb_noconn("db2 reset database manager configuration",$once);
}

#-----#
# Function: createNodegroups                        #
#-----#
sub createNodegroups{
    &outtime("*** Creating the nodegroups.");
    my $src;
    if ( $nodegroupdef ne "NULL"){
        $src =
&dodb2file($dbname,"$ddlpath${delim}$nodegroupdef",$once);
    }
}

#-----#
# Function: createExplainTbls                      #
#-----#
sub createExplainTbls{
    &outtime("*** Creating the EXPLAIN tables.");
    my $src;
    my $explnPathFile;
    my $shome;
    my $sqlpath;

    if ( $explainDDL ne "NULL"){
        $explnPathFile="$explainDDL";
    }
    else{
        if ( $platform eq "ptx"){
            $shome=$ENV{"HOME"};
            $sqlpath="$shome${delim}sqllib";
        }
        if ( $platform ne "nt"){
            $shome=$ENV{"HOME"};
            $sqlpath="$shome${delim}sqllib";
        }
        else{
            $sqlpath=$ENV{"DB2PATH"};
        }
    }

    $explnPathFile="$sqlpath${delim}misc${delim}EXPLAIN.DDL";
    $src = &dodb_conn($dbname,
    "db2 -tvf $explnPathFile $sep \
    db2 alter table explain_instance locksize table append on $sep \
    db2 alter table explain_statement locksize table append on $sep \
    db2 alter table explain_argument locksize table append on $sep \
    db2 alter table explain_object locksize table append on $sep \
    db2 alter table explain_operator locksize table append on $sep \
    db2 alter table explain_predicate locksize table append on $sep \
    db2 alter table explain_stream locksize table append on",
    $once);
}

```

```

#-----#
# Function: createBufferPools                      #
#-----#
sub createBufferPools{
    my $src;
    &outtime("*** Creating the bufferpools");
    if ( $buffpooldef ne "NULL" ){
        #run the create bufferpool ddl
        $src =
&dodb2file($dbname,"$ddlpath${delim}$buffpooldef",$once);
    }
}

#-----#
# Function: createTablespaces                      #
#-----#
sub createTablespaces{
    &outtime("*** Ready to start creating the tablespaces");
    # setup required variables
    my $src;
    $src = &dodb2file($dbname,"$ddlpath${delim}$tblspddl",$once);
    ($src == 0) || return $src;
    # 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
        $src =
&dodb2file($dbname,"$ddlpath${delim}$stagingTbl",$once);
        ($src == 0) || return $src;
        if ( $qual ne "QUAL" ){
            if ( $preloadSampleUF ne "NULL" ){
                # preload the sample UF data for
statistics gathering
                $src = system ("perl
$ddlpath${delim}$preloadSampleUF");
                #($src == 0) || return $src;
            }
            if ( $deleteSampleUF ne "NULL" ){
                # delete the sample rows now that
stats have been gathered
                $src =
&dodb2file($dbname,"$ddlpath${delim}$deleteSampleUF",$once);
                #($src == 0) || return $src;
            }
        }
    }
}

#-----#
# Function: createTables                          #
#-----#
sub createTables{
    my $src;
    $src = &dodb2file($dbname,"$ddlpath${delim}$ddl",$once);
    ($src == 0) || return $src;
    # update the locksize on the non-updated tables to be table level
locking
    # update the tables for appendmode
    if ($appendOn eq "yes"){
        $src = &dodb_conn($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{
        $src = &dodb_conn($dbname,

```

```

        "db2 alter table tpcd.nation locksize table $sep \
        db2 alter table tpcd.region locksize table $sep \
        db2 alter table tpcd.customer locksize table $sep \
        db2 alter table tpcd.supplier locksize table $sep \
        db2 alter table tpcd.part locksize table $sep \
        db2 alter table tpcd.partsupp locksize table $sep \
        db2 alter table tpcd.lineitem pctfree 0 $sep \
        db2 alter table tpcd.orders pctfree 0",
        $once);
    }
}

#-----#
# Function: createIndexes                                     #
#-----#
sub createIndexes{
    # setup required variables
    local @args = @_;
    my $indexType = @args[0];
    my $src;
    &outtime("**** Starting to create $indexType indexes");
    if( $indexType eq "extra"){
        $src =
&dodb2file($dbname,"$ddlpath${delim}$extraindex",$once);
    }elseif( $indexType eq "early" || $indexType eq "normal"){
        $src =
&dodb2file($dbname,"$ddlpath${delim}$indexddl",$once);
    }
    &outtime("**** Create $indexType index completed");
    return $src;
}

#-----#
# Function: setLoadConfig                                     #
#-----#
sub setLoadConfig{

    &outtime("**** Setting LOAD configuration.");
    my $src;
    my $buffpage;
    my $sortheap;
    my $sheapthres;
    my $util_heap_sz;
    my $ioservers;
    my $ioclnrs=          1;
    my $chngpgs=          60;

    if($loadconfigfile eq "NULL"){
        if( $machine eq "small" ){
            $buffpage = 5000;
            $sortheap = 3000;
            $sheapthres = 8000;
            $util_heap_sz = 5000;
            $ioservers = 6;
        }
        elseif( $machine eq "medium" ){
            $buffpage = 10000;
            $sortheap = 8000;
            $sheapthres = 20000;
            $util_heap_sz = 10000;
            $ioservers = 10;
        }
        elseif( $machine eq "big" ){
            $buffpage = 30000;
            $sortheap = 20000;
            $sheapthres = 50000;
            $util_heap_sz = 30000;
            $ioservers = 20;
        }
        else {
            die "Neither a LOAD config filename nor a
valid machine size has \

```

```

        been specified!\n";
    }
    $src = &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);
    }
    else {
        $src =
&dodb2file_noconn("$ddlpath${delim}$loadconfigfile",$all_in);
    }
    ($src == 0) || return $src;
    if($loadDBMconfig ne "NULL"){
        $src =
&dodb2file_noconn("$ddlpath${delim}$loadDBMconfig",$once);
    }
    else {
        $src = &dodb_noconn("db2 update dbm cfg using
sheapthres $sheapthres",$once);
    }
    ($src == 0) || return $src;
    &dodb_noconn("db2 terminate",$once);
    $src = &stopStart;
    return $src;
}

#-----#
# Function: loadData                                         #
#-----#
sub loadData{
    # start the dbgen and load....call the specific mode for loading
    (uni,smp,mln)
    my $src;
    if(( $mode eq "uni" ) || ( $mode eq "smp" )){
        &outtime("**** Starting the load");
        # call the appropriate dbgen/load for uni/smp
        if( $loadscript eq "NULL"){
            $src = system("perl genloaduni $qual");
            ($src == 0) || print "ERROR: genloaduni failed rc
= $rc\n";
        }
        else {
            $src =
&dodb2file_noconn("$ddlpath${delim}$loadscript",$once);
            ($src == 0) || print "ERROR: load script:
$loadscript failed. rc = $rc\n";
        }
    }
    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"){
            $src = system("perl genloadmpp $qual");
            ($src == 0) || print "ERROR: genloadmpp failed.
rc = $rc\n";
        }
        else {
            system("$ddlpath${delim}$loadscript");
            # $src =
&dodb2file_noconn("$ddlpath${delim}$loadscript $sf");
            # ($src == 0) || print "ERROR: load script
$loadscript failed. rc = $rc\n";
        }
    }
    else {
        print "TPCD_MODE not set to one of uni, smp, mln or
mpp\n";
    }
}

```

```

        Src = 1;
    }
    (Src == 0) || &outtime("*** Load complete");
    return Src;
}

#-----#
# Function: doRunStats                                     #
#-----#
sub doRunStats{
    # if loadstats not gathered, then index stats not gathered either.
    &outtime("*** Runstats started");
    if ( $runstatShort ne "NULL" ){
        # we've specified a second runstats file...This runstats file should do
        # runstats for all table except lineitem. The lineitem runstats command
        # should be left in the main runstats file.
        if ( $platform eq "aix" || $platform eq "sun" || $platform eq "ptx" ){
            print "runstats from $ddlpath${delim}$runstatShort running now\n";
            $rc = system("db2 -tvf \"$ddlpath${delim}$runstatShort\" >
\"$auditDir${delim}tools${delim}runstatShort.out\" & ");
            print "rc from runstatshort=$rc\n";
        }
        elsif ( $platform eq "nt" ){
            system("start db2 -tvf $ddlpath${delim}$runstatShort");
        }
        else
        {
            print "Don't know how to start in background on $platform platform\n";
            print "therefore running runstats serially\n";
            &dodb2file($dbname,"$ddlpath${delim}$runstatShort",$once);
        }
    }
    # run the full runstats, or the remainder of what wasn't put into the short
    # runstats file. You should be sure that this runstats will take longer
    # than the short runstats that is running in the background, otherwise
    # setting the config will happen before this is done.
    &dodb2file($dbname,"$ddlpath${delim}$runstats",$once);
    &outtime("*** Runstats completed");
}

#-----#
# Function: setConfiguration                               #
#-----#
sub setConfiguration{
    my $ret = 0;
    &dodb_noconn("db2 update database configuration for $dbname using
dft_degree $smpdegree",$all_in);
    &dodb_noconn("db2 update database manager configuration using
max_querydegree $smpdegree",$once);
    &dodb2file_noconn("$ddlpath${delim}$configfile",$all_in);
    &dodb2file_noconn("$ddlpath${delim}$dbmconfig",$once);

    if ( $agentpri ne "NULL" ){
        &dodb_noconn("db2 update dbm cfg using AGENTPRI
$agentpri",$once);
    }
    # set the db2 environment variables for running the benchmark
    if ( $setScript ne "NULL" ){
        if ( $platform eq "aix" || $platform eq "sun" || $platform eq "ptx" ){
            $ret=system("$ddlpath${delim}$setScript");
        }
        elsif ( $platform eq "nt" ){
            if (($mode eq "uni" ) || ($mode eq "smp" )){
                $ret = system("perl $ddlpath${delim}$setScript");
            }
            else{
                $ret = system("rah \" cd $ddlpath & $setScript \"");
            }
        }
    }
    #($ret == 0 ) || die "failure setting runtime db2set parms from
$setScript \n";
}

```

```

#-----#
# Function: createBackup                                     #
#-----#
sub createBackup{
    my $rc;
    &dodb_noconn("db2 update database configuration for $dbname using
LOGRETAIN yes",$all_in);
    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_in);

    if ( $qual eq "QUAL" ){
        &outtime("*** Starting the backup");
        if (( $mode eq "mln" ) || ( $mode eq "mpp" )){
            # must back up catalog node first...assume node 00
            $rc=system("db2_all \${}<<<+000< db2 \"backup database
$dbname to $backupdir without prompting\" \");
            ($rc == 0) || print "ERROR: backup of catalog node
failed rc = $rc\n";

            # back up remaining nodes
            $rc=system("db2_all \${}<<<-000< db2 backup database
$dbname to /filesystem/backup/qual without prompting\" ");
            ($rc == 0) || print "ERROR: backup of remaining nodes
failed rc = $rc\n";
        }
        else{
            $rc = &dodb_noconn("db2 backup database $dbname to
/filesystem/backup/qual without prompting",$once);
        }
        ($rc == 0) || &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.
        $rc = &dodb_noconn("db2 backup database tpcd to
/filesystem/backup/test with 16 BUFFERS PARALLELISM 8 without
prompting",$once);
        $rc = &dodb_noconn("db2 update db cfg for tpcd using
newlogpath /dev/raw/raw1 logfilsiz 100000 logprimary 20 softmax 1600",
$once);
    }
    return $rc;
}

#-----#
# Function: printSummary                                     #
#-----#
sub printSummary{
    if ( $buildStage ne "ALL" ){
        print " ***** STARTING the build process at the $buildStage Stage
*****\n";
    }
    print "Building a TPC-D Version $tpcdVersion $sf GB database on
$dbbpath with: \n";
    print " Mode = $mode \n";
}

```

```

print " Tablespace ddl in $ddlpath${delim}$tblspddl \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 parameters taken from
$ddlpath${delim}$dss.dbconfig${sfReal}GB \n";
    $configfile="dss.dbconfig${sfReal}GB";
}

```

```

}
if ( $dbmconfig ne "NULL" ){
    print " Database Manager Configuration parameters taken from
$ddlpath${delim}$dbmconfig \n";
}
else {
    print " Database Manager Configuration parameters taken from
$ddlpath${delim}$dss.dbmconfig${sfReal}GB \n";
    $configfile="dss.dbmconfig${sfReal}GB";
}
#print " Copy image for load command created in $copydir \n";
if ( $log eq "yes" ){
    print " Backup files placed in $backupdir \n";
}
else {
    print " No backup will be taken. \n";
}
print " Log retain set to $log \n";
if ( $logDir eq "NULL" ){
    print " Log files remain in database path \n";
}
else {
    print " Log file path set to $logDir \n";
}
if ( $logprimary eq "NULL" ){
    print " Log Primary left at default \n";
}
else {
    print " Log Primary set to $logprimary \n";
}
if ( $logsecond eq "NULL" ){
    print " Log Second left at default \n";
}
else {
    print " Log second set to $logsecond \n";
}
if ( $logfilsiz eq "NULL" ){
    print " Logfilsiz left at default \n";
}
else {
    print " Logfilsiz set to $logfilsiz \n";
}
if ( ($loadconfigfile eq "") || ($loadconfigfile eq "NULL") ){
    print " Machine size set to $machine so the following
configuration \n";
    print " parameters are used for load, create index and runstats: \n";
    print " BUFFPAGE = $buffpage \n";
    print " SORTHEAP = $sortheap \n";
    print " SHEAPTHRES = $sheapthres \n";
    print " NUM_IOSERVERS = $ioservers \n";
    print " NUM_IOCLEANERS = $ioclnrs \n";
    print " CHNGPGS_THRESH = $chngpgs \n";
    print " UTIL_HEAP_SZ = $util_heap_sz \n";
    print " Degree of parallelism (dft_degree and max_querydegree)
set to $smpdegree \n";
    print " Parameters for load are: temp file = $ldtemp \n";
    print " sort buf = $sortbuf \n";
    print " ld parallelism = $load_parallelism \n";
    if ( $fparse eq "yes" ){
        print " FASTPARSE used on load \n";
    }
}
if ( $loadscript ne "NULL" ){
    print " Load commands in $ddlpath${delim}$loadscript \n";
}
print " Degree of parallelism (dft_degree and max_querydegree) set
to $smpdegree \n";
if ( $agentpri ne "NULL" ){
    print " AGENTPRI set to $agentpri \n";
}
if ( $activate eq "yes" ){
    print " Database will be activated when build is complete \n";
}
if ( $explainDDL ne "NULL" ){

```

```

        print "  EXPLAIN DDL being used from
$ddlpath${delim}$explainDDL\n";
    }
    else {
        print "  EXPLAIN DDL being used from default sqllib directory\n";
    }
}

1;

```

createmseedme.pl

```

#!/usr/bin/perl

push(@INC, split(':', $ENV{'PATH'}));

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

$seedTime; #holds timestamp which all seeds are created from
$numSeeds; #number of seeds to create
$seedFile; #filename of seedfile

#create base seed
$seedTime = (localtime)[4]; #gets month
$seedTime++; #Months start at 0, not 1, so increment month so that april is 4 and
not 3
# ensures a standard length of 9 or 10 (depending on the month) for
mm/dd/hh/mm/ss
# ie 404040404 instead of 44444 for april 4 04:04:04. A '0' is not necessary for a
# month < 10 though.
# (localtime)[3] gets day, [2] gets hour, [1] gets minute, and [0] gets second.
for ($i = 3; $i > -1; $i--) {
    $t = (localtime)[$i];
    if ($t < 10) {
        $t = "0".$t; #inserts a '0' infront of single digit number
    }
    $seedTime = $seedTime.$t
}

print "****Createmseedme base timestamp is: $seedTime\n";

#set # of seeds and seed filename
if (@ARGV eq 1) {
    $numSeeds = int($ARGV[0]);
    if ($numSeeds eq 0) {
        $numSeeds = 1000;
    }
}
else {
    $numSeeds = 1000; #default value
}

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

$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$seedFile = "$auditDir${delim}auditruns${delim}mseedme";

#create seed file and populate it, with each new seed incremented by 1.
open(SEEDFILE, ">$seedFile") || warn ("Can not open the file $seedFile!\n");
for ($i = 0; $i < $numSeeds; $i++)
{
    print SEEDFILE $seedTime++."\n";
}
close SEEDFILE || warn ("Can not close the file $seedFile!\n");
1;

```

createUfTables

```

-----
-- Create Update Function Tables
-----

```

```

CREATE TABLE TPCDTEMP.ORDERS_NEW ( APP_ID INTEGER NOT
NULL,

```

```

        O_ORDERKEY    INTEGER NOT NULL,
        O_CUSTKEY     INTEGER NOT NULL,
        O_ORDERSTATUS  CHAR(1) NOT NULL,
        O_TOTALPRICE   FLOAT NOT NULL,
        O_ORDERDATE    DATE NOT NULL,
        O_ORDERPRIORITY CHAR(15) NOT NULL,
        O_CLERK        CHAR(15) NOT NULL,
        O_SHIPPRIORITY  INTEGER NOT NULL,
        O_COMMENT      VARCHAR(79) NOT NULL WITH

```

```

DEFAULT)
    PARTITIONING KEY (O_ORDERKEY)
    IN DATA_INDEX;

```

```

CREATE TABLE TPCDTEMP.ORDERS_DEL ( APP_ID INTEGER NOT
NULL,

```

```

        O_ORDERKEY    INTEGER NOT NULL)
    PARTITIONING KEY (O_ORDERKEY)
    IN DATA_INDEX;

```

```

CREATE TABLE TPCDTEMP.LINEITEM_NEW ( APP_ID INTEGER NOT
NULL,

```

```

        L_ORDERKEY    INTEGER NOT NULL,
        L_PARTKEY     INTEGER NOT NULL,
        L_SUPPKEY     INTEGER NOT NULL,
        L_LINENUMBER  INTEGER NOT NULL,
        L_QUANTITY    FLOAT NOT NULL,
        L_EXTENDEDPRICE FLOAT NOT NULL,
        L_DISCOUNT   FLOAT NOT NULL,
        L_TAX         FLOAT NOT NULL,
        L_RETURNFLAG  CHAR(1) NOT NULL,
        L_LINESTATUS  CHAR(1) NOT NULL,
        L_SHIPDATE    DATE NOT NULL,
        L_COMMITDATE  DATE NOT NULL,
        L_RECEIPTDATE DATE NOT NULL,
        L_SHIPINSTRUCT CHAR(25) NOT NULL,
        L_SHIPMODE    CHAR(10) NOT NULL,
        L_COMMENT     VARCHAR(44) NOT NULL WITH

```

```

DEFAULT)
    PARTITIONING KEY (L_ORDERKEY)
    IN DATA_INDEX;

```

```

CREATE INDEX TPCDTEMP.I_ORDERS_NEW ON
TPCDTEMP.ORDERS_NEW

```

```

( APP_ID,
  O_ORDERKEY,
  O_CUSTKEY,
  O_ORDERSTATUS,
  O_TOTALPRICE,
  O_ORDERDATE,
  O_ORDERPRIORITY,
  O_CLERK,
  O_SHIPPRIORITY,
  O_COMMENT);

```

```

CREATE INDEX TPCDTEMP.I_LINEITEM_NEW ON
TPCDTEMP.LINEITEM_NEW (APP_ID);

```

```

CREATE UNIQUE INDEX TPCDTEMP.I_ORDERS_DEL ON
TPCDTEMP.ORDERS_DEL
(APP_ID, O_ORDERKEY);

```

```

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;

```

dss.bp

```
alter bufferpool IBMDEFAULTBP size -1;
create bufferpool BP32KDATA size 40000 numblockpages 10000 blocksize 16
pagesize 32K;
commit;
```

dss.db2set

```
db2set DB2_EXTENDED_OPTIMIZATION=y;
db2set DB2_CORRELATED_PREDICATES=Y;
db2set DB2_LIKE_VARCHAR=Y,Y;
db2set DB2_HASH_JOIN=Y;
db2set DB2BPVARS=/home/tpch/tpcd/custom/diskio/bpvars;
db2set DB2_VECTOR=Y;
db2set DB2OPTIONS='-t -v +c';
db2set DB2_PARALLEL_IO='*';
```

dss.db2set.load

```
db2set DB2_EXTENDED_OPTIMIZATION=y;
db2set DB2_CORRELATED_PREDICATES=Y;
db2set DB2_LIKE_VARCHAR=Y,Y;
db2set DB2_HASH_JOIN=Y;
db2set DB2BPVARS=/home/tpch/tpcd/custom/diskio/bpvars;
db2set DB2_VECTOR=Y;
db2set DB2OPTIONS='-t -v +c';
db2set DB2_PARALLEL_IO='*';
db2set DB2NOLIOAIO=no;
```

dss.dbcfg

```
update db cfg for tpcd using
locklist      100000
buffpage      160000
sortheap      14000
dbheap        6000
util_heap_sz  5000
num_ioservers 12
num_iocleaners 6
chngpgs_thresh 60
NUM_FREQVALUES 0
NUM_QUANTILES 300
maxfilop      1024
dft_degree    6
app_ctl_heap_sz 2048
CATALOGCACHE_SZ 64
maxappls      192;
```

dss.dbcfg.load

```
update database configuration for tpcd using
DFT_QUERYOPT  7
buffpage      20000
sortheap      20000
locklist      16384
maxlocks      60
util_heap_sz  80000
num_ioservers 12
num_iocleaners 4
chngpgs_thresh 60
NUM_FREQVALUES 0
NUM_QUANTILES 300
maxfilop      1024
dft_degree    6
```

```
app_ctl_heap_sz 2048
logbufsz      64
logfilsiz     10000
logprimary    10 ;
```

dss.dbmcfg

```
update dbm cfg using
diaglevel     0
notifylevel   0
health_mon    off
numdb         1
FEDERATED     no
sheapthres    144000
max_querydegree 8
NUM_INITAGENTS 4
FCM_NUM_BUFFERS 512
NUM_POOLAGENTS 8
intra_parallel yes;
```

dss.dbmcfg.load

```
update dbm cfg using
CPUSPEED      2.676617e-07
FEDERATED     no
--HEALTH_MON  off
sheapthres    144000
max_querydegree 4
NUM_INITAGENTS 4
FCM_NUM_BUFFERS 512
--FCM_NUM_RQB 256
NUM_POOLAGENTS 4
intra_parallel yes;
```

dss.index

```
-----
-- Create Indexes
-----

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

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

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

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

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

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

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

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

```

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

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

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

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

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

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

--values(current timestamp);
--CREATE UNIQUE INDEX TPCD.O_OK ON TPCD.ORDERS
(O_ORDERKEY ASC) PCTFREE 3;
--commit work;

--Q22 degrades from 34 -> 61.9 seconds still removed for UF performance
--values(current timestamp);
--CREATE INDEX TPCD.O_CK ON TPCD.ORDERS (O_CUSTKEY ASC)
PCTFREE 3;
--commit work;

--values(current timestamp);

```

```

--CREATE INDEX TPCD.L_OK ON TPCD.LINEITEM (L_ORDERKEY ASC)
PCTFREE 3;
--commit work;

--Q17 improved from 298 seconds -> 191 seconds. Removed for UF performance
--values(current timestamp);
--create index tpcd.l_pk on tpcd.lineitem (l_partkey asc) pctfree 3;
--commit work;
--values(current timestamp);

create index tpcd.l_sk_pk on tpcd.lineitem (l_suppkey asc, l_partkey asc) pctfree
3;
commit work;

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

```

dss.load

```

-- connect is needed
connect to tpcd;

values ( current timestamp, '*** start load of region' );
LOAD FROM /data/region.tbl OF DEL
MODIFIED BY COLDEL| FASTPARSE
MESSAGES /tmp/tpch/region.msg
REPLACE INTO TPCD.REGION
STATISTICS NO NONRECOVERABLE;

COMMIT WORK;

values ( current timestamp, '*** start load of nation' );
LOAD FROM /data/nation.tbl OF DEL
MODIFIED BY COLDEL| FASTPARSE
MESSAGES /tmp/tpch/nation.msg
REPLACE INTO TPCD.NATION
STATISTICS NO
NONRECOVERABLE;

COMMIT WORK;

values ( current timestamp, '*** start load of partsupp' );
LOAD FROM
/data/partsupp.tbl
OF DEL
MODIFIED BY COLDEL| FASTPARSE
MESSAGES /tmp/tpch/partsupp.msg
REPLACE INTO TPCD.PARTSUPP
STATISTICS NO
NONRECOVERABLE
CPU_PARALLELISM 12 DISK_PARALLELISM 12;

COMMIT WORK;

values ( current timestamp, '*** start load of supplier' );
LOAD FROM /data/supplier.tbl OF DEL
MODIFIED BY COLDEL| FASTPARSE
MESSAGES /tmp/tpch/supplier.msg
REPLACE INTO TPCD.SUPPLIER
STATISTICS NO
NONRECOVERABLE
CPU_PARALLELISM 12 DISK_PARALLELISM 12;

COMMIT WORK;

values ( current timestamp, '*** start load of part' );
LOAD FROM /data/part.tbl OF DEL
MODIFIED BY COLDEL| FASTPARSE

```

```

MESSAGES /tmp/tpch/part.msg
REPLACE INTO TPCD.PART
STATISTICS NO
NONRECOVERABLE
CPU_PARALLELISM 12 DISK_PARALLELISM 12;

```

```
COMMIT WORK;
```

```

values ( current timestamp, '*** start load of customer' );
LOAD FROM /data/customer.tbl OF DEL
MODIFIED BY COLDEL| FASTPARSE
MESSAGES /tmp/tpch/customer.msg
REPLACE INTO TPCD.CUSTOMER
STATISTICS NO
NONRECOVERABLE
CPU_PARALLELISM 12 DISK_PARALLELISM 12;

```

```
COMMIT WORK;
```

```

values ( current timestamp, '*** start load of orders' );
LOAD FROM
/data/orders.tbl.1.merged,
/data/orders.tbl.2.merged,
/data/orders.tbl.3.merged,
/data/orders.tbl.4.merged,
/data/orders.tbl.5.merged,
/data/orders.tbl.6.merged,
/data/orders.tbl.7.merged,
/data/orders.tbl.8.merged,
/data/orders.tbl.9.merged,
/data/orders.tbl.10.merged
of DEL
MODIFIED BY COLDEL| FASTPARSE
MESSAGES /tmp/tpch/order.msg
REPLACE INTO TPCD.ORDERS
STATISTICS NO
NONRECOVERABLE
CPU_PARALLELISM 12 DISK_PARALLELISM 12;

```

```
COMMIT WORK;
```

```

values ( current timestamp, '*** start load of lineitem' );
LOAD FROM
/data/lineitem.tbl.1.merged,
/data/lineitem.tbl.2.merged,
/data/lineitem.tbl.3.merged,
/data/lineitem.tbl.4.merged,
/data/lineitem.tbl.5.merged,
/data/lineitem.tbl.6.merged,
/data/lineitem.tbl.7.merged,
/data/lineitem.tbl.8.merged,
/data/lineitem.tbl.9.merged,
/data/lineitem.tbl.10.merged,
/data/lineitem.tbl.11.merged,
/data/lineitem.tbl.12.merged,
/data/lineitem.tbl.13.merged,
/data/lineitem.tbl.14.merged,
/data/lineitem.tbl.15.merged,
/data/lineitem.tbl.16.merged,
/data/lineitem.tbl.17.merged,
/data/lineitem.tbl.18.merged,
/data/lineitem.tbl.19.merged,
/data/lineitem.tbl.20.merged,
/data/lineitem.tbl.21.merged,
/data/lineitem.tbl.22.merged,
/data/lineitem.tbl.23.merged,
/data/lineitem.tbl.24.merged,
/data/lineitem.tbl.25.merged,
/data/lineitem.tbl.26.merged,
/data/lineitem.tbl.27.merged,
/data/lineitem.tbl.28.merged,
/data/lineitem.tbl.29.merged,
/data/lineitem.tbl.30.merged,
/data/lineitem.tbl.31.merged,

```

```

/data/lineitem.tbl.32.merged,
/data/lineitem.tbl.33.merged,
/data/lineitem.tbl.34.merged,
/data/lineitem.tbl.35.merged,
/data/lineitem.tbl.36.merged,
/data/lineitem.tbl.37.merged,
/data/lineitem.tbl.38.merged,
/data/lineitem.tbl.39.merged,
/data/lineitem.tbl.40.merged,
/data/lineitem.tbl.41.merged,
/data/lineitem.tbl.42.merged,
/data/lineitem.tbl.43.merged,
/data/lineitem.tbl.44.merged,
/data/lineitem.tbl.45.merged,
/data/lineitem.tbl.46.merged,
/data/lineitem.tbl.47.merged,
/data/lineitem.tbl.48.merged,
/data/lineitem.tbl.49.merged,
/data/lineitem.tbl.50.merged
of DEL
MODIFIED BY COLDEL| FASTPARSE
MESSAGES /tmp/tpch/lineitem.msg
REPLACE INTO TPCD.LINEITEM
STATISTICS NO
NONRECOVERABLE
CPU_PARALLELISM 12 DISK_PARALLELISM 12;

```

```
COMMIT WORK;
```

```

values ( current timestamp, '*** start counts after loads' );
select count(*) from tpcd.lineitem;
select count(*) from tpcd.orders;
select count(*) from tpcd.customer;
select count(*) from tpcd.supplier;
select count(*) from tpcd.part;
select count(*) from tpcd.partsupp;
select count(*) from tpcd.nation;
select count(*) from tpcd.region;
commit;

```

```

connect reset;
Terminate;

```

dss.load.qual

```

-- connect is needed
connect to tpcd;

```

```

values ( current timestamp, '*** start load of region' );
LOAD FROM /data/qual/region.tbl OF DEL
MODIFIED BY COLDEL| FASTPARSE
MESSAGES /tmp/tpch/region.msg
REPLACE INTO TPCD.REGION
STATISTICS NO NONRECOVERABLE;

```

```
COMMIT WORK;
```

```

values ( current timestamp, '*** start load of nation' );
LOAD FROM /data/qual/nation.tbl OF DEL
MODIFIED BY COLDEL| FASTPARSE
MESSAGES /tmp/tpch/nation.msg
REPLACE INTO TPCD.NATION
STATISTICS NO
NONRECOVERABLE;

```

```
COMMIT WORK;
```

```

values ( current timestamp, '*** start load of partsupp' );
LOAD FROM
/data/qual/partsupp.tbl
OF DEL

```



```

MODIFIED BY COLDEL| FASTPARSE
MESSAGES /tmp/tpch/partsupp.msg
REPLACE INTO TPCD.PARTSUPP
STATISTICS NO
NONRECOVERABLE
CPU_PARALLELISM 12 DISK_PARALLELISM 12;

```

```
COMMIT WORK;
```

```

values ( current timestamp, '*** start load of supplier' );
LOAD FROM /data/qual/supplier.tbl OF DEL
MODIFIED BY COLDEL| FASTPARSE
MESSAGES /tmp/tpch/supplier.msg
REPLACE INTO TPCD.SUPPLIER
STATISTICS NO
NONRECOVERABLE
CPU_PARALLELISM 12 DISK_PARALLELISM 12;

```

```
COMMIT WORK;
```

```

values ( current timestamp, '*** start load of part' );
LOAD FROM /data/qual/part.tbl OF DEL
MODIFIED BY COLDEL| FASTPARSE
MESSAGES /tmp/tpch/part.msg
REPLACE INTO TPCD.PART
STATISTICS NO
NONRECOVERABLE
CPU_PARALLELISM 12 DISK_PARALLELISM 12;

```

```
COMMIT WORK;
```

```

values ( current timestamp, '*** start load of customer' );
LOAD FROM /data/qual/customer.tbl OF DEL
MODIFIED BY COLDEL| FASTPARSE
MESSAGES /tmp/tpch/customer.msg
REPLACE INTO TPCD.CUSTOMER
STATISTICS NO
NONRECOVERABLE
CPU_PARALLELISM 12 DISK_PARALLELISM 12;

```

```
COMMIT WORK;
```

```

values ( current timestamp, '*** start load of orders' );
LOAD FROM
/data/qual/orders.tbl
of DEL
MODIFIED BY COLDEL| FASTPARSE
MESSAGES /tmp/tpch/order.msg
REPLACE INTO TPCD.ORDERS
STATISTICS NO
NONRECOVERABLE
CPU_PARALLELISM 12 DISK_PARALLELISM 12;

```

```
COMMIT WORK;
```

```

values ( current timestamp, '*** start load of lineitem' );
LOAD FROM
/data/qual/lineitem.tbl
of DEL
MODIFIED BY COLDEL| FASTPARSE
MESSAGES /tmp/tpch/lineitem.msg
REPLACE INTO TPCD.LINEITEM
STATISTICS NO
NONRECOVERABLE
CPU_PARALLELISM 12 DISK_PARALLELISM 12;

```

```
COMMIT WORK;
```

```

values ( current timestamp, '*** start counts after loads' );
select count(*) from tpch.lineitem;
select count(*) from tpch.orders;
select count(*) from tpch.customer;
select count(*) from tpch.supplier;

```

```

select count(*) from tpch.part;
select count(*) from tpch.partsupp;
select count(*) from tpch.nation;
select count(*) from tpch.region;
commit;

```

```

connect reset;
terminate;

```

dss.runstats

```
RUNSTATS ON TABLE TPCD.NATION WITH DISTRIBUTION on all columns
```

```

and columns (
  n_name like statistics,
  n_comment like statistics )
AND detailed INDEXES ALL;

```

```
commit;
```

```
RUNSTATS ON TABLE TPCD.REGION WITH DISTRIBUTION on all columns
```

```

and columns (
  r_name like statistics,
  r_comment like statistics )
AND detailed INDEXES ALL;

```

```
commit;
```

```
RUNSTATS ON TABLE TPCD.SUPPLIER WITH DISTRIBUTION on all columns
```

```

and columns (
  s_name like statistics,
  s_address like statistics,
  s_phone like statistics,
  s_comment like statistics)
AND detailed INDEXES ALL;

```

```
commit;
```

```
RUNSTATS ON TABLE TPCD.PART WITH DISTRIBUTION on all columns
```

```

and columns (
  p_name like statistics,
  p_mfgr like statistics,
  p_brand like statistics,
  p_type like statistics,
  p_container like statistics,
  p_comment like statistics)
AND detailed INDEXES ALL;

```

```
commit;
```

```
RUNSTATS ON TABLE TPCD.PARTSUPP WITH DISTRIBUTION on all columns
```

```

and columns (
  ps_comment like statistics)
AND detailed INDEXES ALL;

```

```
commit;
```

```
RUNSTATS ON TABLE TPCD.CUSTOMER WITH DISTRIBUTION on all columns
```

```

and columns (
  c_name like statistics,
  c_address like statistics,
  c_phone like statistics,
  c_mktsegment like statistics,
  c_comment like statistics)
AND detailed INDEXES ALL;

```

```
commit;
```

```
RUNSTATS ON TABLE TPCD.ORDERS WITH DISTRIBUTION on all columns
```

```

and columns (
  o_orderstatus like statistics,
  o_orderpriority like statistics,
  o_clerk like statistics,
  o_comment like statistics)
AND detailed INDEXES ALL;

```

```
commit;
```

```
RUNSTATS ON TABLE TPCD.LINEITEM WITH DISTRIBUTION on all columns
```

```

and columns (

```

```

l_returnflag like statistics,
l_linestatus like statistics,
l_shipinstruct like statistics,
l_shipmode like statistics,
l_comment like statistics)
AND INDEXES ALL;
COMMIT WORK;

```

dss.tbl

```

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 DATA_INDEX;

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

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 DATA_INDEX;

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 DATA_INDEX;

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 DATA_INDEX;

CREATE TABLE TPCD.CUSTOMER (
    C_CUSTKEY INTEGER NOT NULL,
    C_NAME VARCHAR(25) NOT NULL,
    C_ADDRESS VARCHAR(40) NOT NULL,
    C_NATIONKEY INTEGER NOT NULL,
    C_PHONE CHAR(15) NOT NULL,
    C_ACCTBAL FLOAT NOT NULL,
    C_MKTSEGMENT CHAR(10) NOT NULL,
    C_COMMENT VARCHAR(117) NOT NULL)
IN DATA_INDEX;

CREATE TABLE TPCD.ORDERS (
    O_ORDERKEY INTEGER NOT NULL,
    O_CUSTKEY INTEGER NOT NULL,

```

```

O_ORDERSTATUS CHAR(1) NOT NULL,
O_TOTALPRICE FLOAT NOT NULL,
O_ORDERDATE DATE NOT NULL,
O_ORDERPRIORITY CHAR(15) NOT NULL,
O_CLERK CHAR(15) NOT NULL,
O_SHIPPRIORITY INTEGER NOT NULL,
O_COMMENT VARCHAR(79) NOT NULL)
ORGANIZE BY (O_ORDERDATE)
IN DATA_INDEX;
--PARTITION BY RANGE (O_ORDERDATE)
--(STARTING '1/1/1992' ENDING '12/31/1998' EVERY 1 MONTH);

```

```

CREATE TABLE TPCD.LINEITEM (
    L_ORDERKEY INTEGER NOT NULL,
    L_PARTKEY INTEGER NOT NULL,
    L_SUPPKEY INTEGER NOT NULL,
    L_LINENUMBER INTEGER NOT NULL,
    L_QUANTITY FLOAT NOT NULL,
    L_EXTENDEDPRICE FLOAT NOT NULL,
    L_DISCOUNT FLOAT NOT NULL,
    L_TAX FLOAT NOT NULL,
    L_RETURNFLAG CHAR(1) NOT NULL,
    L_LINestatus CHAR(1) NOT NULL,
    L_SHIPDATE DATE NOT NULL,
    L_COMMITDATE DATE NOT NULL,
    L_RECEIPTDATE DATE NOT NULL,
    L_SHIPINSTRUCT CHAR(25) NOT NULL,
    L_SHIPMODE CHAR(10) NOT NULL,
    L_COMMENT VARCHAR(44) NOT NULL)
ORGANIZE BY (L_SHIPDATE)
IN DATA_INDEX;
--PARTITION BY RANGE (L_SHIPDATE)
--(STARTING '1/1/1992' ENDING '12/31/1998' EVERY 1 MONTH);

```

COMMIT WORK;

```

alter table tpcd.nation pctfree 0;
alter table tpcd.region pctfree 0;
alter table tpcd.customer pctfree 0;
alter table tpcd.supplier pctfree 0;
alter table tpcd.part pctfree 0;
alter table tpcd.partsupp pctfree 0;
alter table tpcd.lineitem pctfree 0;
alter table tpcd.orders pctfree 0;

```

commit;

dss.tbbsp

```

create regular tablespace DATA_INDEX
pagesize 32k
managed by database
using (
    DEVICE '/dev/sdb2' 25393152K ,
    DEVICE '/dev/sdc2' 25393152K ,
    DEVICE '/dev/sdd2' 25393152K ,
    DEVICE '/dev/sde2' 25393152K ,
    DEVICE '/dev/sdf2' 25393152K ,
    DEVICE '/dev/sdg2' 25393152K
)
bufferpool BP32KDATA
extentsize 16
prefetchsize 384;
-- overhead 25;
-- transferrate 1.5;

create temporary tablespace TEMP32
pagesize 32k
managed by database using (
    DEVICE '/dev/sdb1' 16603120K ,

```

```

    DEVICE '/dev/sdc1' 16603120K ,
    DEVICE '/dev/sdd1' 16603120K ,
    DEVICE '/dev/sde1' 16603120K ,
    DEVICE '/dev/sdf1' 16603120K ,
    DEVICE '/dev/sdg1' 16603120K
)
bufferpool BP32KDATA
extentsize 16
prefetchsize 384;
-- overhead 8
-- transferrate 1.5;

create temporary tablespace TEMP4K
pagesize 4k
managed by database using (
    DEVICE '/dev/sdb3' 16603136K ,
    DEVICE '/dev/sdc3' 16603136K ,
    DEVICE '/dev/sdd3' 16603136K ,
    DEVICE '/dev/sde3' 16603136K ,
    DEVICE '/dev/sdf3' 16603136K ,
    DEVICE '/dev/sdg3' 16603136K
)
EXTENTSIZE 128
PREFETCHSIZE 1536;
commit work;
drop tablespace tempspace1;
commit work;

```

ploaduf1

```

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

```

ploaduf2

```

#!/bin/ksh
RFpair=$1;
db2 connect to tpcd
db2 "load from /data/update/delete.${RFpair}.new of del modified by coldel|
fastparse messages /dev/null replace into TPCDTEMP.ORDERS_DEL
nonrecoverable;"
db2 commit;
db2 connect reset
db2 terminate

```

tpcd.setup

```

# NOTE: ALL variable defitions must have a comment at the end - haven't got
# the getvars script recognizing the uncommented line yet
TPCD_PLATFORM=linux # aix, nt, sun ....
TPCD_VERSION=2 # 1 or 2 (Version of tpcd). Default 1
TPCD_DBNAME=TPCD # name to create database under
TPCD_WORKLOAD=H # TPC version (R for TPCR, H for TPCH)
TPCD_AUDIT_DIR=/home/tpch/tpcd # top level directory of tar file for
# all the tpcd scripts
TPCD_PRODUCT=v5 # v5 or pe
# Use pe if you really are using pe v1.2!
# but I won't guarantee that it will work!
TPCD_MODE=smp # uni/smp/mln/mp
TPCD_PHYS_NODE=1 # number of physical nodes
TPCD_LN_PER_PN=1 # number of logical nodes per physical node
TPCD_SF=100 # size of the database (1=1GB,...) to
# get test size databases use:
# 0.012 = 12MB
# 0.1 = 100MB
TPCD_BUILD_STAGE=ALL # where to start the build - currently the

```

```

# following is possible:
# ALL - do everything (create,load,
# index,stats,config) (Default)
# CRTTBSP - start after create db and
# config setting. Start righ at
# create tbsp
# LOAD - start from the load of the tables
# INDEX - start from the index creation
# (NOTE if earlyindex is specified,
# then this will do the create index
# followed by the load...)
# RUNSTATS - start from the runstats
# (NOTE Do not use this option if
# distribution stats are gathered
# as part of the load, this will
# start after the load and indices
# have been created.
# CONFIG - start from the setting up of
# the benchmark runs config setup
#

```

```

TPCD_DBPATH=/filesystem/dbpath # path for database (defaults to home)
TPCD_DDLPATH=/home/tpch/tpcd/custom # path for all ddl files and
customized

```

```

# scripts (load script), config files,etc

```

```

TPCD_BUFFERPOOL_DEF=dss.bp # name of file with bufferpool
definitions

```

```

# and sizes

```

```

TPCD_NODEGROUP_DEF=NULL # name of file in ddlpath with
nodegroup

```

```

# definitions

```

```

TPCD_EXPLAIN_DDL=NULL # file with DDL for explains
statements

```

```

# 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=dss.tbsp # ddl file for tablespaces

```

```

TPCD_DDL=dss.tbl # ddl file for tables

```

```

TPCD_QUAL_TBSP_DDL=dss.ddl12MB.tbsp.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=dss.index # ddl file for indexes

```

```

TPCD_EXTRAINDEX=no # no = no extra indexes
# filename = If you want to create some
# indices before
# the load, and some indices after, then
# use this additional file to specify the

```

```

TPCD_ADD_RI=NULL # file name that contains any RI
# constraints to add after index creation
# set to NULL (default) if unused
# indices to create after the load.

```

```

TPCD_AST=NULL # file name that contains complete AST
# definition including connection to
# the database, summary table creation,
# population, indexing and runstats.

```

```

TPCD_RUNSTATS=dss.runstats # ddl file for runstats. If you have
# created indices before the load (ie
# TPCD_EARLYINDEX=yes), have specified to
# gather stats on the load command (either
# through your own load script or by using
# TPCD_LOADSTATS=yes, AND you have
# specified a file for TPCD_EXTRAINDEX
# then this runstats file should include
# the runstats commands specifically for
# the extra indices.

```

```

TPCD_RUNSTATSHORT=NULL # NOTE!! THIS IS BUGGY....I
can't get it to

```

```

# work on UNI successfully
# ddl file for short runstats that are
# run in the background while the
# TPCD_RUNSTATS are run in the foreground
# of the build. If this is used, then

```

```

#TPCD_RUNSTATS should have the runstats
# command for lineitem and
#TPCD_RUNSTATSHORT should have runstats
# commands for all other tables.
TPCD_DBGEN=/home/tpch/tpcd/appendix.v2/dbgen # path name to data
generation code

# Parameters used to specify source of
# data for load scripts
TPCD_INPUT=NULL # NULL - use dbgen generated data OR
# path name - to the pre-generated
# flat files
# /gw1/dss/12MB - path for pregenerated 12MB
# /gw1/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=/home/tpch/tpcd/tailor # path name for the directory used
to
# generate split specific config files
# only used for partitioned environment

TPCD_EARLYINDEX=no # create indexes before the load

# LOAD specific parameters follow:
TPCD_LOAD_DB2SET_SCRIPT=dss.db2set.load # Script that contains
the db2set commands
# for the load process Use NULL if not
# specified
TPCD_LOAD_CONFIGFILE=dss.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=dss.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_QUALCONFIGFILE=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_QUALCONFIGFILE=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=/tmp/tpch # path for LOAD temp files
# defaults to /u/<instance>/sql/lib/tmp
# used in load script only
TPCD_SORTBUF=4096 # sortbuf size for LOAD
# used in load script only
TPCD_LOAD_PARALLELISM=12 # 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=/filesystem/backup/ # directory where backup files are
placed

```

```

TPCD_LOGPRIMARY=NULL # NULL/value = how many primary
log files
# to configure. If NULL is specified then
# the default is not changed.
TPCD_LOGFILSIZ=NULL # NULL/value = how 4KB pages to use
for
# logfilsiz db cfg parameter. If NULL is
# specified then the default is not changed
TPCD_LOGSECOND=NULL # NULL/value = how many secondary
log files
# to configure. If NULL is specified then
# the default is not changed.
TPCD_LOG_DIR=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=yes # yes/no - whether to turn LOG_RETAIN on
# i.e. are backups needed to be taken

# CONFIG specific parameters
TPCD_DB2SET_SCRIPT=dss.db2set # Script that contains the db2set
commands
# for the benchmark run. Use NULL if not
# specified
TPCD_CONFIGFILE=dss.dbcfg # name of configuration file in ddl path
# that will be used for the benchmark run
TPCD_DBM_CONFIG=dss.dbmcfg # name of config file for database manager
# cfg parms
TPCD_QUALCONFIGFILE=dss.dbconfig0.012GB # name of database
cfg file in ddl path
# for qualification database
TPCD_DBM_QUALCONFIG=dss.dbmconfig0.012GB # 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, sheaphres
# and ioservers parms for load, create
# index and runstats
# NOTE that this parameter is ignored if
# a TPCD_LOAD_CONFIGFILE

TPCD_SMPDEGREE=8 # 1...# of degrees of parallelism to run
# with
TPCD_AGENTPRI=NULL # set agentpri to this value (default
# is SYSTEM)

TPCD_ACTIVATE=no # activate the database upon build
# completion

# run specific parameters
TPCD_AUDIT=yes # no/yes
# no - don't set up qualification db stuff
# yes - set up qualification db queries
# - build the update function tables
# and data before we get into the
# timing of the creation of the
# tables and the load.
TPCD_TMP_DIR=/tmp/tpch # 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=tpcdqual      # name of qualification database
TPCD_NUMSTREAM=5               # number of streams for the throughput
test
TPCD_FLATFILES=/data/update # 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=preloadUF # 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=NULL # 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=1024       # number of chunks to split the update
                             # function into.
TPCD_CONCURRENT_INSERTS=64    # number of insert chunks that are
run
                             # concurrently. TPCD_SPLIT_UPDATES
                             # should be evenly divisible by this number
TPCD_CONCURRENT_INSERTS_LOAD=4 # 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=1024      # number of portions to split the delete
                             # function into.
                             # this variable is only valid in UNI/SMP
                             # mode.
TPCD_CONCURRENT_DELETES=64    # number of delete chunks that
are run
TPCD_GEN_UPDATEPAIRS=18       # number of pairs of update function
data
                             # to generate
                             # if 0 the update data generation and
                             # setup will not be done. use this if
                             # you don't want to run the update
                             # functions (Update functions not
                             # fully tested in new env't yet)
TPCD_GENERATE_SEED_FILE=yes   # yes/no These are the seed files
for
                             # generating the query substitution values
                             # yes - generate a seed file base on
                             # year/month/day (for audited runs)
                             # no - use qgen's default seeds
TPCD_RUN_ON_MULTIPLE_NODES=NULL # pe V1.2 only - will we be
running each
                             # query stream of throughput starting at
                             # different nodes or from same node
TPCD_STATS_INTERVAL=30       # timing interval for vmstats/iostats
TPCD_STATS_THRU_INT=300      # timing interval for vmstats/iostats
for
                             # throughput run
TPCD_GATHER_STATS=off         # on/off - only implement for AIX yet
                             # on = gather statistics around power
                             # test run (vmstat,iostat,netstat)
                             # off = no stats gathered during power run

```

```

TPCD_UFTEMP=UFTEMP           # base name of tablespace(s) where the
                             # staging tables for the update functions
                             # are created
                             # this name will be used as the
                             # basename for the tablespaces...eg
                             # UFTEMP1 UFTEMP2 ....
TPCD_HAVECOMPILER=yes        # rebuild tpcdbatch executable
                             # yes/no
TPCD_SLEEP=5                 # ?
TPCD_INLISTMAX=default       # max num of keys to delete at a time
                             # for UF2, use "default" for default.
TPCD_LOAD_SCRIPT=dss.load     # 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=/home/tpch/sql/lib/db2dump # directory where the db2diag.log
can
                             # be found for the durability tests
TPCD_APPEND_ON=yes           # set to no if the cluster indexes are used

```

Appendix C: Qualification Query Output

Qualification Queries

Start timestamp 12/08/04 17:56:16.634556

-- Query 01 - Var_0 Rev_01 - Pricing Summary Report Query

Tag: Q1 Stream: -1 Sequence number: 17

```
select
l_returnflag,
l_linestatus,
sum(l_quantity) as sum_qty,
sum(l_extendedprice) as sum_base_price,
sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
avg(l_quantity) as avg_qty,
avg(l_extendedprice) as avg_price,
avg(l_discount) as avg_disc,
count(*) as count_order
from
tpcd.lineitem
where
l_shipdate <= date ('1998-12-01') - 90 day
group by
l_returnflag,
l_linestatus
order by
l_returnflag,
l_linestatus
```

L_RETURNFLAG	L_LINESTATUS	SUM_QTY	SUM_BASE_PRICE	SUM_DISC_PRICE	SUM_CHARGE	AVG_QTY
AVG_PRICE	AVG_DISC	COUNT_ORDER				

A	F	37734107.000	56586554400.730			
53758257134.870		55909065222.827	25.522			38273.130
0.050	1478493					
N	F	991417.000	1487504710.380			1413082168.054
1469649223.194		25.516	38284.468			0.050
38854						
N	O	74476040.000	111701729697.740			
106118230307.605		110367043872.498	25.502			38249.118
0.050	2920374					
R	F	37719753.000	56568041380.900			
53741292684.604		55889619119.832	25.506			38250.855
0.050	1478870					

Number of rows retrieved is: 4

Stop timestamp 12/08/04 17:56:19.931953
Query Time = 3.3 secs
Start timestamp 12/08/04 17:55:22.051323

-- Query 02 - Var_0 Rev_02 - Minimum Cost Supplier Query

Tag: Q2 Stream: -1 Sequence number: 2

```
select
s_acctbal,
s_name,
n_name,
p_partkey,
p_mfgr,
s_address,
s_phone,
s_comment
from
tpcd.part,
tpcd.supplier,
tpcd.partsupp,
tpcd.nation,
tpcd.region
where
p_partkey = ps_partkey
and s_suppkey = ps_suppkey
and p_size = 15
and p_type like '%BRASS'
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = 'EUROPE'
and ps_supplycost = (
select
min(ps_supplycost)
from
tpcd.partsupp,
tpcd.supplier,
tpcd.nation,
tpcd.region
where
p_partkey = ps_partkey
and s_suppkey = ps_suppkey
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = 'EUROPE'
)
order by
s_acctbal desc,
n_name,
s_name,
p_partkey
fetch first 100 rows only
```

S_ACCTBAL	S_NAME	N_NAME
P_PARTKEY	P_MFGR	S_ADDRESS
S_PHONE	S_COMMENT	

9938.530	Supplier#000005359	UNITED KINGDOM	
185358	Manufacturer#4	QKuHYh,vZGiwu2FWEJoLDx04	
33-429-790-6131	blithely silent pinto beans are furiously. slyly final deposits across		
9937.840	Supplier#000005969	ROMANIA	108438
Manufacturer#1	ANDENSOSmk,miq23Xfb5RWt6dvUcvt6Qa		
29-520-692-3537	carefully slow deposits use furiously. slyly ironic platelets above the ironic		
9936.220	Supplier#000005250	UNITED KINGDOM	
249	Manufacturer#4	B3rqp0xbSEim4Mpy2RH J	
33-320-228-2957	blithely special packages are. stealthily express deposits across the closely final instructi		
9923.770	Supplier#000002324	GERMANY	29821
Manufacturer#4	y3OD9UywSTOK		17-779-299-1839

quickly express packages breach quiet pinto beans. requ

9871.220 Supplier#000006373 GERMANY 43868
Manufacturer#5 J8fcXWsTqM 17-813-485-8637

never silent deposits integrate furiously blit

9870.780 Supplier#000001286 GERMANY 81285
Manufacturer#2 YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH
17-516-924-4574 final theodolites cajole slyly special,

9870.780 Supplier#000001286 GERMANY 181285
Manufacturer#4 YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH
17-516-924-4574 final theodolites cajole slyly special,

9852.520 Supplier#000008973 RUSSIA 18972
Manufacturer#2 t5L67YdBYyH6o,Vz24jpDyQ9
32-188-594-7038 quickly regular instructions wake-- carefully unusual braids
into the expres
9847.830 Supplier#000008097 RUSSIA 130557
Manufacturer#2 xMe97bpE69NzdWLoX 32-375-640-3593
slyly regular dependencies sleep slyly furiously express dep

9847.570 Supplier#000006345 FRANCE 86344
Manufacturer#1 VSt3rzK3qG698u6ld8HhOByvrTcSTsvQIDQDag
16-886-766-7945 silent pinto beans should have to snooze carefully along the
final reques

....Lines Deleted

7937.930 Supplier#000009012 ROMANIA 83995
Manufacturer#2 iUiTziH,Ek3i4lwSgunXMgrcTzwdb
29-250-925-9690 blithely bold ideas haggle quickly final, regular request

7914.450 Supplier#000001013 RUSSIA 125988
Manufacturer#2 riRcntps4KEDtYScjpMIWeYF6mNnR
32-194-698-3365 final, ironic theodolites alongside of the ironic

7912.910 Supplier#000004211 GERMANY 159180
Manufacturer#5 2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG
17-266-947-7315 final requests integrate slyly above the silent, even

7912.910 Supplier#000004211 GERMANY 184210
Manufacturer#4 2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG
17-266-947-7315 final requests integrate slyly above the silent, even

7894.560 Supplier#000007981 GERMANY 85472
Manufacturer#4 NSJ96vMROAbeXP 17-963-404-3760
regular, even theodolites integrate carefully. bold, special theodolites are slyly
fluffily iron

7887.080 Supplier#000009792 GERMANY 164759
Manufacturer#3 Y28ITVeYriT3klGdV2K8fSZ V2UqT5H1Otz
17-988-938-4296 pending, ironic packages sleep among the carefully ironic
accounts. quickly final accounts

7871.500 Supplier#000007206 RUSSIA 104695
Manufacturer#1 3w fNCnrVmvJJE95sgWZzvW
32-432-452-7731 furiously dogged pinto beans cajole. bold, express notornis
until the slyly pending

7852.450 Supplier#000005864 RUSSIA 8363
Manufacturer#4 WCNfBPZeSxh3h,c 32-454-883-3821
blithely regular deposits

7850.660 Supplier#000001518 UNITED KINGDOM
86501 Manufacturer#1 ONda3YJiHKJOC
33-730-383-3892 furiously final accounts wake carefully idle requests. even
dolphins wake acc

7843.520 Supplier#000006683 FRANCE 11680
Manufacturer#4 2Z0JGkiv01Y00oCFwUGfviIbhzCdy
16-464-517-8943 carefully bold accounts doub

Number of rows retrieved is: 100

Stop timestamp 12/08/04 17:55:22.542715
Query Time = 0.5 secs
Start timestamp 12/08/04 17:56:09.850128

-- Query 03 - Var_0 Rev_01 - Shipping Priority Query

Tag: Q3 Stream: -1 Sequence number: 11

```
select
l_orderkey,
sum(l_extendedprice * (1 - l_discount)) as revenue,
o_orderdate,
o_shippriority
from
tpcd.customer,
tpcd.orders,
tpcd.lineitem
where
c_mktsegment = 'BUILDING'
and c_custkey = o_custkey
and l_orderkey = o_orderkey
and o_orderdate < date ('1995-03-15')
and l_shipdate > date ('1995-03-15')
group by
l_orderkey,
o_orderdate,
o_shippriority
order by
revenue desc,
o_orderdate
fetch first 10 rows only
```

L_ORDERKEY	REVENUE	O_ORDERDATE	O_SHIPPRIORITY
2456423	406181.011	1995-03-05	0
3459808	405838.699	1995-03-04	0
492164	390324.061	1995-02-19	0
1188320	384537.936	1995-03-09	0
2435712	378673.056	1995-02-26	0
4878020	378376.795	1995-03-12	0
5521732	375153.922	1995-03-13	0
2628192	373133.309	1995-02-22	0
993600	371407.459	1995-03-05	0
2300070	367371.145	1995-03-13	0

Number of rows retrieved is: 10

Stop timestamp 12/08/04 17:56:10.356682
Query Time = 0.5 secs
Start timestamp 12/08/04 17:56:14.594347

-- Query 04 - Var_0 Rev_01 - Order Priority Checking Query

Tag: Q4 Stream: -1 Sequence number: 14

select

```

o_orderpriority,
count(*) as order_count
from
tpcd.orders
where
o_orderdate >= date ('1993-07-01')
and o_orderdate < date ('1993-07-01') + 3 month
and exists (
select
*
from
tpcd.lineitem
where
l_orderkey = o_orderkey
and l_commitdate < l_receiptdate
)
group by
o_orderpriority
order by
o_orderpriority

```

O_ORDERPRIORITY ORDER_COUNT

```

-----
1-URGENT          10594
2-HIGH            10476
3-MEDIUM         10410
4-NOT SPECIFIED  10556
5-LOW             10487

```

Number of rows retrieved is: 5

```

-----
Stop timestamp 12/08/04 17:56:15.933238
Query Time = 1.3 secs
Start timestamp 12/08/04 17:56:25.245908

```

-- Query 05 - Var_0 Rev_02 Local Supplier Volume Query

Tag: Q5 Stream: -1 Sequence number: 20

```

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

```

```

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

```

Number of rows retrieved is: 5

```

-----
Stop timestamp 12/08/04 17:56:28.581472
Query Time = 3.3 secs
Start timestamp 12/08/04 17:55:32.605787

```

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

Tag: Q6 Stream: -1 Sequence number: 5

```

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

```

```

REVENUE
-----
123141078.228

```

Number of rows retrieved is: 1

```

-----
Stop timestamp 12/08/04 17:55:33.100589
Query Time = 0.5 secs
Start timestamp 12/08/04 17:56:28.581472

```

-- Query 07 - Var_0 Rev_01 - Volume Shipping Query

Tag: Q7 Stream: -1 Sequence number: 21

```

select
supp_nation,
cust_nation,
l_year,
sum(volume) as revenue
from
(
select
n1.n_name as supp_nation,
n2.n_name as cust_nation,
year(l_shipdate) as l_year,
l_extendedprice * (1 - l_discount) as volume
from
tpcd.supplier,
tpcd.lineitem,

```



```

tpcd.orders,
tpcd.customer,
tpcd.nation n1,
tpcd.nation n2
where
s_suppkey = l_suppkey
and o_orderkey = l_orderkey
and c_custkey = o_custkey
and s_nationkey = n1.n_nationkey
and c_nationkey = n2.n_nationkey
and (
(n1.n_name = 'FRANCE' and n2.n_name = 'GERMANY')
or (n1.n_name = 'GERMANY' and n2.n_name = 'FRANCE')
)
and l_shipdate between date('1995-01-01') and date('1996-12-31')
) as shipping
group by
supp_nation,
cust_nation,
l_year
order by
supp_nation,
cust_nation,
l_year

```

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

Number of rows retrieved is: 4

Stop timestamp 12/08/04 17:56:35.272202
Query Time = 6.7 secs
Start timestamp 12/08/04 17:55:45.972125

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

Tag: Q8 Stream: -1 Sequence number: 8

```

select
o_year,
sum(case
when nation = 'BRAZIL' then volume
else 0
end) / sum(volume) as mkt_share
from
(
select
year(o_orderdate) as o_year,
l_extendedprice * (1 - l_discount) as volume,
n2.n_name as nation
from
tpcd.part,
tpcd.supplier,
tpcd.lineitem,
tpcd.orders,
tpcd.customer,
tpcd.nation n1,
tpcd.nation n2,

```

```

tpcd.region
where
p_partkey = l_partkey
and s_suppkey = l_suppkey
and l_orderkey = o_orderkey
and o_custkey = c_custkey
and c_nationkey = n1.n_nationkey
and n1.n_regionkey = r_regionkey
and r_name = 'AMERICA'
and s_nationkey = n2.n_nationkey
and o_orderdate between date('1995-01-01') and date('1996-12-31')
and p_type = 'ECONOMY ANODIZED STEEL'
) as all_nations
group by
o_year
order by
o_year

```

O_YEAR	MKT_SHARE
1995	0.034
1996	0.041

Number of rows retrieved is: 2

Stop timestamp 12/08/04 17:55:51.765714
Query Time = 5.8 secs
Start timestamp 12/08/04 17:55:22.542715

-- 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	31342867.234
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	51923746.576

..... Lines deleted

UNITED STATES	1995	48809032.423
UNITED STATES	1994	49296747.183
UNITED STATES	1993	48029946.801
UNITED STATES	1992	48671944.498
VIETNAM	1998	30442736.059
VIETNAM	1997	50309179.794
VIETNAM	1996	50488161.410
VIETNAM	1995	49658284.613
VIETNAM	1994	50596057.261
VIETNAM	1993	50953919.152
VIETNAM	1992	49613838.315

Number of rows retrieved is: 175

Stop timestamp 12/08/04 17:55:30.516189
Query Time = 8.0 secs
Start timestamp 12/08/04 17:56:19.931953

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

Tag: Q10 Stream: -1 Sequence number: 18

```
select
c_custkey,
c_name,
sum(l_extendedprice * (1 - l_discount)) as revenue,
c_acctbal,
n_name,
c_address,
c_phone,
c_comment
from
tpcd.customer,
tpcd.orders,
tpcd.lineitem,
tpcd.nation
where
c_custkey = o_custkey
and l_orderkey = o_orderkey
and o_orderdate >= date ('1993-10-01')
and o_orderdate < date ('1993-10-01') + 3 month
and l_returnflag = 'R'
and c_nationkey = n_nationkey
group by
c_custkey,
```

c_name,
c_acctbal,
c_phone,
n_name,
c_address,
c_comment
order by
revenue desc
fetch first 20 rows only

C_CUSTKEY	C_NAME	REVENUE	C_ACCTBAL
N_NAME	C_ADDRESS		C_PHONE
C_COMMENT			

57040	Customer#0000057040	734235.246	632.870
JAPAN	Eioyzjf4pp	22-895-641-3466	requests
sleep blithely about the furiously i	143347	Customer#0000143347	721002.695
EGYPT	1aReFYv,Kw4		2557.470
bold excuses haggle finally after the u	60838	Customer#0000060838	679127.308
BRAZIL	64EaJ5vMAHWJIBOXJklpNc2RJiWE		2454.770
12-913-494-9813 furiously even pinto beans integrate under the ruthless foxes;			
ironic, even dolphins across the slyl	101998	Customer#0000101998	637029.567
UNITED KINGDOM	01c9CILnNtfOQYmZj		3790.890
33-593-865-6378 accounts doze blithely! enticing, final deposits sleep blithely			
special accounts. slyly express accounts pla	125341	Customer#0000125341	633508.086
GERMANY	S29ODD6bceU8QSuuEJznkNaK		4983.510
17-582-695-5962 quickly express requests wake quickly blithely			
25501	Customer#0000025501	620269.785	7725.040
ETHIOPIA	W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ		
15-874-808-6793 quickly special requests sleep evenly among the special			
deposits. special deposi	115831	Customer#0000115831	596423.867
FRANCE	rFeBbEEyk dl ne7zV5fDrmiql oK09wV7pxqCgIc		5098.100
16-715-386-3788 carefully bold excuses sleep alongside of the thinly idle			
84223	Customer#0000084223	594998.024	528.650
UNITED KINGDOM	nAVZCs6BaWap rrM27N 2qBnzc5WBauxbA		
33-442-824-8191 pending, final ideas haggle final requests. unusual, regular			
asymptotes affix according to the even foxes.	54289	Customer#0000054289	585603.392
IRAN	vXCxoCsU0Bad5JQI ,oobkZ		5583.020
express requests sublate blithely regular requests. regular, even ideas solve.			20-834-292-4707
39922	Customer#0000039922	584878.113	7321.110
GERMANY	Zgy4s50l2GKN4pLDPBU8m342gIw6R		
17-147-757-8036 even pinto beans haggle. slyly bold accounts inte			
6226	Customer#0000006226	576783.761	2230.090
UNITED KINGDOM	8gPu8,NPGkfyQQ0hclYUGPIBWc,ybP5g,		
33-657-701-3391 quickly final requests against the regular instructions wake			
blithely final instructions. pa	922	Customer#0000000922	576767.533
GERMANY	Az9RFaut7NkPnc5zSD2PwHgVwr4jRzq		3869.250
17-945-916-9648 boldly final requests cajole blith			
147946	Customer#0000147946	576455.132	2030.130
ALGERIA	iANyZHjqhy7Ajah0pTrYyhJ		
10-886-956-3143 furiously even accounts are blithely above the furiousl			

115640 Customer#0000115640 569341.193 6436.100
ARGENTINA Vtgfia9ql 7EpHgecU1X 11-411-543-4901
final instructions are slyly according to the

73606 Customer#0000073606 568656.858 1785.670
JAPAN xuR0Tro5yChDfOCrjkd2ol 22-437-653-6966
furiously bold orbits about the furiously busy requests wake across the furiously
quiet theodolites. d

110246 Customer#0000110246 566842.981 7763.350
VIETNAM 7KzflgX MDOq7sOkI 31-943-426-9837
dolphins sleep blithely among the slyly final

142549 Customer#0000142549 563537.237 5085.990
INDONESIA ChqEoK43OysjdHbtKCp6dKqjNyvvi9
19-955-562-2398 regular, unusual dependencies boost slyly; ironic attainments
nag fluffily into the unusual packages?
146149 Customer#0000146149 557254.986 1791.550
ROMANIA s87fvzFQpU 29-744-164-6487 silent,
unusual requests detect quickly slyly regul

52528 Customer#0000052528 556397.351 551.790
ARGENTINA NFztyTOR10UOJ 11-208-192-3205
unusual requests detect. slyly dogged theodolites use slyly. deposit

23431 Customer#0000023431 554269.536 3381.860
ROMANIA HgiV0phqhala9aydNollb 29-915-458-2654
instructions nag quickly. furiously bold accounts cajol

Number of rows retrieved is: 20

Stop timestamp 12/08/04 17:56:21.933141
Query Time = 2.0 secs
Start timestamp 12/08/04 17:56:15.933238

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

Tag: Q11 Stream: -1 Sequence number: 15

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

)
order by
value desc

PS_PARTKEY	VALUE
129760	17538456.860
166726	16503353.920
191287	16474801.970
161758	16101755.540
34452	15983844.720
139035	15907078.340
9403	15451755.620
154358	15212937.880
38823	15064802.860

... Lines Deleted

79373	7896186.000
113808	7893353.880
27901	7892952.000
128820	7892882.720
25891	7890511.200
122819	7888881.020
154731	7888301.330
101674	7879324.600
51968	7879102.210
72073	7877736.110
5182	7874521.730

Number of rows retrieved is: 1048

Stop timestamp 12/08/04 17:56:16.373117
Query Time = 0.4 secs
Start timestamp 12/08/04 17:56:35.272202

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

Tag: Q12 Stream: -1 Sequence number: 22

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

and l_receiptdate < date ('1994-01-01') + 1 year
group by
l_shipmode
order by
l_shipmode

L_SHIPMODE HIGH_LINE_COUNT LOW_LINE_COUNT

MAIL	6202	9324
SHIP	6200	9262

Number of rows retrieved is: 2

Stop timestamp 12/08/04 17:56:40.066912
Query Time = 4.8 secs
Start timestamp 12/08/04 17:56:05.269146

-- 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

Stop timestamp 12/08/04 17:56:09.850128
Query Time = 4.6 secs
Start timestamp 12/08/04 17:55:21.756184

--#SET ROWS_OUT -1 ROWS_FETCH -1

-- Query 14 - Var_0 Rev_01 - Promotion Effect Query

Tag: Q14 Stream: -1 Sequence number: 1

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

PROMO_REVENUE

16.381

Number of rows retrieved is: 1

Stop timestamp 12/08/04 17:55:22.051323
Query Time = 0.3 secs
Start timestamp 12/08/04 17:56:16.373117

-- Query 15 - Var_a Rev_01 - Top Supplier Query

Tag: Q15a Stream: -1 Sequence number: 16

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

S_SUPPKEY	S_NAME	S_ADDRESS
S_PHONE	TOTAL_REVENUE	

8449	Supplier#000008449	Wp34zim9qYFbVctdW
20-469-856-8873	1772627.209	

Number of rows retrieved is: 1

Stop timestamp 12/08/04 17:56:16.634556
Query Time = 0.3 secs
Start timestamp 12/08/04 17:56:13.745309

-- 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

.... Lines deleted

Brand#32	STANDARD ANODIZED COPPER	23	3
Brand#33	SMALL ANODIZED BRASS	9	3
Brand#35	MEDIUM ANODIZED TIN	19	3
Brand#51	SMALL PLATED BRASS	23	3
Brand#52	MEDIUM BRUSHED BRASS	45	3
Brand#53	MEDIUM BRUSHED TIN	45	3
Brand#54	ECONOMY POLISHED BRASS	9	3
Brand#55	PROMO PLATED BRASS	19	3
Brand#55	STANDARD PLATED TIN	49	3

Number of rows retrieved is: 18314

Stop timestamp 12/08/04 17:56:14.594347
Query Time = 0.8 secs
Start timestamp 12/08/04 17:55:33.100589

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

Tag: Q17 Stream: -1 Sequence number: 6

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

```

0.2 * avg(l_quantity)
from
tpcd.lineitem
where
l_partkey = p_partkey
)

```

```

AVG_YEARLY
-----
348406.054

```

```

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

```

```

Stop timestamp 12/08/04 17:55:36.114784
Query Time = 3.0 secs
Start timestamp 12/08/04 17:55:36.114784
-----

```

-- 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#0000128120      128120   4722021  1994-04-07
544089.090              323.000
Customer#0000144617      144617   3043270  1997-02-12
530604.440              317.000

```

```

Customer#0000013940      13940   2232932  1997-04-13
522720.610              304.000
Customer#0000066790      66790   2199712  1996-09-30
515531.820              327.000
Customer#0000046435      46435   4745607  1997-07-03
508047.990              309.000

```

.... Deleted Lines

```

Customer#0000069904      69904   1742403  1996-10-19
408513.000              305.000
Customer#0000017746      17746    6882    1997-04-09   408446.930
303.000
Customer#0000013072      13072   1481925  1998-03-15
399195.470              301.000
Customer#0000082441      82441    857959  1994-02-07
382579.740              305.000
Customer#0000088703      88703   2995076  1994-01-30
363812.120              302.000

```

```

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

```

```

Stop timestamp 12/08/04 17:55:45.972125
Query Time = 9.9 secs
Start timestamp 12/08/04 17:56:21.933141
-----

```

-- Query 19 - Var_0 Rev_01 - Discounted Revenue Query

Tag: Q19 Stream: -1 Sequence number: 19

```

select
sum(l_extendedprice* (1 - l_discount)) as revenue
from
tpcd.lineitem,
tpcd.part
where
(
p_partkey = l_partkey
and p_brand = 'Brand#12'
and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
and l_quantity >= 1 and l_quantity <= 1 + 10
and p_size between 1 and 5
and l_shipmode in ('AIR', 'AIR REG')
and l_shipinstruct = 'DELIVER IN PERSON'
)
or
(
p_partkey = l_partkey
and p_brand = 'Brand#23'
and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
and l_quantity >= 10 and l_quantity <= 10 + 10
and p_size between 1 and 10
and l_shipmode in ('AIR', 'AIR REG')
and l_shipinstruct = 'DELIVER IN PERSON'
)
or
(
p_partkey = l_partkey
and p_brand = 'Brand#34'
and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
and l_quantity >= 20 and l_quantity <= 20 + 10
and p_size between 1 and 15
and l_shipmode in ('AIR', 'AIR REG')
)

```

and l_shipinstruct = 'DELIVER IN PERSON'
)

REVENUE

3083843.058

Number of rows retrieved is: 1

Stop timestamp 12/08/04 17:56:25.245908
Query Time = 3.3 secs
Start timestamp 12/08/04 17:55:30.516189

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

Tag: Q20 Stream: -1 Sequence number: 4

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

S_NAME	S_ADDRESS
Supplier#000000020	iybAE,RmTymrZVYaFZva2SH,j
Supplier#000000091	YV45D7TkfdQanOOZ7q9QxkyGUapU1oOWU6q3
Supplier#000000197	YC2Acon6kjY3zj3Fbxs2k4Vdf7X0cd2F
Supplier#000000226	83qOdU2EYRdPQAQhEtn GRZEd
Supplier#000000285	Br7e1nnt1yxrw6ImgpJ7YdhFDjuBf

.... Deleted Lines

Supplier#000009862	rJzweWeN58
Supplier#000009868	ROjGgx5gvtkmnUUoeyy7v
Supplier#000009869	ucLqxzrpBTRMewGSM29t0rNTM30g1Tu3Xgg3mKag
Supplier#000009899	7XdpAHrztU,UQFZE
Supplier#000009974	7wJ,J5DKcxSU4Kp1cQLpbcAvB5AsvKT

Number of rows retrieved is: 204

Stop timestamp 12/08/04 17:55:32.605787
Query Time = 2.1 secs
Start timestamp 12/08/04 17:55:51.765714

-- 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

.... Deleted Lines

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

Number of rows retrieved is: 100

Stop timestamp 12/08/04 17:56:05.269146
 Query Time = 13.5 secs
 Start timestamp 12/08/04 17:56:10.356682

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

Tag: Q22 Stream: -1 Sequence number: 12

```

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

CNTRYCODE NUMCUST TOTACCTBAL

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

Number of rows retrieved is: 7

Stop timestamp 12/08/04 17:56:13.745309
 Query Time = 3.4 secs

First 10 Rows of the Database

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

2 BRAZIL	1 always pending pinto beans sleep sil
3 CANADA	1 foxes among the bold requests
12 JAPAN	2 blithely final packages cajole quickly even dependencies? blithely regular deposits haggle express, ironic re
18 CHINA	2 bold accounts are. slyly ironic escapades haggle acc
4 EGYPT	4 pending accounts haggle furiously. furiously bold accounts detect. platelets at the packages haggle caref
10 IRAN	4 blithely even accounts about the furiously regular foxes nag slyly final accounts. quickly final fo
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

7 GERMANY 3 blithely ironic foxes grow. quickly
pending accounts are b
19 ROMANIA 3 deposits boost against the brave id

10 record(s) selected.

SELECT * FROM TPCD.PART FETCH FIRST 10 ROWS ONLY

P_PARTKEY	P_NAME	P_MFGR
P_BRAND	P_TYPE	P_SIZE
P_RETAILPRICE	P_COMMENT	P_CONTAINER
654	cyan burlywood beige olive navajo	Manufacturer#2
Brand#23	PROMO PLATED COPPER	2 WRAP BAG
+1.554650000000000E+003	slyly ironi	
655	tomato linen almond rosy olive	Manufacturer#1
Brand#12	STANDARD BURNISHED TIN	23 JUMBO CAN
+1.555650000000000E+003	regular pinto beans c	
656	black deep lace coral magenta	Manufacturer#1
Brand#15	STANDARD BURNISHED COPPER	17 MED CASE
+1.556650000000000E+003	quickly s	
657	saddle metallic purple cornsilk hot	Manufacturer#3
Brand#31	ECONOMY BURNISHED COPPER	36 LG PACK
+1.557650000000000E+003	specia	
658	dodger rosy tomato honeydew saddle	Manufacturer#1
Brand#15	MEDIUM PLATED COPPER	27 SM JAR
+1.558650000000000E+003	caref	
659	ivory green pink orange chartreuse	Manufacturer#3
Brand#34	MEDIUM BRUSHED BRASS	20 LG JAR
+1.559650000000000E+003	even,	
660	lime peach thistle royal pink	Manufacturer#5
Brand#51	STANDARD POLISHED TIN	8 SM CAN
+1.560660000000000E+003	slyly regular	
661	midnight drab ivory purple lawn	Manufacturer#3
Brand#32	PROMO PLATED STEEL	35 SM PKG
+1.561660000000000E+003	bold, even excuse	
662	puff brown orchid turquoise hot	Manufacturer#4
Brand#45	STANDARD PLATED STEEL	35 LG PACK
+1.562660000000000E+003	carefully even asympt	
663	ivory olive orchid cream honeydew	Manufacturer#4
Brand#41	PROMO POLISHED COPPER	6 WRAP PACK
+1.563660000000000E+003	blithely pendi	

10 record(s) selected.

SELECT * FROM TPCD.SUPPLIER FETCH FIRST 10 ROWS ONLY

S_SUPPKEY	S_NAME	S_ADDRESS
S_NATIONKEY	S_PHONE	S_COMMENT
1	Supplier#000000001	N kD4on9OM Ipw3.gf0JBoQDd7tgrzrddZ
17	27-918-335-1736	+5.755940000000000E+003 requests haggle carefully.
		accounts sublate finally. carefully ironic pa
2	Supplier#000000002	89eJ5ksX3ImxJBvXObC,
15	679-861-2259	+4.032680000000000E+003 furiously stealthy frays thrash
		alongside of the slyly express deposits. blithely regular req

3 Supplier#000000003 q1,G3Pj6OjluUYfUoH18BFTKP5aU9bEV3
1 11-383-516-1199 +4.192400000000000E+003 furiously regular
instructions impress slyly! carefu
4 Supplier#000000004 Bk7ah4CK8SYQTepEmvMkkgMwg
15 25-843-787-7479 +4.641080000000000E+003 final ideas cajole. furiously
close dep

5 Supplier#000000005 Gcdm2rJRzL5qITVzc 11
21-151-690-3663 -2.838400000000000E+002 carefully silent instructions are
slyly accepting t

6 Supplier#000000006 tQxuVm7s7CnK 14
24-696-997-4969 +1.365790000000000E+003 even requests wake carefully!
fluffily final pinto beans run slyly among t

7 Supplier#000000007 s,4TicNGB4uO6PaSqNBUq
23 33-990-965-2201 +6.820350000000000E+003 carefully express packages
believe furiously after the fur

8 Supplier#000000008 9Sq4bBH2FQEmFOocY45sRTxo6yuoG
17 27-498-742-3860 +7.627850000000000E+003 carefully express
escapades are slyly

9 Supplier#000000009 1KhUgZegwM3ua7dsYmekYBsK
10 20-403-398-8662 +5.302370000000000E+003 slyly regular decoys mold
slyly ironic dugouts. requests are carefully-- carefully

10 Supplier#000000010 Saygah3gYWMp72i PY 24
34-852-489-5855 +3.891910000000000E+003 ironic deposits poach quickly
furiously final accounts. carefull

10 record(s) selected.

SELECT * FROM TPCD.PARTSUPP FETCH FIRST 10 ROWS ONLY

PS_PARTKEY	PS_SUPPKEY	PS_AVAILQTY	PS_SUPPLYCOST
PS_COMMENT			

250 750251 4647 +2.810600000000000E+002 slyly bold instructions
nag carefully according to the

251 252 6748 +7.034700000000000E+002 blithely unusual ideas
sleep. excuses sleep slyly. requests sleep. pending

251 250252 6020 +1.541100000000000E+002 slyly ironic depths
are. quickly final pinto beans sleep. carefully even ideas about the slyly unusual
dolphins might doze against the slyly even ideas. carefully idle instruct

251 500252 9298 +8.835700000000000E+002 fluffily final ideas use
furiously quickly even deposits. carefully regular theodolites doze furiously quick

251 750252 5130 +1.831900000000000E+002 regular, pending
accounts use furiously: furiously express asymptotes are about the even ideas.
blithely silent packages are about

252 253 8207 +1.680500000000000E+002 blithely ironic pinto
beans are carefully. blithely special requests are. theodolites wake furiously busy
accounts. carefull

252 250253 1220 +4.125400000000000E+002 regular packages are
slyly final instructions. quickly bold deposits should have to are furious

252 500253 2813 +7.341500000000000E+002 unusual, ironic
requests use. brave packages affix carefully

252 750253 7645 +2.752300000000000E+002 unusual accounts
sleep blithely foxes. express, final deposits are alongside of the final, unusual
pinto bean

253 254 544 +6.580800000000000E+002 slyly unusual deposits
hinder. quickly ironic pinto beans haggle blithely except the quickly even

accounts. dinos alongside of the special, express foxes use above the carefully final plate

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

546 Customer#0000000546 GZtBXX3OaqFLbC9JNi1hmF1JFLbmRs9
19 29-936-444-8248 +3.116500000000000E+003 MACHINERY
silent instructions cajole slyly. special excuses after the fina

547 Customer#0000000547 4h SK3dVKe1tQ0NCh
22 32-696-724-2981 +6.058080000000000E+003 BUILDING carefully special
platelets detect fluffily along the blithely
548 Customer#0000000548 98nP31ToAGK
tCCKYm7HqBZi0dLjy0JzIMMRCmkj 4 14-787-370-8722
+9.045000000000000E+001 BUILDING regular, ironic ideas doze among

549 Customer#0000000549
vSuqfeHLiL1IELejUDnagWqP5pKWa9LtoemziGV 24 34-825-998-8579
+9.153000000000000E+001 BUILDING furiously dogged warthogs wake
quietly. daring realms believe carefull
550 Customer#0000000550 q5 gKwc7PBQOyd,H
17 27-938-997-6262 +7.270820000000000E+003 FURNITURE fluffily regular
instructions dazzle closely against the dependencies.
551 Customer#0000000551 holp1DkjYzznatSwjG 15
25-209-544-4006 -3.348900000000000E+002 MACHINERY fluffily regular
forges haggle quickly ironic foxes. slyly ironic packages use. blithely unusu

552 Customer#0000000552 EbjtaRaiok7eqbQ5VJi7q 2
12-669-784-2191 +1.353240000000000E+003 FURNITURE packages wake
furiously stealthily final packages. regular instructions boost blithely. slyly
regular accounts
553 Customer#0000000553 8tTlavJ sT 4
14-454-146-3094 +4.804570000000000E+003 BUILDING slyly regular foxes
eat. theodolites along the asymptotes use furiousl
554 Customer#0000000554 RluaguNRAJhYXmn,CWxcOC,Ly7
2 12-938-503-7317 +8.395570000000000E+003 HOUSEHOLD slyly
final foxes serve deposits. furiously regular pinto beans are a

555 Customer#0000000555 chm8jY6TfQ8CEnsypuL6azNZzkqGcZcO8
15 25-548-367-9974 +5.486520000000000E+003 BUILDING
blithely silent deposits across the carefully ironic deposits detect furiously express
theodolites. bold, even cour

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

8854660 8013349 F +2.674617600000000E+005 01/01/1992
3-MEDIUM Clerk#000072176 0 furiously ironic instruction

8862912	6730126 F	+1.930690500000000E+005	01/01/1992
1-URGENT	Clerk#000013189	0 carefully regular packages boost	
regularly qu	8865731	11227304 F	+2.091845500000000E+005 01/01/1992
5-LOW	Clerk#000030995	0 slyly final deposits above the furiously	

8867330	449923 F	+1.109640300000000E+005	01/01/1992
3-MEDIUM	Clerk#000009918	0 quickly bold ideas haggle slyly.	
carefully final packages boost. fina	8886755	11000207 F	+1.509150500000000E+005 01/01/1992
2-HIGH	Clerk#000095451	0 furiously quick foxes af	

8888837	813773 F	+1.801443300000000E+005	01/01/1992 4-NOT
SPECIFIED Clerk#000093312	0 carefully pending instructions wake		
slyly	8938115	1729660 F	+1.201625000000000E+005 01/01/1992

4-NOT SPECIFIED Clerk#000029050	0 furiously ironic pinto beans nag		
furiously. even, sly dependencies wak	8942020	14198426 F	+1.059932500000000E+005 01/01/1992

1-URGENT	Clerk#000088978	0 carefully ironic theo	
----------	-----------------	-------------------------	--

8949634	9165571 F	+1.125197200000000E+005	01/01/1992
4-NOT SPECIFIED Clerk#000099745	0 carefully special requests detect		
carefully furiou	8954278	9991201 F	+6.756193000000000E+004 01/01/1992

2-HIGH	Clerk#000057836	0 quickly pending excuses boost slyly	
across the quickly bold pinto beans? de			

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

256128994 4127495 377500 4 +1.000000000000000E+000
+1.522290000000000E+003 +8.000000000000000E-002
+0.000000000000000E+000 A F 01/04/1992 02/26/1992
01/29/1992 TAKE BACK RETURN TRUCK blithely pending foxes
detect c
256206145 3138305 888315 4 +1.900000000000000E+001
+2.551985000000000E+004 +6.000000000000000E-002
+3.000000000000000E-002 A F 01/04/1992 03/31/1992 01/25/1992
COLLECT COD MAIL express deposits wake furiously
256262117 12357932 107969 4 +4.000000000000000E+001
+7.957280000000000E+004 +5.000000000000000E-002
+3.000000000000000E-002 A F 01/04/1992 02/15/1992 01/11/1992
NONE MAIL final dependencies doze slyl
256269767 9687438 937448 1 +3.000000000000000E+000
+4.274850000000000E+003 +1.000000000000000E-002
+0.000000000000000E+000 R F 01/04/1992 03/18/1992
01/14/1992 COLLECT COD REG AIR quickly express deposi

256285894	12109600	609625	7	+2.400000000000000E+001
+3.861600000000000E+004	+2.000000000000000E-002			
+0.000000000000000E+000 R	F	01/04/1992	03/12/1992	
01/31/1992	NONE	SHIP	carefully sil	
256317955	15779298	779299	2	+1.000000000000000E+000
+1.376510000000000E+003	+1.000000000000000E-001			
+1.000000000000000E-002 A	F	01/04/1992	02/08/1992 01/28/1992	
TAKE BACK RETURN	RAIL	furiously special packa		

```

256351974 17944398 694450      3 +7.000000000000000E+000
+1.009050000000000E+004 +6.000000000000000E-002
+2.000000000000000E-002 R      F      01/04/1992 03/20/1992 01/12/1992
NONE      REG AIR      carefully close courts are furious
256408996 10955601 705632      2 +3.800000000000000E+001
+6.293028000000000E+004 +1.000000000000000E-001
+6.000000000000000E-002 A      F      01/04/1992 02/23/1992 01/18/1992
DELIVER IN PERSON      FOB      furiously pending requests cajole
carefully
256484994 16201774 701807      1 +3.600000000000000E+001
+6.029856000000000E+004 +6.000000000000000E-002
+3.000000000000000E-002 A      F      01/04/1992 02/10/1992 01/27/1992
TAKE BACK RETURN      MAIL      blithely unusua
256583525 52793 552794      5 +6.000000000000000E+000
+1.047474000000000E+004 +1.000000000000000E-002
+4.000000000000000E-002 R      F      01/04/1992 02/28/1992 01/08/1992
DELIVER IN PERSON      AIR      even Tiresias after the special, spec

```

10 record(s) selected.

Query Substitution Parameters

```

Power stream      Seed = 1205103618
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 1205103618 as a seed to the RNG
Q1 DELTA      84
Q2 SIZE      28
TYPE      NICKEL
REGION      AMERICA
Q3 SEGMENT      BUILDING
DATE      1995-03-12
Q4 DATE      1994-10-01
Q5 REGION      EUROPE
DATE      1994-01-01
Q6 DATE      1994-01-01
DISCOUNT      0.05
QUANTITY      24
Q7 NATION1      IRAN
NATION2      INDONESIA
Q8 NATION      INDONESIA
REGION      ASIA
TYPE      LARGE POLISHED BRASS
Q9 COLOR      navy
Q10 DATE      1993-04-01
Q11 NATION      ETHIOPIA
FRACTION      0.0000010000
Q12 SHIPMODE1      REG AIR
SHIPMODE2      TRUCK
DATE      1997-01-01
Q13 WORD1      pending
WORD2      deposits
Q14 DATE      1997-02-01
Q15 DATE      1995-03-01
Q16 BRAND      Brand#51
TYPE      PROMO PLATED
SIZE1      14
SIZE2      17
SIZE3      9
SIZE4      12
SIZE5      6
SIZE6      10
SIZE7      2
SIZE8      30
Q17 BRAND      Brand#15
CONTAINER      LG BAG
Q18 QUANTITY      315
Q19 BRAND1      Brand#22
BRAND2      Brand#32

```

```

BRAND3      Brand#11
QUANTITY1      3
QUANTITY2      16
QUANTITY3      23
Q20 COLOUR      firebrick
DATE      1995-01-01
NATION      ETHIOPIA
Q21 NATION      IRAN
Q22 I1      34
I2      11
I3      14
I4      17
I5      21
I6      20
I7      22

```

Throughput Stream = 1 Seed = 1205103619
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 1205103619 as a seed to the RNG

```

Q1 DELTA      92
Q2 SIZE      16
TYPE      TIN
REGION      MIDDLE EAST
Q3 SEGMENT      MACHINERY
DATE      1995-03-28
Q4 DATE      1997-05-01
Q5 REGION      MIDDLE EAST
DATE      1994-01-01
Q6 DATE      1994-01-01
DISCOUNT      0.02
QUANTITY      24
Q7 NATION1      BRAZIL
NATION2      ARGENTINA
Q8 NATION      ARGENTINA
REGION      AMERICA
TYPE      LARGE BURNISHED BRASS
Q9 COLOR      metallic
Q10 DATE      1994-01-01
Q11 NATION      CHINA
FRACTION      0.0000010000
Q12 SHIPMODE1      SHIP
SHIPMODE2      TRUCK
DATE      1997-01-01
Q13 WORD1      unusual
WORD2      deposits
Q14 DATE      1997-06-01
Q15 DATE      1997-10-01
Q16 BRAND      Brand#41
TYPE      SMALL BRUSHED
SIZE1      13
SIZE2      6
SIZE3      3
SIZE4      22
SIZE5      5
SIZE6      36
SIZE7      18
SIZE8      34
Q17 BRAND      Brand#12
CONTAINER      LG PKG
Q18 QUANTITY      313
Q19 BRAND1      Brand#24
BRAND2      Brand#15
BRAND3      Brand#11
QUANTITY1      9
QUANTITY2      17
QUANTITY3      30
Q20 COLOUR      pink
DATE      1993-01-01

```

NATION SAUDI ARABIA
 Q21 NATION BRAZIL
 Q22 I1 19
 I2 28
 I3 30
 I4 14
 I5 26
 I6 34
 I7 24

Throughput Stream = 2 Seed = 1205103620
 -- TPC TPC-H Parameter Substitution (Version 1.3.0)
 -- using 1205103620 as a seed to the RNG

Q1 DELTA 100
 Q2 SIZE 4
 TYPE COPPER
 REGION ASIA
 Q3 SEGMENT BUILDING
 DATE 1995-03-14
 Q4 DATE 1995-02-01
 Q5 REGION AFRICA
 DATE 1994-01-01
 Q6 DATE 1994-01-01
 DISCOUNT 0.08
 QUANTITY 25
 Q7 NATION1 ROMANIA
 NATION2 CHINA
 Q8 NATION CHINA
 REGION ASIA
 TYPE MEDIUM BRUSHED BRASS
 Q9 COLOR light
 Q10 DATE 1994-10-01
 Q11 NATION FRANCE
 FRACTION 0.0000010000
 Q12 SHIPMODE1 MAIL
 SHIPMODE2 TRUCK
 DATE 1997-01-01
 Q13 WORD1 unusual
 WORD2 deposits
 Q14 DATE 1997-09-01
 Q15 DATE 1995-07-01
 Q16 BRAND Brand#21
 TYPE ECONOMY ANODIZED
 SIZE1 24
 SIZE2 6
 SIZE3 40
 SIZE4 16
 SIZE5 48
 SIZE6 36
 SIZE7 18
 SIZE8 30
 Q17 BRAND Brand#14
 CONTAINER MED CASE
 Q18 QUANTITY 314
 Q19 BRAND1 Brand#31
 BRAND2 Brand#43
 BRAND3 Brand#55
 QUANTITY1 4
 QUANTITY2 18
 QUANTITY3 27
 Q20 COLOUR brown
 DATE 1996-01-01
 NATION IRAN
 Q21 NATION ROMANIA
 Q22 I1 25
 I2 34
 I3 23
 I4 18

I5 13
 I6 26
 I7 12

Throughput Stream = 3 Seed = 1205103621
 -- TPC TPC-H Parameter Substitution (Version 1.3.0)
 -- using 1205103621 as a seed to the RNG

Q1 DELTA 108
 Q2 SIZE 42
 TYPE BRASS
 REGION MIDDLE EAST
 Q3 SEGMENT HOUSEHOLD
 DATE 1995-03-30
 Q4 DATE 1997-09-01
 Q5 REGION AMERICA
 DATE 1994-01-01
 Q6 DATE 1994-01-01
 DISCOUNT 0.05
 QUANTITY 24
 Q7 NATION1 IRAQ
 NATION2 IRAN
 Q8 NATION IRAN
 REGION MIDDLE EAST
 TYPE MEDIUM PLATED STEEL
 Q9 COLOR ivory
 Q10 DATE 1993-07-01
 Q11 NATION ROMANIA
 FRACTION 0.0000010000
 Q12 SHIPMODE1 TRUCK
 SHIPMODE2 MAIL
 DATE 1997-01-01
 Q13 WORD1 unusual
 WORD2 deposits
 Q14 DATE 1997-12-01
 Q15 DATE 1993-03-01
 Q16 BRAND Brand#51
 TYPE STANDARD PLATED
 SIZE1 34
 SIZE2 27
 SIZE3 1
 SIZE4 50
 SIZE5 12
 SIZE6 37
 SIZE7 44
 SIZE8 2
 Q17 BRAND Brand#11
 CONTAINER MED BAG
 Q18 QUANTITY 312
 Q19 BRAND1 Brand#33
 BRAND2 Brand#35
 BRAND3 Brand#54
 QUANTITY1 9
 QUANTITY2 19
 QUANTITY3 23
 Q20 COLOUR medium
 DATE 1995-01-01
 NATION UNITED STATES
 Q21 NATION IRAQ
 Q22 I1 20
 I2 32
 I3 18
 I4 31
 I5 13
 I6 17
 I7 16

Throughput Stream = 4 Seed = 1205103622
 -- TPC TPC-H Parameter Substitution (Version 1.3.0)

-- using 1205103622 as a seed to the RNG

Q1 DELTA 116
Q2 SIZE 29
TYPE NICKEL
REGION ASIA
Q3 SEGMENT BUILDING
DATE 1995-03-16
Q4 DATE 1995-06-01
Q5 REGION ASIA
DATE 1994-01-01
Q6 DATE 1994-01-01
DISCOUNT 0.03
QUANTITY 24
Q7 NATION1 CANADA
NATION2 BRAZIL
Q8 NATION BRAZIL
REGION AMERICA
TYPE MEDIUM ANODIZED STEEL
Q9 COLOR goldenrod
Q10 DATE 1994-05-01
Q11 NATION GERMANY
FRACTION 0.0000010000
Q12 SHIPMODE1 RAIL
SHIPMODE2 MAIL
DATE 1993-01-01
Q13 WORD1 unusual
WORD2 packages
Q14 DATE 1993-03-01
Q15 DATE 1995-10-01
Q16 BRAND Brand#41
TYPE MEDIUM POLISHED
SIZE1 12
SIZE2 48
SIZE3 28
SIZE4 31
SIZE5 42
SIZE6 29
SIZE7 21
SIZE8 2
Q17 BRAND Brand#13
CONTAINER MED PKG
Q18 QUANTITY 313
Q19 BRAND1 Brand#35
BRAND2 Brand#13
BRAND3 Brand#53
QUANTITY1 5
QUANTITY2 20
QUANTITY3 30
Q20 COLOUR turquoise
DATE 1993-01-01
NATION KENYA
Q21 NATION CANADA
Q22 I1 18
I2 17
I3 10
I4 20
I5 33
I6 32
I7 13

Throughput Stream = 5 Seed = 1205103623
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 1205103623 as a seed to the RNG

Q1 DELTA 63
Q2 SIZE 17
TYPE TIN
REGION AFRICA
Q3 SEGMENT HOUSEHOLD

DATE 1995-03-01
Q4 DATE 1993-03-01
Q5 REGION EUROPE
DATE 1995-01-01
Q6 DATE 1995-01-01
DISCOUNT 0.08
QUANTITY 25
Q7 NATION1 SAUDI ARABIA
NATION2 ROMANIA
Q8 NATION ROMANIA
REGION EUROPE
TYPE SMALL POLISHED STEEL
Q9 COLOR firebrick
Q10 DATE 1993-02-01
Q11 NATION SAUDI ARABIA
FRACTION 0.0000010000
Q12 SHIPMODE1 AIR
SHIPMODE2 MAIL
DATE 1993-01-01
Q13 WORD1 unusual
WORD2 packages
Q14 DATE 1993-07-01
Q15 DATE 1993-07-01
Q16 BRAND Brand#21
TYPE PROMO ANODIZED
SIZE1 31
SIZE2 19
SIZE3 15
SIZE4 4
SIZE5 2
SIZE6 17
SIZE7 33
SIZE8 25
Q17 BRAND Brand#15
CONTAINER JUMBO CASE
Q18 QUANTITY 315
Q19 BRAND1 Brand#42
BRAND2 Brand#51
BRAND3 Brand#43
QUANTITY1 10
QUANTITY2 10
QUANTITY3 26
Q20 COLOUR grey
DATE 1997-01-01
NATION EGYPT
Q21 NATION VIETNAM
Q22 I1 14
I2 28
I3 15
I4 31
I5 34
I6 24
I7 20

Appendix D: Driver Source Code

load_line_uf

```
#!/bin/ksh
RFpair=$1;
db2 connect to tpcd
db2 "load from /data/update/lineitem.tbl.u${RFpair}.new of del modified by coldel| fastparse messages /dev/null replace into TPCDTEMP.LINEITEM_new nonrecoverable;"
db2 commit;
db2 connect reset
db2 terminate
```

load_orders_uf

```
#!/bin/ksh
RFpair=$1;
db2 connect to tpcd
db2 "load from /data/update/orders.tbl.u${RFpair}.new of del modified by coldel| fastparse messages /dev/null replace into TPCDTEMP.ORDERS_new nonrecoverable;"
db2 commit;
db2 connect reset
db2 terminate
```

makefile

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

#LOCAL=tpcd

```
BASE=$(HOME)/sqllib
COMPILE_FLAGS=-c -DSQLAIX -DLINUX -I$(BASE)/include -g
#COMPILE_FLAGS=-c -DSQLAIX -I$(BASE)/include -g
# if using an installed db2 image use the 2nd link_flags value
LINK_FLAGS=-o ${@} -L$(BASE)/lib -ldb2
#LINK_FLAGS=-o ${@} -L$(BASE)/lib -Xlinker $(BASE)/lib
$(BASE)/lib/libdb2.so
#LINK_FLAGS=-o ${@} -L/usr/lpp/db2_05_00/lib -ldb2
COMPILER=cc
LIB_LINKER=ld
LIB_LINK_FLAGS=-o ${@} -H512 -T512 -bE:${@}.exp -L$(BASE)/lib -ldb2 -lc
```

cleanup :

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

all : tpcdbatch

```
tpcdbatch.c : tpcdbatch.sqc
@echo -e 'connect to $(TPCD_DBNAME) \n prep tpcdbatch.sqc
BINDFILE PACKAGE ISOLATION RR BLOCKING ALL OPTLEVEL 1
DATETIME ISO \n connect reset \n terminate \n' | db2 -c +p -v +t
```

```
tpcdUF.c : tpcdUF.sqc
@echo -e 'connect to $(TPCD_DBNAME) \n prep tpcdUF.sqc
BINDFILE PACKAGE ISOLATION RS BLOCKING ALL OPTLEVEL 1
DATETIME ISO \n connect reset \n terminate \n' | db2 -c +p -v +t
```

```
tpcdbatch : tpcdUF.c tpcdbatch.c
$(COMPILER) $(COMPILE_FLAGS) tpcdUF.c
$(COMPILER) $(COMPILE_FLAGS) tpcdbatch.c
$(COMPILER) $(LINK_FLAGS) tpcdUF.o tpcdbatch.o
```

ploadUF1

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

ploadUF2

```
#!/bin/ksh
RFpair=$1;
db2 connect to tpcd
db2 "load from delete.new.$RFpair of del modified by coldel| fastparse messages /usr/tpcd/temp/deleteUF.msg replace into TPCDTEMP.ORDERS_DEL nonrecoverable data buffer 256 partitioned db config mode load_only part_file_location /flatfiles/300GB_8mIn_UF_flatfiles/cleanFiles;"
db2 commit;
db2 connect reset
db2 terminate
```

preloadUF

```
#!/bin/ksh
# Please indicate which pair you want to use for the preloading of the
# update functions. This pair can NOT be used for the actual benchmark.
# In general we pre-load the pair which the largest number. For example
# if we created 12 pairs, we'll preload pair 12. This number is the parameter
# passed to ploaduf1 and ploaduf2 to load the data.
```

toolsDir=/home/tpch/tpcd/tools

```
${toolsDir}/ploaduf1 18
${toolsDir}/ploaduf2 18
```

```
db2 "connect to tpcd"
db2 "RUNSTATS ON TABLE TPCDTEMP.LINEITEM_NEW WITH DISTRIBUTION AND DETAILED INDEXES ALL"
db2 "RUNSTATS ON TABLE TPCDTEMP.ORDERS_NEW WITH DISTRIBUTION AND DETAILED INDEXES ALL"
db2 "RUNSTATS ON TABLE TPCDTEMP.ORDERS_DEL WITH DISTRIBUTION AND DETAILED INDEXES ALL"
db2 "terminate"
```

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("ls -l >> $runDir/sysunused$runNum");
    system("echo \"The following volume groups are currently online\" >>
    $runDir/sysunused$runNum");
    $curTs = `perl gettimestamp "long"`;
    system("echo \"$curTs\" >> $runDir/sysunused$runNum");
    system("ls -l >> $runDir/sysunused$runNum");
    # show the disks that are used/unused
    #system("getdisks \"Before the start of the Power Test\"");

}
else
{
    # for all other platforms
    system("echo \"Assume that all portions of the system are used >>
    $runDir${delim}sysunused$runNum");
}

&getConfig("p");
if ( $rootPriv eq "yes" )
{
    # get the o/s tuning parameters...currently AIX only and only if your
    # user has root privileges to run this
    &getOSTune("p");
}
if ($gatherstats eq "on")
{
    # gather vm io and net stats
    if ($platform eq "aix" || $platform eq "sun" || $platform eq "ptx" ||
        $platform eq "hp" || $platform eq "linux")
    {
        # gather vmstats and iostats (and net stats if in mpp mode)
        system("perl getstats p &");
    }
    else
    {
        print "Stats gather not set up for current platform $platform\n";
    }
}

# print to screen what type of run is running and set variables to run
# the query and update streams in parallel
if ($runUF ne "UF")
{
    $semcontrol = "off";
    print "Beginning power stream....no update functions\n";

    $streamEx = "";
    $streamExNT = "";
}
else
{
    $semcontrol = "on";
    print "Beginning power stream....with update functions\n";
    if ( $platform eq "nt" )
    {
        $streamExNT = "start /b";
        $streamEx = "";
    }
}

```

```

}
else
{
    $streamExNT = "";
    $streamEx = "&";
}
}

# bbe This new line (below) runs queries for power test

print "Starting tpcdbatch...\n";
$ret=system("$streamExNT $auditDir${delim}auditruns${delim}tpcdbatch -d
$dbname -f $runDir${delim}qtextpow.sql -r on -b on -s $sf -u p1 -m $inlistmax -n
0 -p $semcontrol $streamEx");

if ( $runUF eq "UF" )
{
    $ret2 = system("$auditDir${delim}auditruns${delim}tpcdbatch -d $dbname -f
    $runDir${delim}qtextquf.sql -r on -b on -s $sf -u p2 -m $inlistmax -n 0");
}
else
{
    $ret2 = 0; # If UFs were not running, then the stream cannot fail
}

if (($ret2 == 0) && ($ret == 0))
{
    print "Power stream completed successfully.\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 ( $mode eq "mpp")
{
    $prefix="rah \"";
    $a="\"";
}
else {
    $prefix = "";
    $a = "";
}

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")
        {
            $src=`$prefix perl5 $auditDir${delim}tools${delim}zap $a"-f$a"
            $a"^sar$a" `;
            $src=`$prefix perl5 $auditDir${delim}tools${delim}zap $a"-f$a"
            $a"^sadc$a" `;
        }
        else
        {
            $src=`$prefix perl5 $auditDir${delim}tools${delim}zap $a"-f$a"
            $a"^vmstat$a" `;
            $src=`$prefix perl5 $auditDir${delim}tools${delim}zap $a"-f$a"
            $a"^iostat$a" `;
        }
    }
}

```



```

Src= ` $prefix iostat $suffix `;

}

Src= ` $prefix perl5 $auditDir$ {delim} tools$ {delim} zap $a"-f$a"
$a"^getstats$a" `;

}

}

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 dependency 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 'calcmetrics.pl';
if ( $runUF eq "UF" )
{
    system("perl calcmetrics.pl UF");
}
else
{
    system("perl calcmetrics.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.metrics");

if ($runUF eq "UF") {

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

# 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 VM database 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");
}

```

runthroughput

```

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

# usage runthroughput [UF]
# where UF is the optional parameter that says to run the throughput test
# with the update functions. By default, the update functions are not
# run
# If UF is not supplied and a number is supplied, then that number is taken
# as the number of concurrent throughput streams to run. This is also optional

push(@INC, split(':', $ENV{'PATH'}));

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

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

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

@reqVars = ("TPCD_AUDIT_DIR",
            "TPCD_RUN_DIR",
            "TPCD_DBNAME",
            "TPCD_RUNNUMBER",
            "TPCD_SF",
            "TPCD_PLATFORM",
            "TPCD_PATH_DELIM",
            "TPCD_PRODUCT",
            "TPCD_AUDIT",
            "TPCD_PHYS_NODE",
            "TPCD_MODE",
            "TPCD_ROOTPRIV",
            "TPCD_NUMSTREAM");

&setVar(@reqVars, "ERROR");

if (length($ENV{"TPCD_LOG_DIR"}) <= 0)
{
    $ENV{"TPCD_LOG_DIR"} = "NULL";
}

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

```

```

$gatherstats=$ENV{"TPCD_GATHER_STATS"};
$logDir=$ENV{"TPCD_LOG_DIR"};
$rootPriv=$ENV{"TPCD_ROOTPRIV"};
$mode=$ENV{"TPCD_MODE"};

$spath="$auditDir${delim}auditruns";

if (( $mode eq "uni" ) || ( $mode eq "smp" ))
{
    $all_in="once";
    $all_pn="once";
    $once="once";
}
else
{
    $all_in="all_in";
    $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: $!";
$curTs = `perl gettimestamp "long"`;
print MISO "Timestamp and isolation level of tpcdbatch before throughput run at
: $curTs\n";
close(MISO);

if ( $product eq "pe" )
{
    system("db2 \\"connect to $dbname\\"; db2 \\"select
name,creator,valid,unique_id,isolation from sysibm.sysplan where name like
'TPCD%'\" >> $runDir${delim}miso$runNum ");
}
else
{
    &verifyTPCDBatch("$misofile", "$dbname");
}

# kick off the script that will monitor for the database applications during
# the running of the throughput tests. This will quit when the mtinterX.metrics
# (where X=runnumber) file has been created.

# set variables to run streams in parallel
if ( $platform eq "nt" )
{
    $streamExNT = "start /b";
}

```

```

$streamEx = "";
}
else
{
    $streamExNT = "";
    $streamEx = "&";
}
if ( $platform eq "aix" || $platform eq "sun" || $platform eq "nt" || $platform eq
"hp" || $platform eq "linux")
{
    system("$streamExNT perl watchstreams $streamEx");
}
else
{
    die "platform not supported, can't start watchstreams in background";
}

# show the disks that are used/unused
#if ($platform eq "aix")
#{
#    system("getdisks \"Before the start of the Throughput Test\");
#}

if ($gatherstats eq "on")
{
    # gather vm io and net stats
    if ($platform eq "aix" || $platform eq "sun" || $platform eq "ptx" || $platform eq
"hp" || $platform eq "linux")
    {
        # gather vmstats and iostats (and net stats if in mpp mode)
        system("perl getstats t &");
    }
    else
    {
        print "Stats gather not set up for current platform $platform\n";
    }
}

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

$loopStream=1;

for ( $loopStream = 1; $loopStream <= $numStream; $loopStream++)
{
    print "starting stream $loopStream\n";
    system("echo Executing stream $loopStream out of $numStream.");
    # run the queries
    if ( $platform eq "aix" || $platform eq "sun" || $platform eq "nt" || $platform eq
"ptx" ||
        $platform eq "hp" || $platform eq "linux")
    {
        system("$streamExNT $path${delim}tpcdbatch -d $dbname -f
$runDir${delim}qtextt$loopStream.sql -r on -b on -s $sf -u t1 -m $inlistmax -n
$loopStream $streamEx");
    }
    else
    {
        die "platform $platform not supported yet";
    }
}

# run the update function stream....this will wait until the queries have
# completed to kick off the updates
print "starting update stream\n";

if ($runUF eq "no") {
    $ret=system("$auditDir${delim}auditruns${delim}tpcdbatch -d $dbname -f
$runDir${delim}quft.sql -r on -b on -s $sf -u t2 -m $inlistmax -n $numStream");
}
else {
    $ret=system("$auditDir${delim}auditruns${delim}tpcdbatch -d $dbname -f
$runDir${delim}quft.sql -r on -b on -s $sf -u t2 -m $inlistmax -n $numStream");
}

```

```

}
print "update stream done\n";

&getConfig("t");
if ( $rootPriv eq "yes" )
{
    # get the o/s tuning parameters...currently AIX only and only if your
    # user has root privileges to run this
    &getOSTune("t");
}

#if ($platform eq "aix")
#{
    # show the disks that are used/unused
    #    system("getdisks \"After the completion of the Throughput Test\");
    #}
if ($gatherstats eq "on")
{
    # gather vm io and net stats
    if ($platform eq "aix" || $platform eq "sun" || $platform eq "ptx" || $platform eq
"linux")
    {
        # kill the stats that were being gathered
        if ($platform eq "ptx")
        {
            $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}strent*","$runDir${delim}mstrent$runNum");
#(NOTE: there is a dependency that this mstrent file exist before the
# calmetrics.pl script is called, both because it is used as input for
# calmetrics.pl, and because the output from calmetrics 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.metrics");

if ($runUF ne "no") {

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

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");
$src=&dodb_noconn("db2 get db cfg for $dbname | grep -i log >>
$runDir$ {delim} endLog.Info", $all_ln);
if ( $logDir ne "NULL" )
{
    $src=&dodb_noconn("$dircmd $logDir >>
$runDir$ {delim} endLog.Info", $all_ln);
}

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

#Create Catalog info
$src = system("perl catinfo.pl p");

if ( $src != 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

```

```

{
    $stesttype=$_[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
{
    $stesttype=$_[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 verifyTPCDBbatch
{
    $logfile=$_[0];
    $dbname=$_[1];
    $file="verifytpcdbbatch.clp";
    open(VERTBL, ">$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");
}

```

tpcdbbatch.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
*   18 mar 02 ken Increased version to 5.7

```

```

*****
*****/

/** 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 "5.7"

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

#define TPCDBATCH_MAX_COLS 100       /* @d30369 tlg */

#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 */

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

#define PARALLEL_UPDATES 1

#ifndef PARALLEL_UPDATES
#define UF1OUTSTREAMPATTERN "%s%cuf1.%02d.%d.out"
#define 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.%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_LEN 17
#if defined(SQLUNIX) || defined(SQLAIX) || defined(SQLHP)
#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 "0"
#define READMODE "r0"
#define WRITEMODE "w0"
#define APPENDMODE "a0"
#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 tlg */
/* 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) || defined (SQLHP))
/*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 */

```

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 msecme 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 counting 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_sqllda 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 ufl
* 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 function to output files, not
* stderr

```

```

* 10 nov 97 jen jenCI Updated version number to 2.3
* Added handling of TPCD_CONCURRENT_INSERTS. Change
control of
* chunk processing to use the concurrent_inserts value as the
* control. Now the inserts will be run in
TPCD_CONCURRENT_INSERTS
* sets, each having concurrent_inserts/
* 13 nov 97 jen jen DEADLOCK. Fixed bug that Alex found where deadlock
count
* (maxwait) was incremented on every execution of the stmt as
* opposed to just when deadlock really happened.
* 14 nov 97 jen jenSEM - fix up error reporting on semaphore failure
* sem_op now returns failure to caller so caller can report where
* failure has happened.
* Forced dbname to be upper case, and 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 gettimeofday 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_ufl and load_ufl2 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 calcmetricp.pl. This
*      is a result of the runpower being split into two streams and thus
*      tpcdbatch not having access to all data.
* 21 jul 99 bbeaton The start time for runpower UF2 now does not start until after
the query stream is complete so that its wait time is not included
*      NOTE: The wait time that the first UF1 in runthroughput still
*      includes the wait period that occurs waiting on queries.
* 18 mar 02 kentond removed the need for list files. Instead of using the *.list
files to determine the name of the output files, the tags for the
*      source sql files are used.
* 07 Jan 04 jregier Added Christian's change to the create_semaphore function,
*      simply checks for the existence of the semaphore first and
*      removes it if it wasn't properly cleaned up previously.
*****
*****/

/* 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 */
    /* kjd715 */
    /* char str_file_name[256]; /* output filename */
    /* kjd715 */
    char infile[256]; /* input filename */
    int intStreamNum; /* integer version of stream number */
    int a_commit; /* auto-commit flag */
    int short_time; /* time interval flag */
    int update;
    int outfile;
};

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

/* Structure used to hold precision for decimal numbers */
/*****
*****/

struct declen
{ /* kmw */
    unsigned char m; /* # of digits left of decimal */
    unsigned char n; /* # of digits right of decimal */
};

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

/* Structure containing control flags passed between functions */
/*****
*****/

struct ctrl_flags
{ /* @d25594 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);
#define 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 */
/* jen LONG */
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 sqllda *sqllda; /* 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 sqllda_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
Timer_struct temp_time_struct; /* holds end time value to be copied into
start_time of next query bbeaton */

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

/*****
**/
/* Start main program processing. */
/*****
**/
int main(int argc, char *argv[])
{
/* kjd715 */
/*struct comm_line_opt c_1_opt = { "\0", "\0", 0, 1, 0, 0, 0 };*/ /* kjd715 */
struct comm_line_opt c_1_opt = { "\0", 0, 1, 0, 0, 0 };
/* kjd715 */
/* command line options */
Timer_struct start_time; /* start point for elapsed time */

struct stmt_info s_info = { -1, -1, 0, 1, -1, -1, "\0", "\0", "\0", 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_stamp;
}
/* 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 tjc */
    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 tjc */
                    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")));
            }
        }
    } while (1);
}

```

```

/* 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 tlg */
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 tlg */
}

```

```

} 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 statements
with no SQL, there's nothing to process...Exit TPCDBATCH */

else /* @d23684 tlg */
    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->sqlcl; loopvar++)
    {
        switch(sqlda->sqlvar[loopvar].sqltype)
        {
            case SQL_TYP_INTEGER: /* INTEGER */
            case SQL_TYP_NINTEGER:
                if((sqlda->sqlvar[loopvar].sqldata=
                    (TPCDBATCH_CHAR *)malloc(sizeof(sqlint32))) == NULL)
                    mem_error("allocating INTEGER");
                break;
            case SQL_TYP_BIGINT: /* BIGINT */ /*kmwBIGINT*/
            case SQL_TYP_NBIGINT:
                /*#ifdef SQLWINT */
                /* if((sqlda->sqlvar[loopvar].sqldata=
                /* (TPCDBATCH_CHAR *)malloc(sizeof(_int64))) == NULL)*/
                /*#else */
                if((sqlda->sqlvar[loopvar].sqldata=
                    (TPCDBATCH_CHAR *)malloc(sizeof(sqlint64))) == NULL)
                    /*#endif*/
                    mem_error("allocating BIGINT");
                break;
            case SQL_TYP_CHAR: /* CHAR */
            case SQL_TYP_NCHAR:
                if((sqlda->sqlvar[loopvar].sqldata=
                    (TPCDBATCH_CHAR *)calloc(256, sizeof(char))) == NULL)
                    mem_error("allocating CHAR/VARCHAR");
                break;
            case SQL_TYP_VARCHAR: /* VARCHAR */
            case SQL_TYP_NVARCHAR:
                if((sqlda->sqlvar[loopvar].sqldata=
                    (TPCDBATCH_CHAR *)calloc(4002, sizeof(char))) == NULL)
                    mem_error("allocating CHAR/VARCHAR");

```

```

break;
case SQL_TYP_LONG: /* LONG VARCHAR */
case SQL_TYP_NLONG:
if ((sqlda->sqlvar[loopvar].sqldata=
(TPCDBATCH_CHAR *)calloc(32702,sizeof(char))) == NULL)
mem_error("allocating VARCHAR/LONG VARCHAR");
break;
case SQL_TYP_FLOAT: /* FLOAT */
case SQL_TYP_NFLOAT:
if ((sqlda->sqlvar[loopvar].sqldata=
(TPCDBATCH_CHAR *)malloc(sizeof(double))) == NULL)
mem_error("allocating FLOAT");
break;
case SQL_TYP_SMALL: /* SMALLINT */
case SQL_TYP_NSMALL:
if ((sqlda->sqlvar[loopvar].sqldata=
(TPCDBATCH_CHAR *)malloc(sizeof(short))) == NULL)
mem_error("allocating SMALLINT");
break;
case SQL_TYP_DECIMAL: /* DECIMAL */
case SQL_TYP_NDECIMAL:
if ((sqlda->sqlvar[loopvar].sqldata=
(TPCDBATCH_CHAR *)malloc(20)) == NULL)
mem_error("allocating DECIMAL");
break;
case SQL_TYP_CSTR: /* VARCHAR (null terminated) */
case SQL_TYP_NCSTR:
if ((sqlda->sqlvar[loopvar].sqldata=
(TPCDBATCH_CHAR *)calloc(4001,sizeof(char))) == NULL)
mem_error("allocating CHAR/VARCHAR");
break;
case SQL_TYP_DATE: /* DATE */
case SQL_TYP_NDATE:
if ((sqlda->sqlvar[loopvar].sqldata=
(TPCDBATCH_CHAR *)calloc(13,sizeof(char))) == NULL)
mem_error("allocating DATE");
break;
case SQL_TYP_TIME: /* TIME */
case SQL_TYP_NTIME:
if ((sqlda->sqlvar[loopvar].sqldata=
(TPCDBATCH_CHAR *)calloc(11,sizeof(char))) == NULL)
mem_error("allocating TIME");
break;
case SQL_TYP_STAMP: /* TIMESTAMP */
case SQL_TYP_NSTAMP:
if ((sqlda->sqlvar[loopvar].sqldata=
(TPCDBATCH_CHAR *)calloc(29,sizeof(char))) == NULL)
mem_error("allocating TIMESTAMP");
break;
}
if ((sqlda->sqlvar[loopvar].sqlind=
(short *)calloc(1,sizeof(short))) == NULL)
mem_error("allocating indicator");
}
sqlda_allocated = 1; /* fix free() problem on NT
wlc 090597 */
return; /* allocate_sqlda */
}

```

```

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

```

```

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

int col_type; /* Type of column */

```

```

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

```

```

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

```

```

for (col=0; col<sqlda->sqlld; col++) /* Loop through column count */
{

```

```

col_type=sqlda->sqlvar[col].sqltype; /* @d22817 tjc */

```

```

if ((sqlda->sqlvar[col].sqlind)) /* @d30369 tjc */
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 tjc */

```

```

((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].sqlld);m;
n=(*(struct declen *)&sqlda->sqlvar[col].sqlld);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)
    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);
        }
#ifdef (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 : %.8s\n", 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");

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

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

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

fprintf(stderr, "\nCritical SQLCODE. Exiting TPCDBATCH\n");
exit(-1);
}
}
return (temp_sqlca.sqlcode);
} /* error_check */

/*****
*****/
/* Displays a help screen */
/*****
*****/
void display_usage()
{
    printf("\ntpcdbatch -- version %s", TPCDBATCH_VERSION);
    printf("\n\nSyntax is:\n");
    printf("tpcdbatch [-d dbname] [-f file_name] [-l file_name] [-r on/off]");
    printf("\n          [-v on/off] [-b on/off] [-u p/t1/t2]");
    printf("\n          [-s scale_factor] [-n stream_num] [-m inlistmax] [-h]n");
    printf("\n where: -d Database name");
    printf("\n          Default - dbname set in $DB2DBDFT");
    printf("\n          -f Input file containing SQL statements");
    printf("\n          Default - stdin");
    printf("\n          -r Create set of output files containing query results");
    printf("\n          Default - off");
    printf("\n          -v Verbose. Sends information to stderr during");
    printf("\n          query processing");
    printf("\n          Default - off");
    printf("\n          -b Process groups of statements as blocks");
    printf("\n          instead of individually.");
    printf("\n          Default - off");
    printf("\n          -u Update streams: p - for power test");
    printf("\n          t - for throughput test without");
    printf("\n          UFs (run this instead of t2)");
    printf("\n          t1 - for throughput test step 1");
    printf("\n          only running queries");
    printf("\n          t2 - for throughput test step 2");
    printf("\n          running update functions");
    printf("\n          -s Scale factor");
    printf("\n          Default - 0.1");
    printf("\n          -n Stream number");
    printf("\n          Default - 0");
    printf("\n          Qualification - -1");
}

```

```

    printf("\n          Power - 0");
    printf("\n          Throughput - >= 1 (actual number depends on the current
query stream");
    printf("\n          -m Maximum number of keys to delete at a time");
    printf("\n          Default - 400");
    printf("\n          -h Display this help screen");
    printf("\n          -p turns smeaphores on or off");
    printf("\n          Default - off");

    printf("\n\nControl statements specifying output and performance details");
    printf("\n\nCan be included before SQL statements; they will apply for");
    printf("\n\nthat and subsequent statements until updated.");

    printf("\n\nSyntax:  --SET <control option> <value>");
    printf("\n\n option  value  default");
    printf("\n\nROWS_FETCH  -l to n   -l (all rows fetched from answer set)");
    printf("\n\nROWS_OUT    -l to n   -l (all fetched rows sent to output)");
    printf("\n\n--TAG      tag      (user specified tag name for sequence#)");
    printf("\n\n--COMMENT  comment   (user specified comments for
output)");
    printf("\n\nNote: All statements executed with ISOLATION LEVEL RR");
    printf("\n\nand 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 authentic_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_1_opt->infile,argv[++loopvar]);
                    break;
                /* kjd715 */
                case 'l' :
                case 'L' :    loopvar+=1;
                                /*

            strcpy(g_struct->c_1_opt->str_file_name,argv[++loopvar]);
                                */

                    break;
                /* kjd715 */
                case 'r' :    /* @d26350 tlg */
                case 'R' :
                    if (!strcmp(uppercase(argv[++loopvar]),"ON"))
                        g_struct->c_1_opt->outfile=1;
                    else
                        g_struct->c_1_opt->outfile=0;
                    break;

                case 'd' :    /* @d26350 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_1_opt->update=-1; /* init to invalid number */
                    if (!strcmp(uppercase(argv[++loopvar]),"P1"))
                        g_struct->c_1_opt->update=1; /* power query stream */
                    if (!strcmp(uppercase(argv[loopvar]),"P2"))
                        g_struct->c_1_opt->update=3; /* power update with updates */
                    if (!strcmp(uppercase(argv[loopvar]),"P"))
                        g_struct->c_1_opt->update=4; /* power update without updates */
                    if (!strcmp(uppercase(argv[loopvar]),"T1"))
                        g_struct->c_1_opt->update=0; /*throughput query stream */
                    if (!strcmp(uppercase(argv[loopvar]),"T2"))
                        g_struct->c_1_opt->update=2; /* throughput update with updates */
                    if (!strcmp(uppercase(argv[loopvar]),"T"))
                        g_struct->c_1_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_1_opt->intStreamNum = atoi(argv[++loopvar]);
                    break;

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

                case 'h':

```

```

case 'H' :
    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]);
#endif
#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 tpdbatch 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 tpdbatch\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\n\nor in environment variable DB2DBDFT.");
        display_usage();
    }
    else
        strcpy(dbname, testptr);
}

/* kjd715 */
/*
if (g_struct->c_l_opt->outfile) &&
    !strcmp(g_struct->c_l_opt->str_file_name, "0")) {
    fprintf(stderr, "\nMust specify input file for statement list.\n");
    display_usage();
}
*/
/* kjd715 */
}

/*****
/* Converts DECIMAL values to ASCII text */
/*****
int sqlrxd2a(
    /*kmw*/
    /* C++ */char *decptr,
    /* C++ */char *asciiptr,
    short prec,

```



```

short scal)

{ /* */
int allzero = TRUE;
/* C++ */char *srcptr;
unsigned char sign;
/* C++ */char *targptr, decimal_point = '.';
int rc = 0; /*kmw*/
int tmpint, src_nibble;
int count, j, limit[3];

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

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

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

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

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

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

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

void init_setup(int argc, char *argv[], struct global_struct *g_struct)
{

```

```

int connect=0;
#ifndef SQLWINT
char *pid;
#endif
char temparray[256]="\0";
int loopvar=0;
FILE *updateFP;
FILE *fpSeed;
char file_name[256] = "\0";
short seedEntry;
long lSeed;
int i;

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

/*****
*****/
/* Start the mainline report processing. */
/*****
*****/
if (!strcmp(g_struct->c_l_opt->infile,"")) {
    instream=stdin;
}
else {
    instream=NULL;
    if ( (instream = fopen(g_struct->c_l_opt->infile, READMODE)) == NULL ) {
        /* kjd715 */
        fprintf(outstream, "XXThe input file could not be opened.\n\n");
        /* kjd715 */
        fprintf(stdout, "Make sure that the filename is correct.\n");
        fprintf(stdout, "filename = %s\n", g_struct->c_l_opt->infile);
        exit(-1);
    } /* open the input file if specified */
}

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

/* IMPORT (end) */

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

```

```

    {
        fprintf(stderr, "\nThe TPCD_NUMSTREAM environment variable is not
set");
        fprintf(stderr, "....exiting\n");
        exit(-1);
    }

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

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

sprintf(file_name, "%s%sauditruns%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, "%d\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%sstrentuf.%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%spspstrcntuf.%s", g_struct->run_dir,
env_tpcd_path_delim, g_struct->file_time_stamp);
        break;
    case (1):
        /* power query stream */
        sprintf(file_name, "%s%spspstrcnt%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%sstrent%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*/
}

#ifdef 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 sqllda *sqllda, 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 < sqllda->sqlld; col++) {
        col_name_length = sqllda->sqlvar[col].sqlname.length;
        col_type = sqllda->sqlvar[col].sqltype;
        col_width = sqllda->sqlvar[col].sqlllen;
        strncpy(col_name, (char *)sqllda->sqlvar[col].sqlname.data, col_name_length);

        switch (col_type)
        {
            case SQL_TYP_SMALL:
            case SQL_TYP_NSMAIL: /* @d30369 tjc */
                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 tjc */
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 stop */
    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_1_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_1_opt->intStreamNum >= 0) &&
    (g_struct->c_1_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_1_opt->intStreamNum > 0) &&
    (g_struct->c_1_opt->update != 2) )
{
    fprintf( outstream, "Stream number =
%d\n", g_struct->c_1_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===== \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 (which can handle the */
    /* os differences on its own */

#ifdef (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 tlg */
{
    int loopvar;

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

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

/*****
/* processing to run the insert update function */
*****/
void runUF1 ( struct global_struct *g_struct, int updatePair )
{
    char statement[3000];
    char sourcedir[256];

    int split_updates = 2; /* no. of ways update records are split */
    int concurrent_inserts = 2; /* jenCI no of concurrent updates to be */
/* jenCI run at once */
    int loop_updates = 1; /* jenCI no of updates to be run in one */
/* jenCI "concurrent" invocation. should */
/* jenCI be split_updates / concurrent_inserts */

    int i;
    int streamNum;
#ifdef SQLWINT
    /* PROCESS_INFORMATION childprocess[100]; */
    char commandline[256];
    HANDLE su_hSem;
    char UF1_semaphore[256];
#else
    int childpid[100];
    int su_semidx; /* 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);
}
/*semctl(su_semid, 0, IPC_RMID, 0);*/ /*mujib*/
#else /* SQLWINT */
    sprintf (UF1_semfile, "%s.%s.UF1.semfile", env_tpcd_dbname, env_user);
    su_hSem = CreateSemaphore(NULL, 0,
        concurrent_inserts, /*jenCI*/
        (LPCTSTR)(UF1_semfile));
if (su_hSem == NULL)
{

```

```

fprintf(stderr,
    "CreateSemaphore (ready semaphore) failed, GetLastError: %d,
quitting\n",
    GetLastError());
    exit(-1);
}
#endif /* SQLWINT */
if (verbose) fprintf(stderr, "Semaphore created successfully!\n");

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

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

if( (outstream = fopen(outstreamfilename, APPENDMODE)) == NULL )
{
    fprintf(stderr, "\nThe output file could not be opened. ");

```

```

    fprintf(stderr,"Make sure that the filename is correct.\n");
    fprintf(stderr,"filename = %s\n",outstreamfilename);
    exit(-1);
}

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

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

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

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

    int i;
    int streamNum;
#ifdef SQLWINT
    char commandline[256];
    HANDLE su_hSem;
    char UF2_semfile[256];
#else
    int childpid[100];
    char sourcefile[256];
    int su_semid; /* semaphore for controlling split updates */
    key_t su_semkey; /* key to generate semid */
#endif
    if (g_struct->c_l_opt->intStreamNum == 0)
        streamNum = 0;
    else
        streamNum = currentUpdatePair - updatePairStart + 1;

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

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

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

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

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

#ifdef SQLWINT
    sprintf (statement, "perl %s\\tools\\ploaduf2 %d\n", sourcedir, updatePair);
#else
    sprintf (statement, "perl %s/tools/ploaduf2 %d 2", sourcedir, updatePair);

```

```

#endif
    if (system(statement))
    {
        fprintf (stderr, "ploaduf2 failed for UF2, examine UF2.log for cause.
Exiting.\n");
        exit (-1);
    }
    fprintf (outstream, "ploaduf2 finished for UF2.\n");

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

    /* Next we need to get ready to launch a bunch of concurrent processes */
#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);
    }
    /*semctl(su_semid, 0, IPC_RMID, 0);*/ /*mujib*/
#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 */
#ifndef 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);
} /*jenSEM*/
semctl (su_semid, 0, IPC_RMID, 0);
#else
// for (i = 0; i < split_deletes; i++) //DJD Waits forever.....
for (i = 0; i < concurrent_deletes; i++)
{
    if (verbose)
    {
        fprintf(stderr, "About to wait again ...Sets to wait for %d\n",
            split_deletes - i);
        fprintf(stderr, "About to wait again ...Sets to wait for %d\n",
            concurrent_deletes - i);
    }
    if (WaitForSingleObject(su_hSem, INFINITE) == WAIT_FAILED)
    {
        fprintf(stderr,
            "WaitForSingleObject (su_hSem) failed on set %d, error: %d,
quitting\n",
            i, GetLastError());
        exit(-1);
    }
    if (! CloseHandle(su_hSem))
    {
        fprintf(stderr, "Close Sem failed - Last Error: %d\n", GetLastError());
        /* no exit here */
    }
}
#endif

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

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

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

/*-----*/
/*      General semaphore function.              */
/*-----*/

```

```

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

    if (semop(semid, &sembuf, 1) < 0)
    {
        fprintf(stderr, "ERROR*** sem_op 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=0, found=0; /* kjd715 */
    char input_ln[256] = "\0"; /* kjd715 */
    char tag[128] = "\0"; /* kjd715 */

    strcpy(run_dir, g_struct->run_dir);
    sprintf(delim, "%s", env_tpcd_path_delim);
    strcpy(time_stamp, g_struct->file_time_stamp);
    /* kjd715 */
    if (g_struct->stream_list == NULL)
    {
        if ((g_struct->stream_list =
            fopen(g_struct->c_1_opt->infile, READMODE)) == NULL)
        {
            fprintf(stderr, "\nThe input file could not be opened.");
            fprintf(stderr, "Make sure that the filename is correct.\n");
            exit(-1);
        }
    }
    found = 0;
    do {
        fscanf(g_struct->stream_list, "%n%[^n]\n", input_ln);
        if (strstr(input_ln, "--#TAG") == input_ln)
        {
            found = 1;
            strcpy(tag, (input_ln + sizeof("--#TAG")));
            if (strncmp(tag, "UF", 2) == 0)
                qnum = atoi(tag+2)*(-1);
            else if (strncmp(tag, "Q", 1) == 0)
            {
                /* for query 15a the 'a' must be trimmed */
                /* off before converting to integer */
                if (strlen(tag) > 3)
                    tag[3] = '\0';
                qnum = atoi(tag+1);
            }
        }
    }

    if (feof(g_struct->stream_list))
        found = 1;

    } while (!found);
    /*
    if ((g_struct->stream_list =

```

```

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

    fscanf(g_struct->stream_list, "%d", &qnum);
    /*
    /* kjd715 */

    switch (g_struct->c_l_opt->intStreamNum)
    {
    case -1: /* 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 tlg */

/* 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 tlg */

        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 :sqllda;
    if (sqllda.sqlcode == 100)
    {
        sqlcode = sqllda.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_sqllda(sqllda,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) && (sqlcode != -1415))
    {
        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);
    else
        fprintf(outstream, "\n\nNumber of rows retrieved is: %6d",
            s_info_ptr->max_rows_fetch);
#endif
} /* @d28763 tlg */

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%sstrcntuf.%s", g_struct->run_dir,
        env_tpcd_path_delim, g_struct->file_time_stamp);
break;
case (3):
case (4):
    /* update power function stream */
    sprintf(file_name, "%s%spsstrcntuf.%s", g_struct->run_dir,
            env_tpcd_path_delim, g_struct->file_time_stamp);
    break;
case (1):
    /* power query stream */
    sprintf(file_name, "%s%spsstrcnt%d.%s", g_struct->run_dir,
            env_tpcd_path_delim,
            g_struct->c_l_opt->intStreamNum, g_struct->file_time_stamp);
    break;
case (0):
    /* throughput query stream */
    sprintf(file_name, "%s%sstrcnt%d.%s", g_struct->run_dir,
            env_tpcd_path_delim,
            g_struct->c_l_opt->intStreamNum, g_struct->file_time_stamp);
    break;
}

#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);

/* connect reset used to only be done in the semaphore control
   block below, to the best of my knowledge that was incorrect.
   jregier 03/09/30
   */
EXEC SQL CONNECT RESET;

/* 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
*/

#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 tjc */
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);
    }
    /*semctl(semid, 0, IPC_RMID, 0);*/ /* mujib */
}
else
/* THROUGHPUT */
{
    /* TRY TO CREATE IT USING EXCL MODE */
    /* cmgarcia */
    while ( ( semid =
semget(semkey, 1, IPC_CREAT|IPC_EXCL|S_IRUSR|S_IWUSR)) < 0)
    {
        if (errno == EEXIST)
        {
            /* IT ALREADY EXISTS */
            if (verbose)
            {
                fprintf(stderr,
                    "Throughput
can't get initial semaphore! semget failed errno = EEXIST...retrying\n");
            }
            errno = 0;
        }

        /* GET THE SEMAPHORE THAT
ALREADY EXISTS */
        if ( ( semid =
semget(semkey, 1, S_IRUSR|S_IWUSR)) < 0)
        {
            fprintf(stderr,
                "Throughput
can't get (no create) initial semaphore! semget failed errno = %d\n",
                errno);
            exit(1);
        }

        /* REMOVE THE SEMAPHORE */
        if (semctl(semid, 1, IPC_RMID) <
0)
        {
            fprintf(stderr,
                "Throughput
can't remove initial semaphore! semget failed errno = %d\n",
                errno);

```

```

        }
        }
    }
    else
    {
        fprintf(stderr,
            "Throughput can't get
initial semaphore! semget failed errno = %d\n",
            errno);
        exit(1);
    }
    /* IF WE COULDN'T TRY AGAIN */

/* jlr
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);
}
*/

/* semctl(semid,0,IPC_RMID,0);*/ /* mujib */

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 semaphone for %d streams\n",
            g_struct->c_l_opt->intStreamNum);
        exit(1);
    }
    /*jenSEM*/
    fprintf(stderr,"finished waiting on stream semaphore! Ready to run
updates!\n");
    semctl(semid,0,IPC_RMID,0); /* we've finished waiting, now */
    /* remove the semaphore */

#endif
}

void runpower_wait(struct global_struct *g_struct, int sem_num)
{
    char semfile[150];
#ifdef SQLWINT
    HANDLE hSem;

    if (sem_num == 1)
        strcpy (semfile, g_struct->sem_file);
    else
        strcpy (semfile, g_struct->sem_file2);

#else /* AIX */
    int      semid;          /* semaphore for controlling UFs*/
    key_t     semkey;        /* key to generate semid */

    strcpy (semfile, g_struct->update_num_file);

#endif

    if (g_struct->c_l_opt->update == 1)
        fprintf(stderr,"querystream waiting for update stream (UF1) to signal
semaphore based on %s\n", semfile);
    else
        fprintf(stderr,"updatestream (UF2) waiting on querystream semaphore to signal
semaphore based on %s\n", semfile);

#ifdef SQLWINT
    hSem = open_semaphore(g_struct, sem_num);
    if (verbose)
        fprintf(stderr,"Runpower queries about to wait ...\n");
    if (WaitForSingleObject(hSem, INFINITE) == WAIT_FAILED)
    {
        fprintf(stderr,
            "WaitForSingleObject (hSem) failed on stream 0, error: %d, quitting\n",
            GetLastError());
        exit(-1);
    }
    if (! CloseHandle(hSem))
    {
        fprintf(stderr, "Close Sem failed - Last Error: %d\n", GetLastError());
        /* no exit here */
    }
}
#else
    semid = open_semaphore(g_struct);

    /* call the sem_op routine to decrement the semaphore by */

```

```

/* however many streams .... by calling this function with*/
/* a negative number, this stream is forced to wait until */
/* the semaphore gets back to 0 */
/* aix semaphores start at 0, not 1, so sem_num -1 is used */
if (sem_op(semid, sem_num - 1, -1) != 0)
{
    /*jenSEM*/
    fprintf(stderr,
        "Failure to wait on runpower 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

#ifdef
    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

```

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.
*
*****
*****/
#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 tjg */
extern void dumpCa(struct sqlca*); /*kmw*/
extern int sem_op (int semid, int semnum, int value);
extern char *get_time_stamp(int form, Timer_struct *timer_pointer); /*
TIME_ACC jen */

/*****
*****/
/* Declare the SQL host variables. */
/*****
*****/
EXEC SQL BEGIN DECLARE SECTION;
char UF_dbname[9] = "\0";
char UF_userid[9] = "\0";
char UF_passwd[9] = "\0";
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;

#ifdef SQLWINT
int su_semId;
key_t su_semkey;
#else
HANDLE su_hSem;
char UF1_semfile[256];
#endif

char myoutstreamfile[256];
FILE *myoutstream;

strcpy(UF_dbname, dbname);
strcpy(UF_userid, userid);
strcpy(UF_passwd, passwd);

/* Get ready to start logging diagnostic output */
sprintf (myoutstreamfile, UF1OUTSTREAMPATTERN, env_tpcd_tmp_dir,
PATH_DELIM,
updatePair, i);
if ( (myoutstream = fopen (myoutstreamfile, WRITEMODE)) == NULL)
{
fprintf (stderr, "\nThe output file '%s' for update pair %d set %d could not be
opened. runUF1_fn\n",
myoutstreamfile, updatePair, i);
rc=-1;
goto UF1_exit;
}
outstream=myoutstream; /* initialize outstream for error_check dxxxxhar*/

fprintf (myoutstream, "\nUF1 for update pair %d set %d starting at %*.s\n",
updatePair, i,
T_STAMP_1LEN, T_STAMP_1LEN, /* TIME_ACC jen*/
get_time_stamp(T_STAMP_FORM_1, (Timer_struct *)NULL)); /*
TIME_ACC jen*/

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

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

/* Establish a connection to the database */
if (!strcmp(userid, "\0")) /** No authentication provided **/
EXEC SQL CONNECT TO :UF_dbname;
else
EXEC SQL CONNECT TO :UF_dbname USER :UF_userid USING
:UF_passwd;
error_check();
if (sqlca.sqlcode < 0)
{
rc=-1;
goto UF1_exit;
}

/* Start processing each chunk in my range */
#ifdef UF1DEBUG
fprintf (myoutstream, "Before loop_a startChunk = %d, stopChunk = %d\n",
startChunk, stopChunk);
fflush(myoutstream);
#endif

```

```

for ( UF_chunk = startChunk; UF_chunk <= stopChunk; UF_chunk++ )
/*jenCI*/
{
    /* 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)
    {
        sqlcode = 0;
#ifdef UF1DEBUG
        fprintf(myostream, "in loop before orders exec sql\n");
        fflush(myostream);
#endif
        EXEC SQL INSERT INTO TPCD.ORDERS
            SELECT
O_ORDERKEY,O_CUSTKEY,O_ORDERSTATUS,O_TOTALPRICE,

O_ORDERDATE,O_ORDERPRIORITY,O_CLERK,O_SHIPPRIORITY,O_CO
MMENT
            FROM TPCDTEMP.ORDERS_NEW
            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 */
            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_QUANTITY,
L_EXTENDEDPRI, L_DISCOUNT, L_TAX,

L_RETURNFLAG,L_LINESTATUS,L_SHIPDATE,L_COMMITDATE,L_REC
EIPDATE,
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);*/

        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;
            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.*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 tjg */

            goto UF1_exit;
        }

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

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

/* we used the first flat file to generate the semaphore key */

#ifndef 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 */
    printf (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, GetLastError: %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*/

#ifndef SQLWINT
    int su_semid; /* semaphore for controlling split updates*/
    key_t su_semkey; /* key to generate semid */
#else
    HANDLE su_hSem;
    char UF2_semaphore[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=myoutstream; /* initialize outstream for error_check dxxxxhar*/

#ifdef UF2DEBUG
    fprintf(myoutstream, "RunUF2 Called %d %d %d\n",
            updatePair, thisConcurrentDelete, numChunks);
    fflush(myoutstream);
#endif
    fprintf( myoutstream,
            "\nUF2 for update pair %d set %d starting at %s\n",
            updatePair, thisConcurrentDelete, T_STAMP_1LEN,T_STAMP_1LEN,
/* TIME_ACC jen*/
            get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL)); /*
TIME_ACC jen*/

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

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

#ifdef UF2DEBUG
    fprintf( myoutstream, "after connect startchunk= %d, EndChunk = %d\n",
            startChunk, startChunk+numChunks);
    fflush(myoutstream);
#endif

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

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

#ifdef UF2DEBUG
        fprintf( myoutstream, "Chunk = %d\n", UF_chunk);
        fflush(myoutstream);
#endif
        while (sqlcode == SQL_RC_E911 && maxwait <= MAXWAIT && rc == 0)
        {

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

            EXEC SQL DELETE FROM TPCD.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( myoutstream,
                    "\nA deadlock detected while deleting from LINEITEM: update pair %d
set %d chunk %d. Retrying.\n",

```

```

            updatePair, thisConcurrentDelete, UF_chunk);
            dumpCa(&sqlca);
            SleepSome(UF_DEADLOCK_SLEEP);
            maxwait++; /* jen DEADLOCK */
        }
        else if (sqlcode < 0)
        {
            fprintf( myoutstream, "\n%s\n", stmt_str.data);
            fprintf( myoutstream, "\nsqlcode %d occurred deleting from
TPCD.LINEITEM\n", sqlca.sqlcode);
            dumpCa(&sqlca);
            fprintf( myoutstream,
                "for update pair number %d set %d chunk %d. Exiting\n",
                updatePair, thisConcurrentDelete, UF_chunk);
            rc=-1;
        }
        else
        {
            /* accumulate the number of row deleted */
            savedDeletedLineitems = sqlca.sqlerrd[2]; /*kbs*/
        }

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

        /* delete the 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(myostream, " 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(myostream, " 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( myostream,
" 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(myostream, "after loop\n");
fflush(myostream);
#endif
/* report the number of row deleted */
fprintf(myostream, "%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(myostream, "%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(myostream, "# 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(myostream);

/* 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 == ENOENT)
sleep(2);
else {
fprintf(stderr,"UF2 update stream %d: semget failed errno = %d\n",
updatePair, errno);
exit(1);
}
}
if (sem_op (su_semid, 0, 1) != 0 ) /*jenSEM*/
{ /*jenSEM*/
fprintf(stderr,"Failure to increment 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 */
}

```

Appendix E: ACID Transaction Source Code

acid.h

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

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

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

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

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

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

struct update_struct {
    int qnum;
};

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

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

/*
** in acid.sqc
*/

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

```
char del(void);

#ifdef SQLWINT
void sleep (int sec);
#endif
```

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 timer;

void gettimeofday( struct timeval *tv, struct timezone *tz)
{
    ftime(&timer);
    tv->tv_sec = timer.time;
    tv->tv_usec = timer.millitm * 1000;
    tz->dummy = 0;
}
```

```

}
#endif

/*-----*/
/*      acidQ                      */
/*-----*/

int acidQ (struct acidQ_struct *acid)
{
    time_t timeT;
    FILE *out;
    char out_fn[50];
    struct timeval tv;
    struct timezone tz;
    int mypid;
    int rc = 0;

    EXEC SQL BEGIN DECLARE SECTION;
    sqlint32  okey;
    sqlint32  lEprice;
    double  eprice;
    EXEC SQL END DECLARE SECTION;

    okey = acid->o_key;

    /* mypid = getpid(); */
    mypid = acid->tag;

    sprintf(out_fn, "%s%cacidQ.out.%d",getenv("TPCD_TMP_DIR"),del(),mypid);
    out=fopen(out_fn,"a");
    if (out == NULL)
    {
        fprintf(stderr, "ERROR input file %s could not be appended to!!\n",out_fn);
    }

    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out, "\n----- START of acidQ tag: %d ----- \n\n", mypid);
    fprintf(out, "acidQ tag: %d, begin transaction time: (%us %06uu) %s",
        mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    fprintf(out, "okey: %d\n", okey);

    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out, "acidQ tag: %d, before read of LINEITEM: (%us %06uu) %s",
        mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));

    /*
    ** use the same sql code as used in the consistsql.pl to
    ** run the consistency acid queries. Note we assign an long int
    ** to lEprice (we make it 10s of pennies by * 1000). Then divide
    ** by 1000.0 and cast it to a double (eprice) for printing
    */

    EXEC SQL
    SELECT
        INTEGER(DECIMAL(SUM(DECIMAL(INTEGER(INTEGER(DECIMAL
        (INTEGER(100*DECIMAL(L_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_EXTENDEDPRI*(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_l_eprice; /* use ast_eprice for printout to be consistent*/

gettimeofday(&tv, &tz);
time(&timeT);
fprintf(out, "AST_ACID tag: %d, after read of LINEITEM: (%us %06uu) %s",
  mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
fprintf(out, "sum(l_extendedprice): %0.3f\n",
  ast_eprice);

EXEC SQL COMMIT;
if (sqlca.sqlcode != 0) {
  rc = sqlca.sqlcode;
  fprintf(out, "ast_acidQ **ERROR** sqlcode = %d\n", sqlca.sqlcode);
  sqlerror("ast_acidQ: COMMIT", &sqlca);
  goto Qerror;
}
acid->l_extendedprice = ast_eprice;

rc = 0;
goto Qexit;

Qerror:
EXEC SQL rollback work;
if (sqlca.sqlcode != 0) sqlerror("ast_acidQ: ROLLBACK FAILED", &sqlca);

Qexit:
fprintf(out, "\n----- END of ast_acidQ tag: %d ----- \n\n", mypid);
fflush(out); fclose(out);
return(rc);
}
/*-----*/
/*      acidT                                */
/*-----*/
int acidT (struct acidT_struct *acid)
{

  time_t timeT;
  FILE *out;
  char out_fn[50];
  struct timeval tv;
  struct timezone tz;
  int mypid;
  int rc = 0;

  EXEC SQL BEGIN DECLARE SECTION;
  sqlint32  o_key, l_key, delta;
  sqlint32  l_partkey, l_suppkey;
  double    l_quantity, l_tax, l_discount, l_extendedprice;
  double    o_totalprice;
  double    new_quantity, rprice, cost, new_extprice, new_ototal, ottotal;
  EXEC SQL END DECLARE SECTION;

  EXEC SQL DECLARE l_cursor CURSOR FOR
    SELECT l_partkey, l_suppkey, l_quantity,
           l_tax, l_discount,
           l_extendedprice
    FROM tpcd.lineitem
    WHERE l_orderkey = :o_key
    AND l_linenum = :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%acidT.out.%d", getenv("TPCD_TMP_DIR"), del(), mypid);
  out = fopen(out_fn, "a");
  gettimeofday(&tv, &tz);
  time(&timeT);
  fprintf(out, "\n----- START of acidT tag: %d ----- \n\n", mypid);
  fprintf(out, "acidT tag: %d, begin transaction time: (%us %06uu) %s",
    mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
  fprintf(out, "o_key: %d\tl_key: %d\tdelta: %d\n", o_key, l_key, delta);
}
#ifdef DEBUG
  printf("o_key: %d\tl_key: %d\tdelta: %d\n", o_key, l_key, delta);
#endif

retry_tran:

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

EXEC SQL OPEN l_cursor;
if (sqlca.sqlcode != 0) {
  if (sqlca.sqlcode == DEADLOCK) goto retry_tran;
  rc = sqlca.sqlcode;
  if (acid->logging) {
    fprintf(out, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
  } else {
    fprintf(stderr, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
  } /* endif */
  sqlerror("acidT: OPEN l_cursor", &sqlca);
  goto Terror;
}

EXEC SQL FETCH l_cursor INTO
  :l_partkey, :l_suppkey, :l_quantity, :l_tax,
  :l_discount, :l_extendedprice;
if (sqlca.sqlcode != 0) {
  if (sqlca.sqlcode == DEADLOCK) goto retry_tran;
  rc = sqlca.sqlcode;
  if (acid->logging) {
    fprintf(out, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
  } else {
    fprintf(stderr, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
  } /* endif */
  sqlerror("acidT: FETCH l_cursor", &sqlca);
  goto Terror;
}

#ifdef DEBUG
  printf("l_quantity = %0.3f\n", l_quantity);
  printf("l_tax = %0.3f\n", l_tax);
  printf("l_discount = %0.3f\n", l_discount);
  printf("l_extendedprice = %0.3f\n", l_extendedprice);
#endif

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

```



```

        fprintf(out, "l_partkey: %d l_suppkey: %d l_quantity: %0.3f\nl_tax: %0.3f\nl_discount: %0.3f l_extendedprice: %0.3f\n",
            l_partkey, l_suppkey, l_quantity, l_tax, l_discount, l_extendedprice);
    }

    rprice = TRUNC2( l_extendedprice/l_quantity );
    cost = TRUNC2( rprice * delta );
    new_extprice = l_extendedprice + cost;
    new_quantity = l_quantity + delta;

#ifdef DEBUG
    printf("rprice = %0.3f\n", rprice );
    printf("cost = %0.3f\n", cost );
    printf("new_extprice = %0.3f\n", new_extprice );
    printf("new_quantity = %0.3f\n", new_quantity );
#endif

EXEC SQL UPDATE tpcd.lineitem
    SET l_extendedprice = :new_extprice,
        l_quantity = :new_quantity
    WHERE CURRENT OF l_cursor;

if (sqlca.sqlcode != 0) {
    if (sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
        fprintf(out, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } else {
        fprintf(stderr, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: UPDATE l_cursor", &sqlca);
    goto Terror;
}

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

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

EXEC SQL OPEN o_cursor;
if (sqlca.sqlcode != 0) {
    if (sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
        fprintf(out, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } else {
        fprintf(stderr, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: OPEN o_cursor", &sqlca);
    goto Terror;
}

EXEC SQL FETCH o_cursor INTO :o_totalprice;
if (sqlca.sqlcode != 0) {
    if (sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
        fprintf(out, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    }
    else
    {
        fprintf(stderr, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    }
}

```

```

    }
    sqlerror("acidT: FETCH o_cursor", &sqlca);
    goto Terror;
}

#ifdef DEBUG
    printf("o_totalprice = %0.3f\n", o_totalprice);
#endif

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

#ifdef DEBUG
{
    double zeroone= l_extendedprice * (1.0- l_discount);
    double zeroonetimes= (l_extendedprice * (1.0- l_discount))*100.0;
    double firstone = TRUNC2(l_extendedprice * (1.0-l_discount));
    double notone= TRUNC2( l_extendedprice * (1.0-l_discount)) * (1.0+l_tax);
    double secondone= TRUNC2( TRUNC2( l_extendedprice * (1.0-l_discount)
        * (1.0+l_tax) );
    printf("firstone= %f\n", firstone);
    printf("zeroone= %f\n", zeroone);
    printf("zeroonetimes= %f\n", zeroonetimes);
    printf("notone= %f\n", notone);
    printf("secondone= %f\n", secondone);
}
#endif

ototal = o_totalprice -
    TRUNC2( TRUNC2( l_extendedprice * (1-l_discount) ) * (1+l_tax) );
new_ototal = TRUNC2( new_extprice * (1.0-l_discount) );
new_ototal = TRUNC2( new_ototal * (1.0+l_tax) );
new_ototal = ototal + new_ototal;

#ifdef DEBUG
    printf("o_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_suppkey, :o_key, :l_key, :delta, CURRENT_TIMESTAMP);
if (sqlca.sqlcode != 0) {
    if (sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
        fprintf(out, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } else {
        fprintf(stderr, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: INSERT INTO history", &sqlca);
    goto TError;
}

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

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

switch (acid->termination) {
case 1:
    {
        if (acid->logging)
        {
            gettimeofday(&tv, &tz);
            time(&timeT);
            fprintf(out, "acidT tag: %d, wait before COMMIT: (%us %06uu) %s",
                mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
        }
        sleep(60);
    }
case 0:
    if (acid->logging) {
        gettimeofday(&tv, &tz);
        time(&timeT);
        fprintf(out, "acidT tag: %d, immediately before COMMIT: (%us %06uu)
%s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
EXEC SQL CLOSE L_CURSOR;
if (sqlca.sqlcode != 0) {
    if (sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;

```

```

    if (acid->logging) {
        fprintf(out, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } else {
        fprintf(stderr, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: CLOSE L_CURSOR", &sqlca);
    goto TError;
}
EXEC SQL CLOSE O_CURSOR;
if (sqlca.sqlcode != 0) {
    if (sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
        fprintf(out, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } else {
        fprintf(stderr, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: CLOSE O_CURSOR", &sqlca);
    goto TError;
}
EXEC SQL COMMIT;
if (sqlca.sqlcode != 0) {
    if (sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
        fprintf(out, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } else {
        fprintf(stderr, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: COMMIT", &sqlca);
    goto TError;
}
if (acid->logging) {
    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out, "acidT tag: %d, after COMMIT: (%us %06uu) %s",
        mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
}
break;
case 3:
    if (acid->logging) {
        gettimeofday(&tv, &tz);
        time(&timeT);
        fprintf(out, "acidT tag: %d, wait before ROLLBACK: (%us %06uu) %s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    sleep(60);
case 2:
    if (acid->logging) {
        gettimeofday(&tv, &tz);
        time(&timeT);
        fprintf(out, "acidT tag: %d, immediately before ROLLBACK: (%us %06uu)
%s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
EXEC SQL CLOSE L_CURSOR;
if (sqlca.sqlcode != 0) {
    if (sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
        fprintf(out, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } else {
        fprintf(stderr, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: CLOSE L_CURSOR", &sqlca);
    goto TError;
}
EXEC SQL CLOSE O_CURSOR;
if (sqlca.sqlcode != 0) {
    if (sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
        fprintf(out, "acidT **ERROR** sqlcode = %d\n", sqlca.sqlcode);
    } else {

```

```

        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: CLOSE O_CURSOR", &sqlca);
    goto Terror;
}
EXEC SQL rollback work;
if (sqlca.sqlcode != 0) {
    if (sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
    } else {
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: ROLLBACK", &sqlca);
    goto Terror;
}
if (acid->logging) {
    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"acidT tag: %d, after ROLLBACK: (%us %06uu) %s",
        mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
}
break;
}

acid->l_partkey = l_partkey;
acid->l_suppkey = l_suppkey;
acid->l_quantity = l_quantity;
acid->l_tax = l_tax;
acid->l_discount = l_discount;
acid->l_extendedprice = l_extendedprice;
acid->o_totalprice = o_totalprice;

rc = 0;
goto Texit;

Terror:
    EXEC SQL CLOSE L_CURSOR;
    EXEC SQL CLOSE O_CURSOR;
    EXEC SQL rollback work;
    if (sqlca.sqlcode != 0) sqlerror("acidT: ROLLBACK FAILED", &sqlca);

Texit:
    if (acid->logging) {
        fprintf(out,"n----- END of acidT tag: %d -----n\n",mypid);
        fflush(out);fclose(out);
    }
    return(rc);
}

/*-----*/
/*      updateQ      */
/*-----*/
int updateQ (struct update_struct *us)
{
    FILE *out;
    time_t timeT;
    struct timeval tv;
    struct timezone tz;
    int qnum;
    int rc = 0;
    int i;
    int secs2sleep;
    char buff[256];
    struct acidtype {int logging;} a, *acid;

    EXEC SQL BEGIN DECLARE SECTION;
    double  acctbal;
    double  discount;
    double  price;
    sqlint32 availqty;
    sqlint32 size;
    EXEC SQL END DECLARE SECTION;

```

```

    qnum = us->qnum;

    acid = &a;
    acid->logging = 1;

    sprintf(buff, "%s%cupdate.out",getenv("TPCD_TMP_DIR"),del());
    out=fopen(buff,"a");

    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"n----- START of update -----n\n");
    fprintf(out, "update query number: %d, begin transaction time: (%us %06uu)
%s",
        qnum, tv.tv_sec, tv.tv_usec, ctime(&timeT));

    sqlca.sqlcode = 0;
    discount = 0.25;
    price = 5000.50;
    acctbal = 1000.00;
    availqty = 10;
    size = 5;

    for (i=1; i <= 2; i++) {
        gettimeofday(&tv, &tz);
        time(&timeT);
        fprintf(out,"update query number: %d, pass %d, immediately before
UPDATE: (%us %06uu) %s",
            qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));

        switch (qnum)
        {
            case 1:
            {
                EXEC SQL
                    UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT +
:discount
                    WHERE L_ORDERKEY IN (326,512,928,995);
                if (sqlca.sqlcode != 0) {
                    rc = sqlca.sqlcode;
                    if (acid->logging)
                    {
                        fprintf(out,"update query number: %d, pass %d, **ERROR** sqlcode =
%d\n",
                            qnum, i, sqlca.sqlcode);
                    }
                    else
                    {
                        fprintf(stderr,"update query number: %d, pass %d, **ERROR** sqlcode
= %d\n",
                            qnum, i, sqlca.sqlcode);
                    }
                    sqlerror("update query number 1", &sqlca);
                    goto Uerror;
                }
                discount = discount * (-1);
                secs2sleep = 300;
                break;
            }
            case 2:
            {
                EXEC SQL
                    UPDATE TPCD.SUPPLIER set S_ACCTBAL = S_ACCTBAL +
:acctbal
                    WHERE S_NAME in
('Supplier#000000647','Supplier#00000070','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%acid.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 < 8; 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 */

```

makefile

```

DBNAME = $(TPCD_QUAL_DBNAME)

INCLUDE = $(HOME)/sqllib/include

#CFLAGS = -I$(INCLUDE) -g -Dpascal= -DLINT_ARGS \
#         -Dfar= -D_loadadds= -DSQLA_NOLINES -qflag=i:i -qlanglvl=ansi

#LFLAGS = -lm -lcurses -ls -ll -ly -liconv -lbsd
CFLAGS = -I$(INCLUDE) -g -Dpascal= -DLINT_ARGS \
         -DSQLA_NOLINES -qflag=i:i -qlanglvl=ansi
# .. sun -DSQLA_NOLINES

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

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

CC = cc

HDR = acid.h
C = mainacid.c
SQC = acid.sqc
SRC = $(HDR) $(C) $(SQC)
OBJ = acid.o
EXEC = mainacid

TARGET = $(EXEC) tsec

.SUFFIXES: .o .c .sqc .bnd

```

```

.c.o: $(CC) -c $(CFLAGS)

all: $(TARGET)

mainacid: $(SRC) $(OBJ) mainacid.o
        $(CC) -o $@ $(CFLAGS) $(OBJ) mainacid.o $(LIB) $(LFLAGS)

acid.c: acid.sqc $(HDR)
        - db2 connect to $(DBNAME); \
        db2 prep acid.sqc BINDFILE ISOLATION RR NOLINEMACRO

PACKAGE; \
        db2 bind acid.bnd GRANT PUBLIC; \
        db2 connect reset; \
        db2 terminate

acid.o: acid.c
        $(CC) $(CFLAGS) -c acid.c -o acid.o

tsec: tsec.c
        $(CC) $(CFLAGS) $(LFLAGS) -o tsec tsec.c

clean:
        rm -f *.o *.bnd $(EXEC) tsec
        rm -f acid.c

```

Appendix F: Price Quotations



Protect Your Data - Grow Your Business

To:
Attention:
Phone:
Fax:
Email:

From: Alan Powers
Phone: (248)223-1020 x344
Fax: (248)223-1026
Email: apowers@compsat.com

QUOTE # : 2W346_120104
DATE: December 13, 2004

IBM x346 Configuration

Part No.	Description	Qty	List Price		Compsat Discounted Price	
			(per unit) US Dollar	(quantity x unit price) US Dollar	(per unit) US Dollar	(quantity x unit price) US Dollar
x346 SERVER						
884041U	x346, 1xXeon 3.6GHz/800MHz, 1MB, 2x512MB, O/Bay	1	\$3,559.00	\$3,559.00	\$2,989.56	\$2,989.56
21P2078	3 YR onsite repair 24x7x4 hour (x346)	1	\$600.00	\$600.00	\$528.00	\$528.00
73P3522	1GB (2x512GB) PC-3200 DDR ECC SDRAM RDIMM	3	\$399.00	\$1,197.00	\$359.10	\$1,077.30
13N0685	3.6GHz Xeon/800MHz 1M L2 Processor Upgrade	1	\$1,429.00	\$1,429.00	\$1,286.10	\$1,286.10
71P8642	IBM ServeRAID-7k Adapter	1	\$449.00	\$449.00	\$404.10	\$404.10
633147N	E54 15" Colour Monitor (Stealth Grey)/MPRII	1	\$149.00	\$149.00	\$134.10	\$134.10
30L9183	3 YR onsite exch. 24x7x4 hour (E54 Monitor)	1	\$90.00	\$90.00	\$79.20	\$79.20
31P7415	IBM Preferred Pro Full-size Keyboard PS/2	1	\$29.00	\$29.00	\$26.10	\$26.10
28L3673	IBM Sleek 2-Button Mouse	1	\$14.95	\$14.95	\$13.46	\$13.46
EXP400(s)						
17331RU	IBM EXP400 Storage Expansion Enclosure	1	\$3,099.00	\$3,099.00	\$2,603.16	\$2,603.16
41L2768	3 YR onsite repair 24x7x4 hour (EXP400)	1	\$760.00	\$760.00	\$668.80	\$668.80
90P1318	IBM 36.4GB 15K Hot-Swap SCSI HDD	20	\$349.00	\$6,980.00	\$314.10	\$6,282.00
TOTAL =				\$18,355.95	TOTAL =	\$16,091.88

12.33%

25330 Telegraph Road / Suite 200 Raleigh Officentre / Southfield, Michigan 48034
Phone: 248-223-1020 / Fax: 248-223-1026 / www.compsat.com

NOTE:

- This quote may include Compsat Technology consulting and configuration charges.
- Mfg. pricing is out of our control and could change without notice.
- Pricing good for 30 Days from date quoted.

SUSE LINUX AG | Maxfeldstr. 5 | D-90409 Nuernberg

Chris King
IBM eServer xSeries Performance

Fax: 919-486-2327

Approved by:

Peter Woodward
Novell Global Alliances

Your sign

Our Sign

Customer No

Offer No

Date

12/08/04

SUSE LINUX Enterprise Server 9 for x86

Per your request for pricing information for SUSE Linux Enterprise Server 9 for AMD64 and Intel EM64T to be used in your TPC-C benchmark testing, Novell is pleased to provide the following quote. All prices are shown in USD excl. VAT.

Partnumber	Description	Units	Unit Price	Extended Price
00662644456164	SUSE Linux Enterprise Server 9 for X86 and for AMD64 & Intel EM64T 1Server up to 2-CPU 3-Year Upgrade Protection	1	\$870.00	\$870.00
			Total	\$870.00

This offering is based for a machine type x86, AMD 64 or Intel EM64T equipped with up to 2 CPUs, i.e. the standard price for SUSE LINUX Enterprise Server 9 for x86 .

Maintenance Program / Upgrade Protection includes Installation Support.

Prices are valid for 30 days.

The quote is based on Novells NSPP program.

Qualifying customers may take advantage of price savings through the Novell Volume Licensing Programs. For more information, see <http://www.novell.com/licensing/price.html>.

References:

<http://www.novell.com/licensing/indemnity/pricelist.html>

http://www.novell.com/licensing/price/linuxvla_cla.xls

http://support.novell.com/linux/linux_server_support.html

http://support.novell.com/linux/linux_support_order.pdf